J. of Math. and Its Appl.
ISSN: 1829-605X
Vol. 1, No. 1 (2004), 21–29

Studi Komparatif antara Jaringan Syaraf Tiruan Boltzman Machine dan Algoritma Genetika untuk Optimasi Traveling Salesman Problem

M Isa Irawan
Jurusan Matematika
Institut Teknologi Sepuluh Nopember, Surabaya

Abstrak

Traveling Salesman Problem (TSP) dikenal sebagai suatu permasalahan optimasi klasik dan Non Deterministic Polynomial-time Complete (NPC). Permasalahan ini melibatkan seorang salesman yang harus melakukan kunjungan sekali pada semua kota sebelum kembali ke kota awalnya, sampai akhirnya perjalanan itu disebut sempurna. Penyelesaian dari masalah ini adalah mencari nilai optimum yang paling murah, misalkan perjalanan dengan jarak terpendek atau yang mempunyai total harga yang termurah.

Dalam paper ini akan dianalisis penyelesaian TSP dengan JST Boltzman Machine dan Algoritma Genetika. Dari hasil komparasi tersebut ternyata JST Boltzman Machine memberikan hasil lebih baik untuk menyelesaikan masalah TSP. Kata kunci : Jaringan Syaraf Tiruan, Boltzman Machine , Algoritma Genetika, TSP.

Kata kunci: *Jaringan Syaraf Tiruan, Boltzman Machine, Algoritma Genetika, TSP*

1 Pendahuluan

Sejak ditemukannya komputer, perkembangan ilmu dan teknologi yang mendapat dukungan komputer mengalami perkembangan yang luar biasa. Komputer sendiri mengalami evolusi dalam hal kecepatan dan kapasitas penyimpanan sementara, sehingga dapat memproses data yang sangat besar dalam waktu yang relatif singkat. Dengan alasan komputer akan dimanfaatkan untuk suatu komputasi menyelesaikan masalah TSP. TSP adalah suatu masalah optimasi klasik yang bersifat NPC, dimana tidak ada penyelesaian yang paling optimal selain harus mencoba seluruh kemungkinan penyelesaian yang ada. Permasalahan ini melibatkan seorang salesman yang harus melakukan perjalanan antar kota dan setiap kota tidak boleh dikunjungi lebih dari sekali, sehingga perjalanannya dikatakan sempurna.

Berawal dari sinilah dikembangkan metode-metode pemecahan masalah optimasi perjalnan tersebut. Disamping metode-metode optimasi yang sudah demikian berkembang, ditawarkan pula pendekatan alternatif yaitu dengan JST Boltzman Machine dan Algoritmna Genetika.

2 Dasar Teori

Masalah optimasi adalah salah satu masalah yang sering ditemui, dimana sampai saat ini belum ditemukan suatu metode yang dapat menyelesaikannya dengan cepat dan tepat. Untuk permasalahan yang demikian, algoritma probabilistik dapat menjadi alternatif untuk mencari penyelesaian yang optimal. Banyak masalah optimasi yang penyelesaiaannya se-

cara probabilistik. Contoh algoritma probabilistik yang ada adalah Algoritma Genetika dan JST Boltzman Machine, yang mana keduanya termasuk metode *derivative-free optimization* (DFO).

Karakteristik umum dari metode DFO adalah sebagai berikut:

a. *Derivative Freeness.* Metode ini tidak membutuhkan informasi secara derivative fungsional untuk melakukan proses pencarian atau penelusuran terhadap parameter-parameter yang akan meminimumkan atau memaksimumkan suatu fungsi objektif, akan tetapi melakukan evaluasi yang dilakukan secara berulang pada fungsi objektif dan arah pencarian atau penelusuran pada titik berikutnya.

b. *Intuitive Guidelines.* Pedoman yang harus diikuti pada proses pencarian yang berdasarkan pada perkiraan saja. Beberapa teori berdasarkan pada evolusi dan termodinamika.

c. *Slowness.* Walaupun tidak menggunakan derivasi pada komputasinya, pada umumnya metode-metode derivative-free optimization lebih lambat dibandingkan dengan metode derivative based optimization untuk permasalahan dalam bentuk kontinu.

d. *Flexibility.* Derivative Freeness juga berlaku pada fungsi objektifnya. Jadi dimungkinkan penggunaan fungsi objektif yang kompleks sehingga dapat disesuaikan dengan spesifikasi-spesifikasi yang dibutuhkan untuk suatu aplikasi tanpa menggunakan terlalu banyak fungsi pengkodean serta waktu komputasi.

e. *Randomness.* Semua metode yang termasuk derivative free optimization adalah stokastik, artinya metode-metode ini menggunakan pembangkit bilangan acak untuk menentukan arah pencarian atau penelusuran pada titik berikutnya. Dikarenakan salah satu karakteristiknya adalah randomness, maka metode derivative free optimization merupakan global optimization. Karakteristik randomness ini merupakan probabilitas untuk menemukan suatu penyelesaian yang optimal (tidak nol) pada waktu komputasi yang tepat. Hal ini tentunya membutuhkan waktu komputasi yang cukup banyak.

f. *Anality Opacity.* Penyajian secara analitik agak sulit dilakukan pada metode derivative free optimization. Hal ini disebabkan oleh sifat karakteristik yang dimiliki yaitu randomness serta gejala alam yang ikut berpengaruh di dalamnya seperti evolusi dan termodinamika.

2.1 Algoritma Genetika

Teori Algoritma Genetika

Algoritma genetika adalah sebuah algoritma probabilistik dengan intelegensi tinggi yang dapat diaplikasikan pada berbagai permasalahan optimasi [2]. Peletak prinsip dasar sekaligus pencipta algoritma genetika adalah John Holland. Algoritma genetika didasarkan pada proses evolusi makhluk hidup, dimana dalam evolusi tersebut makhluk hidup mengalami seleksi alam untuk bertahan hidup. Bagi yang mampu beradaptasi dengan lingkungannya maka ia akan memiliki peluang hidup yang lebih besar dan melakukan reproduksi. Sedangkan yang tidak mampu akan tereliminasi. Artinya gen dari individu yang paling sesuailah yang mampu berkembang menjadi individu-individu yang lebih banyak untuk generasi berikutnya.

Algoritma genetika banyak menggunakan istilah yang terdapat pada bidang ilmu genetika (biologi). Istilah Biologi yang digunakan dalam algoritma genetika adalah:

Tabel 1: Istilah dalam Algoritma Genetika

Istilah Biologi	Keterangan
Kromosom	Individu yang berupa segmen string yang sudah ditentukan
Gen	Bagian dari string
Loci	Posisi dari gen
Allele	Nilai yang dimasukkan dalam Gen
Phenotype	String yang merupakan solusi akhir
Genotype	Sejumlah string hasil perkawinan yang berpotensi sebagai solusi

Secara umum penerapan Algoritma Genetika akan melalui siklus sederhana yang terdiri dari 4 state, yaitu:

i. Membangun sebuah populasi yang terdiri dari beberapa string.

ii. Mengevaluasi masing-masing string.

iii. Proses seleksi agar didapat string yang paling baik.

iv. Manipulasi genetika untuk menciptakan populasi baru dari string.

Karakteristik Algoritma Genetika.
Ada dua hal yang menunjukkan karakteristik algoritma genetika dalam proses pencarian solusi yang optimal, yaitu eksplorasi dan eksploitasi. Eksplorasi berarti algoritma genetika melakukan proses pencarian solusi potensial dalam suatu ruang pencarian, sedangkan eksploitasi dimaksudkan behwa algoritma genetika menggunakan sifat atau informasi yang ada pada individu-individu dalam populasi untuk mendapatkan solusi yang optimal. Michalewicz [2] memberikan perbandingan antara metode *hill climbing*, pencarian acak, dan algoritma genetika. Pencarian hill climbing adalah contoh pencarian yang mengeksploitasi solusi terbaik tetapi mengabaikan eksplorasi daerah. Pencarian acak adalah contoh pencarian yang mengeksplorasi daerah tetapi mengabaikan eksploitasi. Sedangkan algoritma genetika mencari solusi optimal dengan menyeimbangkan eksplorasi daerah pencarian dan eksploitasi solusi terbaik.

Pembangkitan Populasi.
Algoritma genetika dalam melakukan pencarian dengan membangkitkan suatu polulasi yang merupakan sekumpulan nilai awal. Ukuran populasi (*pop_size*) mempengaruhi unjuk kerja yang baik dan keefektifan algoritma genetika. Algoritma genetika dengan populasi yang kecil biasanya unjuk kerjanya buruk karena populasi tidak menyediakan cukup materi untuk mencakup ruang persoalan. Populasi yang lebih besar dibutuhkan untuk merepresentasikan keseluruhan ruang persoalan.

Daerah Pengkodean dan Daerah Solusi. Algoritma genetika bekerja pada daerah pengkodean dan daerah solusi. Operasi genetika (tukar silang dan mutasi) bekerja pada daerah pengkodean (kromosom) sedangkan proses evaluasi bekerja pada daerah solusi atau populasi. Langkah pertama kali yang dilakukan dalam penggunaan algoritma genetika adalah melakukan pengkodean terhadap permasalahan yang diselesaikan. Pada algoritma ini dalam mengasumsikan sebuah solusi untuk sebuah persoalan dimungkinkan dengan diwakili satu set parameter. Parameter-parameter ini dinamakan gen yang berisi nilai-nilai. Allele (representasi) yang bersatu membentuk string (kromosom). Selanjutnya beberapa kromosom berkumpul membentuk populasi. Dari sebuah populasi inilah algoritma genetika memulai untuk melakukan pencarian.

Fungsi Evaluasi (*fitness function*)
Dalam algoritma genetika sebuah fungsi evaluasi (*fitness function*) harus dirancang untuk masing-masing masalah yang diselesaikan. Dengan memberikan kromosom tertentu, fungsi fitness menjadikannya salah satu calon fitness atau hasil. Kondisi fitness dari sebuah individu ditentukan oleh penampilan dari phenotype atau dapat dikatakan bahwa phenotype nmerupakan kromosom yang menunjukkan hasil. Dari sini dapat disimpulkan bahwa dari genotype yang berupa kromosom dapat dibentuk phenotype dengan menggunakan fungsi fitness.

Seleksi
Pada tahap ini akan ditentukan individu-individu mana yang akan dipilih untuk membentuk populasi baru. Ada tiga hal penting pada tahap seleksi:

a. *Daerah Sampling.* Prosedur seleksi dilakukan untuk membuat generasi baru dari induk pada suatu populasi yang terdapat pada daerah sampling. Daerah sampling dipengaruhi oleh dua faktor yaitu ukuran dan isi.

b. *Mekanisme Sampling.* Mekanisme sampling menyelesaikan masalah bagaimana memilih kromosom dari daerah sampling.

c. *Probabilitas Seleksi.* Pencarian oleh algoritma genetika yang dilakukan pada tahap selksi sering mengarah pada kekonvergenan dini yaitu kecenderungan sebagian kromosom untuk mendominasi pada proses seleksi awal generasi sehingga kompetisi antar kromosom menjadi kecil pada generasi berikutnya. Alternatif untuk mengatasi masalah di atas adalah dengan penskalaan (*scaling*) yaitu dengan memetakan nilai evaluasi (*fitness value*) pada suatu nilai riil positif kemu-

dian menentukan kromosom terpilih dengan probabilitas ini dan mekanisme rangking (*ranking mechanism*) yaitu dengan menggunakan rangking untuk menentukan probabilitas terpilih.

Operator Genetika.
Operator genetika memegang fungsi eksploitasi dari kromosom-kromosom dalam populasi dengan menghasilkan keturunan dari induk. Keturunan dihasilkan dengan cara melakukan pertukaran informasi atau gen-gen dari tiap kromosom terpilih, dengan harapan akan menghasilkan kromosom baru yang mempunyai nilai lebih baik dari induknya. Ada dua jenis operator yang digunakan yaitu *crossover* (tukar silang) dan mutasi.

Operator tukar silang mengambil dua individu baru atau keturunan dari pertukaran gen kedua induk. Dari kedua *offspring* (anak), masing-masing mewarisi gen tertentu pada kedua induknya. Mutasi diterapkan pada tiap-tiap offspring secara individual setelah tukar silang. Mutasi memodifikasi kromosom dengan memindahkan gen-gen dari posisi semula.

2.2 Boltzman Machine

Definisi Boltzman Machine.
Boltzman Machine adalah salah satu metode *derivative-free optimization* yang dapat digunakan pada permasalahan optimasi [1]. Boltzman Machine banyak digunakan karena keefektifannya dalam menemukan penyelesaian yang optimal untuk permasalahan optimasi kombinatorik seperti *Travelling Salesman Problem* (TSP).

Prinsip yang melatar belakangi Boltzman Machine dapat dianalogikan dengan proses pendinginan logam pada temperatur yang terkontrol. Temperatur yang berangsur-angsur turun menyebabkan struktur atom dalam logam mengkristal dengan kepadatan tinggi serta memiliki energi yang relatif rendah, seringkali proses ini disebut *Simulated Annealling*. Tetapi jika terjadi sebaliknya, temperatur turun secara cepat, atom-atom dalam logam tidak memiliki cukup waktu untuk menyesuaikan strukturnya. Akibat yang ditimbulkan dari keadaan ini adalah hasilnya berupa material tanpa bentuk dengan kandungan energi yang tinggi.

Pada Boltzman Machine, fungsi objektif yang akan diminimasi dapat dianalogkan dengan energi pada sistem termodinamika. Dengan temperatur yang tinggi, Boltzman Machine mempunyai kecenderungan untuk menerima titik/solusi baru dengan energi yang lebih tinggi. Hal ini dapat disamakan dengan keadaan dimana suatu atom dengan mobilitas tinggi mencoba untuk menyesuaikan strukturnya dengan atom-atom lain yang berasal dari luar dan energi yang dapat secara tiba-tiba berubah dalam waktu yang relatif cepat.

Bagian yang paling penting dari Boltzman Machine adalah *machine schedule*/*cooling schedule*, yang menentukan kecepatan penurunan temperatur dari keadaan temperatur tinggi ke rendah. Yang perlu diperhatikan pada proses cooling schedule adalah jika penurunan temperatur terlalu lambat, waktu CPU akan terbuang sia-sia. Sebaliknya jika penurunan temperatur dilakukan terlalu cepat maka terdapat kemungkinan akan terjebak pada proses pencarian pada satu daerah saja. Proses Boltzman Machine digunakan untuk mencari energi global yang minimum. Penurunan temperatur dilakukan secara bertahap (*gradual*), hal ini dilakukan agar jaringan dapat mengatur susunannya pada konfigurasi yang tepat. Penurunan temperatur secara gradual merupakan cara terbaik untuk menghindari lokal minimal tanpa membutuhkan waktu tunggu yang lama.

Algoritma Bolzman Machine

```
SA_Alg(ncity,T0,α)
{
   x ← x0
   T ← T0
   do {
       frozen=1
       for (loop=1;loop<ncity;loop++)
       {
         x' ← mutasi(x);
         Δf ← f(x')-f(x);
         r ← bilangan acak 0 − 1;
         if(Δf<0 or r<exp(-Δf/T))
         {
           x ← x';
           frozen=0;
         }
         T ← T*α;
       }
   } while (not frozen)
   return x
}
```

3 Traveling Salesman Problem

3.1 Sejarah

Traveling Salesman Problem (TSP) adalah persoalan optimasi yang relatif lama, didokumentasikan pertama kali pada awal 1759 oleh Euler, yang mempunyai ketertarikan untuk menyelesaikan problem perjalanan pion perwira dalam permainan catur itu mendatangi hanya sekali dari masing-masing 64 kotak kecil yang ada dalam papan catur.

Istilah TSP pertama kali digunakan dalam sebuah buku yang dikeluarkan oleh negara Jerman *The Traveling Salesman How and What He Should Do to Get Commissions and Be succesfull in His Bussiness* pada tahun 1932, yang ditulis oleh seorang veteran traveling salesman.

Definisi. Definisi dari Traveling Salesman Problem (TSP) yaitu diberikan n buah kota dan d_{ij} yang merupakan jarak anatara kota c_i dan kota c_j, seorang sales ingin membuat suatu lintasan tertutup dengan mengunjungi setiap kota. Tujuannya adalah memilih lintasan tertutup yang total jaraknya minimum diantara pilihan dari semua kemungkinan lintasan.

Traveling Salesman Problem memiliki batasan masalah dalam dua kondisi, yaitu batasan lemah dan batasan kuat. Batasan yang lemah adalah batasan masalah yang jika tidak terpenuhi maka tidak membuat penyelesaian yang didapatkan menjadi salah. Meskipun begitu, sedapat mungkin batasan-batasan masalah tersebut harus dipenuhi. Contoh dari batasan lemah adalah lintasan dengan jarak tempuh minimum, karena kadang-kadang lintasan yang dapat ditemukan yang menghubungkan semua kota yang ada dengan setiap kota yang hanya dikunjungi sekali, bukan merupakan lintasan dengan jarak tempuh paling minimum. Sedangkan batasan masalah yang kuat adalah batasan masalah yang harus dipenuhi. Jika batasan masalah tersebut tidak dipenuhi maka akan membuat penyelesaian yang didapatkan menjadi salah. Misalnya adalah bahwa setiap kota yang terdapat dalam daftar rencana perjalanan harus dikunjungi tepat sekali, tidak akan melakukan kunjungan dua kali atau lebih dalam waktu yang bersamaan dan dalam perjalanan diawali dan diakhiri pada kota yang sama.

3.2 Penyelesaian dengan Algoritma Genetika

Pengkodean.
Pengkodean sebagai langkah awal dalam penggunaan algoritma genetika, dalam penyelesaian TSP ini juga harus dilakukan dahulu. Selama ini telah ditemukan beberapa skema representasi yang sesuai untuk permasalahan kombinatorial, diantaranya adalah representasi permutasi dan representasi kunci-kunci random.

Representasi Permutasi
Representasi permutasi merupakan representasi yang paling alami untuk lintasan dalam TSP, dimana kota-kota merupakan bagian yang harus dilalui oleh seorang traveling salesman. Wilayah pencarian untuk representasi ini adalah himpunan permutasi dari kota-kota berupa simbol abjad. Sebagai contoh dari 9 kota dalam TSP:

3 - 2 - 5 - 4 - 7 - 1 - 6 - 9 - 8

dari representasi di atas dapat diuraikan bahwa lintasan berangkat dari kota 3 menuju kota 2, 5, 4 dan seterusnya. Representasi ini sering juga disebut *representasi path* atau *representasi order*.

Representasi kunci-kunci random
Representasi ini menghasilkan sebuah genotype dengan bilangan acak antara 0 - 1. Nilai ini digunakan sebagai kunci-kunci untuk mengkodekan penyelesaian. Sebagai contoh, 1 kromosom untuk 9 kota bisa direpresentasikan [0.23 0.82 0.45 0.74 0.87 0.11 0.56 0.69 0.78] dimana posisi i dalam list menunjukkan kota i. Nilai acak dalam posisi i mementukan urutan didatanginya kota i dalam lintasan TSP. Dengan kunci-kunci random di atas, kita dapat menentukan bahwa nilai 0.11 nilai yang paling kecil sehingga kota ke-6 menempati urutan pertama, 0.23 nilai terkecil kedua sehingga kota ke-1 menempati urutan kedua dan seterusnya, sehingga dari kunci-kunci random di atas kita dapatkan lintasan

6 - 1 - 3 - 7 - 8 - 4 - 9 - 2 - 5

kunci-kunci random dapat memperkecil keturunan yang tidak mungkin dengan merepresentasikan dengan cara sederhana.

Populasi Awal
Populasi awal sangat berpengaruh terhadap kinerja algoritma genetika. Cara inisialisasi awal untuk menghasilkan seluruh kromosom

untuk permasalahan kombinatorial ini adalah pembangkitan bilangan secara acak.

Salah satu cara untuk menghasilkan populasi awal acak adalah dengan permutasi Josephus. Misal ada kota dari 1 - 9. Permutasi dari lintasan dapat dilakukan dengan menentukan titik awal dan selang. Misal titik awal adalah 6 dan selang 5, maka lintasan berangkat dari kota 6, selang 5 dari 6 adalah 2 (dengan asumsi kota 1 - 9 membentuk *circular list*). Kota 2 dihapus dari list, kemudian selang 5 adalah 7. Proses ini diulang hingga ada satu lintasan dalam list. Hasil dari permutasi ini adalah 2 - 7 - 3 - 8 - 4 - 9 - 5 - 1 - 6

Fungsi Evaluasi
Untuk mengevaluasi kromosom-kromosom, maka digunakan fungsi penilaian

$$\text{eval}(V_k) = \sum_{i=1}^{\text{city\_size}} \text{dist}(c_i - c_{i+1}) + \text{dist}(c_{\text{city\_size}} - c_1)$$

dimana: city_size = banyaknya kota dalam lintasan dan $\text{dist}(c_i - c_{i+1})$ = jarak antara kota c_i dengan kota c_{i+1} yang ada dalam lintasan.

Seleksi
Seleksi yang digunakan adalah sampling stokastik dengan wilayah yang tetap. Sampling stokastik diimplementasikan dengan piringan Roulette. Berikut ini langkah-langkah proses seleksi.

Prosedur pra-seleksi yaitu dengan piringan Roulette:

i. Hitung nilai fitness bagi tiap kromosom

$$\text{eval}(V_i) = f(x)$$

$i = 1, 2, 3, \ldots, \text{pop\_size}$

ii. Hitung total nilai fitness

$$F = \sum_{i=1}^{\text{pop\_size}} \text{eval}(V_i)$$

iii. Hitung probabilitas seleksi P_i untuk tiap kromosom

$$Pi = \frac{\text{eval}(V_i)}{F}$$

$i = 1, 2, 3, \ldots, \text{pop\_size}$

iv. Hitung probabilitas komulatif q_i untuk tiap kromosom

$$q_i = \sum_{j=1}^{i} P_j$$

$i = 1, 2, 3, \ldots, \text{pop\_size}$

Prosedur Seleksi:

i. Bangkitkan bilangan acak r antara 0 dan 1.

ii. Jika $r \leq q_1$ maka pilih kromosom pertama (V_1). Jika tidak, maka pilih kromosom V_i ($2 \leq i \leq \text{pop\_size}$) sedemikian rupa sehingga $q_{i-1} < r \leq q_i$.

Operator - operator Algoritma
Dua jenis operator yang biasa digunakan adalah tukar silang dan mutasi.

a. **Tukar Silang.** Tukar silang merupakan teknik dimana sepasang kromosom diproduksi dengan mengkopi komplemen induk pada keturunan. Beberapa metode operator tukar silang diciptakan untuk representasi permutasi seperti *partial mapped crossover* (PMX), *order crossover* (OX), *cycle crossover* (CX), *position based crossover*, *order based crossover*, *heuristic crossover*, dan lain-lain.

b. **Mutasi.** Mutasi digunakan untuk memodifikasi suatu kromosom agar menghasilkan perubahan kromosom yang random. Beberapa operator mutasi telah diciptakan untuk representasi permutasi seperti metode inversion, insertion, displacement.

Inversion mutation, yaitu memilih dua posisi dari kromosom kemudian menukar posisi dari substring.

Insertion mutation, yaitu memilih kota secara random dan disisipkan pada posisi yang acak pada suatu kromosom.

Displacement mutation, yaitu memilih sebuah kota secara acak dan menukar pada posisi yang acak pula.

3.3 Penyelesaian dengan Boltzman Machine

Dari algoritma Boltzman Machine yang telah dijelaskan sebelumnya dapat dilakukan langkah-langkah untuk menyelesaikan TSP.

Boltzman Machine

i. Inisialisasi/bangkitkan solusi awal (X_0). Solusi pertama (2 3 5 4 1) ditempatkan secara acak, menentukan T_0 = maxint = 217483647, mementukan cooling rate α = 0.95

ii. Evaluasi fungsi objektif (fitness). Cara kerjanya adalah membaca pada matrik yang berisi data jarak untuk tiap simpul
SQRT(SQR($X_2 - X_1$) + SQR($Y_2 - Y_1$))
Misal solusi pertama yang terbentuk:
Fitness(2,3) = 375,76
Fitness(3,5) = 132,77
Fitness(5,4) = 134,73
Fitness(4,1) = 171,77
Fitness(1,2) = 438,52
Σ Fitness solusi pertama=815,03
selama system belum frozen maka akan dilakukan proses mutasi sampai batas mutasi yang diperbolehkan.

iii. Operasi mutasi. Jumlah mutasi=5 atau sesuai dengan inputan, mutasi akan dilakukan terus (diulang) sampai system frozen (=tidak dapat menemukan solusi baru yang memiliki energi yang lebih rendah) atau telah menemukan solusi baru dengan fungsi fitness lebih kecil dari solusi yang lama (Xnew<Xold). Jika belum memenuhi kondisi tersebut maka mutasi diulang sebanyak 5 kali.

   a. Mutasi pertama masih tetap. ($X_1 \leq X_0$) (2 3 4 5 1) dengan fitness=815,03
   b. Mutasi kedua Dihasilkan X_1= (2 5 4 3 1) dengan fitness=827,2. Karena ($X_1 > X_0$) maka mutasi diulang
   c. Mutasi ketiga X_1=(1 4 5 3 2) dengan fitness = 826,4. Karena $X_1 > X_0$ maka mutasi diulang
   d. Mutasi keempat X_1=(1 4 5 3 2) dengan fitness =826,4. Karena $X_1 > X_0$ maka mutasi diulang
   e. Mutasi kelima X_1=(2 3 5 1 4) dengan fitness=813,05 Karena ($X_1 < X_0$) maka mutasi dihentikan.

iv. Keluar dari loop mutasi $\Delta F = X_0 - X_1$= misal diperoleh $-11,37$

v. Jika ($\Delta F < 0$) atau random $(0,1) < \exp(-\Delta F/T)$ maka:
   a. Hitung nilai fitness
   X_0·Fitness=X_1·Fitness
   b. Memperbarui solusi terakhir:
   solusi (fitness) = 813,05
   c. Frozen=0

vi. Jika system sudah frozen maka dilakukan penurunan temperatur

vii. Solusi (lintasan): 2–3–5–1–4

4 Pembahasan Hasil

Dalam analisis ini akan dibandingkan efisiensi dari kedua metode yang digunakan untuk menyelesaikan traveling salesman problem ini, maka dilakukan analisis dengan membandingkan faktor-faktor sebagai berikut:

4.1 Algoritma Genetika

Dalam pembahasan ini yang akan diukur adalah banyaknya operasi utama. Dengan menghitung banyaknya operasi utama akan dibahas tentang banyak kerja rata-rata (*average case* atau *average complexity*) dan banyak kerja terburuk (*worst case* atau *complexityi*).

Himpunan macam-macam input yang berukuran n dinamakan domain input dan disimbolkan D_n. Misalkan i adalah elemen D_n, probabilitas bahwa input i akan dipakai adalah $P(i)$ dan banyaknya operasi yang dilakukan oleh algoritma bila diberi input i adalah $t(i)$.

a. Banyak kerja rata-rata (*average case*).
Banyaknya kerja yang dilakukan untuk macam inputannya yang menyebabkan perbedaan banyaknya kerja $t(i)$ adalah sejumlah kombinasi dari banyaknya generasi (iterasi maksimum), ukuran populasi, probabilitas mutasi dan probabilitas tukar silang. Pada operasi perkalian ini yang akan dihitung operasi dasar untuk perkalian, dimana $t(i)$ yang dimaksud dalam rumus berikut adalah banyaknya operasi dasar perkalian dan $p(i)$ adalah probabilitas untuk input(i).

$$\begin{aligned} A(g,m,n,j,p) &= \sum_{i \in D} p(i)\, t(i) \\ &= 1 * (g(17m + jmn + 2jn + pm + 3)) \\ &= 17gm + gjmn + 2gjn + gpm + 3g \end{aligned}$$

dimana
g = parameter generasi

m = parameter ukuran populasi
n = parameter jumlah kota
p = parameter probabilitas tukar silang
j = parameter probabilitas mutasi

b. Banyaknya kerja terburuk (worst case). Banyaknya kerja terburuk dapat dihitung dengan menentukan kondisi yang menyebabkan kerja dasar terbesar.

$$W(g, n, m, p, j) = \max_{i \in D}(i) = 17gm$$

c. Banyaknya memori yang dipakai.
Algoritma penyelesaian TSP yang merupakan modifikasi algoritma dasar metode Boltzman Machine , memerlukan tempat penyimpanan di dalam memori, yaitu tempat untuk menyimpan instruksi, konstanta dan variabel yang digunakan oleh program tersebut. Besarnya memori utnuk menyimpan variabel ekstranya ini, yang biasa disebut dengan extra space, besarnya tergantung dari besarnya input. Karena besarnya ekstra space bergantung pada besarnya input, maka algoritma ini dikatakan tidak kerja ditempat (*no works in place*)

d. Hasil running time dari algoritma genetika
Uji coba dilakukan dengan cara memasukkan beberapa n kota yang berbeda, selanjutnya dicari jarak, lintasan, penemuan solusi baru dan waktu yang dibutuhkan untuk menyelesaikan traveling salesman problem.

Gambar 1: Hasil Uji-coba dengan Algoritma Genetika

4.2 Boltzman Machine

Dalam pembahasan ini yang akan diukur adalah banyaknya operasi utama. Dengan menghitung banyaknya operasi utama akan dibahas tentang banyak kerja rata-rata (*average case* atau *average complexity*) dan banyak kerja terburuk (*worst case* atau *complexity*). Himpunan macam-macam input yang berukuran n dinamakan domain input dan disimbolkan D_n. Probabilitas bahwa input i akan dipakai adalah $P(i)$ dan banyaknya operasi yang dilakukan oleh algoritma bila diberi input i adalah $t(i)$.

a. Banyak kerja rata-rata (*average case*).
Banyaknya kerja yang dilakukan untuk macam inputannya yang menyebabkan perbedaan banyaknya kerja $t(I)$ adalah sejumlah kombinasi dari banyaknya iterasi maksimum ($T_{\max}$) yang diperbolehkan, suhu awal (T_0) dan cooling rate α adalah sebagai berikut: Pada operasi eksponensial ini yang akan dihitung operasi dasar untuk eksponensial, dimana $t(I)$ yang dimaksud dalam rumus berikut adalah banyaknya operasi dasar eksponensial dan $P(I)$ adalah probabilitas untuk input I

$$\begin{aligned} A(n) &= \sum_{I \in D} P(I)\, t(I) \\ &= \frac{1}{2} * (\text{iterasi\_max} + 1) \end{aligned}$$

b. Banyaknya kerja terburuk (*worst case*).
Banyaknya kerja terburuk dapat dihitung dengan menentukan kondisi yang menyebabkan kerja dasar terbesar, untuk permasalahan ini kondisi tersebut tidak pernah ditemukan karena macam inputnya hanya satu, maka untuk semua kondisi mempunyai banyak kerja dasar yang sama. Dan karena dari ketiga operasi dasar di atas operasi eksponensial yang paling tinggi nilainya, maka banyaknya kerja terburuk sama dengan kerja rata-rata dari operasi eksponensial, yaitu:

$$\begin{aligned} W(n) &= \max_{I \in D} t(I) \\ &= 2 * (\text{iterasi\_max}) \end{aligned}$$

c. Banyaknya memori yang dipakai.
Algoritma penyelesaian TSP yang merupakan modifikasi algoritma dasar metode Boltzman Machine, memerlukan tempat penyimpanan di dalam memori, yaitu tempat untuk menyimpan instruksi, konstanta dan variabel yang digunakan oleh program

tersebut. Besarnya memori utnuk menyimpan variabel ekstranya ini, yang biasa disebut dengan *extra space*, dan besarnya tergantung dari besarnya input. Karena besarnya ekstra space bergantung pada besarnya input, maka algoritma ini dikatakan tidak kerja di tempat (*no works in place*)

d. Hasil uji coba dengan metode Boltzman Machine.
Uji coba dilakukan dengan cara memasukkan n kota yang berbeda, selanjutnya dicari jarak, lintasan, penemuan solusi baru dan waktu yang dibutuhkan untuk menyelesaikan traveling salesman problem.

Gambar 2: Hasil Uji-coba dengan Botzman Machine

5 Kesimpulan

Dari algoritma dan penyelesaian traveling salesman problem dengan metode Algoritma genetika dan metode Boltzman Machine di atas, dapat diambil beberapa kesimpulan, yaitu:

a. Algoritma Genetika
Setelah dilakukan uji coba penyelesaian traveling salesman problem dengan algoritma genetika, metode dengan menggunakan algoritma genetika sebenarnya menunjukkan bahwa algoritma ini dapat.

Ukuran populasi (pop_size) mempengaruhi unjuk kerja yang baik dan keefektifan algoritma genetika. Algoritma genetika dengan populasi yang kecil biasanya unjuk kerjanya buruk karena populasi tidak menyediakan cukup materi untuk mencakup ruang persoalan. Populasi yang lebih besar dibutuhkan untuk merepresentasikan keseluruhan ruang persoalan. Di samping itu input pada parameter probabilitas mutasi dan probabilitas tukar silang juga mempengaruhi unjuk kerja sistem.

b. Boltzman Machine
Dengan percobaan penyelesaian traveling salesman problem yang dibahas, metode Boltzman Machine menunjukkan bahwa algoritma ini dapat menemukan penyelesaian yang lebih baik daripada Algoritma Genetika dari segi efisiensi waktu. Kejadian untuk permasalahan 100 kota, waktu (duration) yang diambil sangat singkat.

Batasan titik beku (frozen), dimana semakin banyak kesempatan yang diberikan pada Boltzman Machine sebelum dinyatakan frozen, maka akan memberikan keleluasaan bagi Boltzman Machine untuk menelusuri penyelesaian yang berikutnya. Hanya saja, hal ini membutuhkan waktu yang lama. Sebaliknya bila faktor frozen diberikan terlalu sedikit maka operasi pencarian terlalu cepat dinyatakan frozen, sehingga kesempatannya menjadi hilang.

Daftar Pustaka

[1] Faucet, L, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice Hall, 1994.

[2] Michalewicz, Z, *Genetics Algoritms + Data Structures = Evolution Program*, Third, Revised, and Extended Edition, Springer-Verlag, 1992.

[3] Mitsuo Gen dan Runwei Cheng, *Genetic Algoritms and Engineering Design*, John Wiley & Sons, Inc., 1997.

[4] Marek Obitko, *Genetic Algoritms*, http:cs.felk.cvut.cz/˜obitko/ga/main.html.

[5] Shaffer, C.A., *A Partical Introduction to Data Structures and Algoritms Analysis*, Prentice Hall International, Inc., 1997.