BAB II

LANDASAN TEORI

2.1 Ant Colony System dan Asal Usulnya

Pada subbab ini akan diuraikan mengenai asal usul *Ant Colony System* (ACS) , yaitu membahas tentang semut dan tingkah lakunya yang merupakan sumber inspirasi lahirnya *ant system*, membahas tentang *ant system* yang merupakan transformasi perilaku semut ke dalam sebuah bentuk metodologi, membahas ACS yang telah dikembangkan berdasar pada *ant system*.

2.1.1 Semut dan Perilakunya

Pada kehidupan sebenarnya, semut – semut meninggalkan sarang untuk mencari makanan dan harus mencari kembali sarang mereka. Misalkan ada segerombolan semut yang mencari makanan, maka semut yang berada di depan harus memilih lintasan tertentu untuk dilewati. Pada saat semut pertama berjalan, semut tersebut meninggalkan *pheromone* yang dapat dicium oleh semut berikutnya, sehingga semut – semut berikutnya tahu apakah tempat tersebut sudah dilewati atau belum.

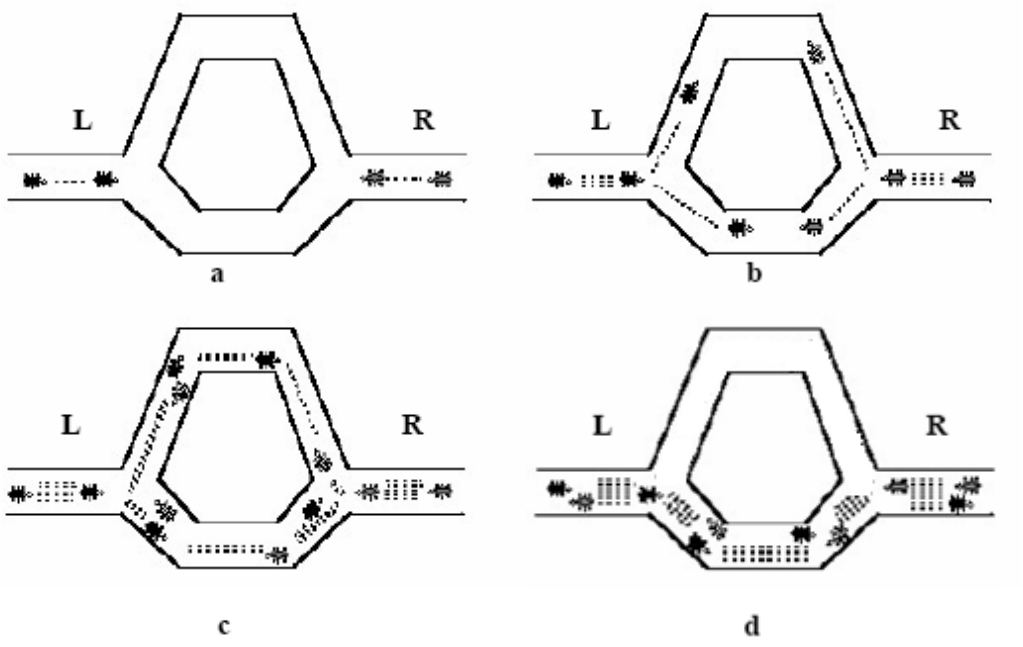
Algoritma semut diperkenalkan oleh Moyson dan Mendrik dan secara meluas dikembangkan oleh Macro Dorigo, merupakan teknik probalilistik untuk menyelesaikan masalah komputasi dengan menemukan jalur terbaik melalui grafik. Algoritma ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan.

Algoritma Semut (Antco) diadopsi dari perilaku koloni semut yang dikenal sebagai system semut. Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka semakin jelas bekas jejak kakinya. Hal

ini menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama semakin berkurang kepadatan semut yang melewatinya, atau bahkan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam jumlah banyak, semakin lama semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut melalui lintasan tersebut.

Pada saat semut berjalan ia meninggalkan sejumlah informasi, yang disebut *pheromone* (dalam jumlah tertentu), ditempat yang dilaluinya dan menandai jalur tersebut. Dengan perantara *pheromone* inilah terjadi komunikasi tidak langsung dan juga pertukaran informasi antar semut selagi membangun suatu solusi. Bentuk komunikasi tidak langsung yang diperlihakan oleh semut ini disebut *stigmergy*.

Gambar dibawah mengilustrasikan proses dari *stigmergy*. Semut menggunakan *pheromone* untuk menemukan jalur terpendek antara dua ujung yang dihubungkan dengan dua cabang: bawah (yang lebih pendek) dan atas (yang lebih panjang).



Gambar 2.1 Perjalanan semut menemukan sumber makanan

Gambar 2.1.a di atas menunjukkan ada dua kelompok semut yang akan melakukan perjalanan. Satu kelompok bernama L yaitu kelompok yang berangkat dari arah kiri yang merupakan sarang semut dan kelompok lain yang bernama kelompok R yang berangkat dari kanan yang merupakan sumber makanan. Kedua kelompok semut dari titik berangkat sedang dalam posisi pengambilan keputusan jalan sebelah mana yang akan diambil. Kelompok semut L membagi dua kelompok lagi. Sebagian melalui jalan atas dan sebagian melalui jalan bawah. Hal ini juga berlaku pada kelompok semut R. Gambar 2.1.b dan gambar 2.1.c menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan *pheromone* atau jejak kaki di jalan yang telah dilalui. *Pheromone* yang ditinggalkan oleh kumpulan semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit dari pada jalan yang di bawah. Hal ini dikarenakan jarak yang ditempuh lebih panjang daripada jalan bawah. Sedangkan *pheromone* yang berada di jalan bawah, penguapannya cenderung lebih lama. Karena semut yang melalui jalan bawah lebih banyak daripada semut yang melalui jalan atas. Gambar 2.1.d menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena *pheromone* yang ditinggalkan masih banyak. Sedangkan *pheromone* pada jalan atas sudah banyak menguap sehingga semut-semut tidak memilih jalan atas tersebut. Semakin banyak semut yang melalui jalan bawah maka semakin banyak semut yang mengikutinya. Demikian juga dengan jalan atas, semakin sedikit semut yang melalui jalan atas, maka *pheromone* yang ditinggalkan semakin berkurang bahkan hilang. Dari sinilah kemudian terpilihlah jalur terpendek antara sarang dan sumber makanan. Hal ini berarti bahwa semakin banyak semut yang mengikuti sebuah jalur maka semakin bertambah menariklah jalur tersebut untuk dilalui. Probabilitas dimana seekor semut memutuskan untuk mengikuti suatu jalur meningkat dengan banyaknya semut yang lebih dulu menggunakan jalur tersebut.

2.1.2 Ant System (AS)

Perilaku koloni semut telah menginspirasikan munculnya sebuah metodologi baru yang di dalamnya terdapat sekumpulan semut buatan, yang dinamai dengan *ants*, yang saling bekerja sama dalam mencari solusi terhadap suatu masalah optimisasi kombinatorial dengan cara bertukar informasi melalui *pheromone* yang diletakkan pada ruas-ruas sebuah graf. Semut – semut buatan (untuk selanjutnya ditulis sebagai *ants*) yang digunakan dalam sistem ini mempunyai perbedaan besar dengan hewan semut yang asli, antara lain *ants* akan memiliki memori, mereka tidak sepenuhnya buta, dan mereka akan berada pada lingkungan dimana waktunya adalah diskrit.

Secara informal, AS bekerja sebagai berikut: Setiap semut memulai tournya melalui sebuah titik yang dipilih secara acak (setiap semut memiliki titik awal yang berbeda). Secara berulang kali, satupersatu titik yang ada dikunjungi oleh semut dengan tujuan untuk menghasilkan sebuah tour. Pemilihan titik-titik yang akan dilaluinya didasarkan pada suatu fungsi probabilitas, dinamai aturan transisi status (*state transition rule*), dengan mempertimbangkan *visibility* (invers dari jarak) titik tersebut dan jumlah *Pheromone* yang terdapat pada ruas yang menghubungkan titik tersebut. Semut lebih suka untuk bergerak menuju ke titik – titik yang dihubungkan dengan ruas yang pendek dan memiliki tingkat *pheromone* yang tinggi. Setiap semut memiliki sebuah memori, dinamai *tabulist*, yang berisi semua titik yang telah dikunjunginya pada setiap tour. *Tabulist* ini mencegah semut untuk mengunjungi titik-titik yang sebelumnya telah dikunjungi selama tour tersebut berlangsung, yang membuat solusinya mendekati optimal.

Setelah semua semut menyelesaikan tour mereka dan *tabulist* mereka menjadi penuh, sebuah aturan pembaruan *pheromone* global (*global Pheromone updating rule*) diterapkan pada setiap semut. Penguapan *pheromone* pada semua ruas dilakukan, kemudian setiap semut menghitung panjang tour yang telah mereka lakukan lalu meninggalkan sejumlah *pheromone* pada edge-edge yang merupakan bagian dari tour mereka yang sebanding dengan kualitas dari solusi yang mereka hasilkan. Semakin pendek sebuah tour yang dihasilkan oleh setiap

semut, jumlah *pheromone* yang ditinggalkan pada edge – edge yang dilaluinya pun semakin besar. Dengan kata lain, edge – edge yang merupakan bagian dari tour – tour terpendek adalah edge – edge yang menerima jumlah *pheromone* yang lebih besar. Hal ini menyebabkan edge – edge yang diberi *pheromone* lebih banyak akan lebih diminati pada tour-tour selanjutnya, dan sebaliknya edge-edge yang tidak diberi *pheromone* menjadi kurang diminati. Dan juga, rute terpendek yang ditemukan oleh semut disimpan dan semua *tabu list* yang ada dikosongkan kembali.

Peranan utama dari penguapan *pheromone* adalah untuk mencegah stagnasi, yaitu situasi dimana semua semut berakhir dengan melakukan tour yang sama. Proses di atas kemudian diulangi sampai tour-tour yang dilakukan mencapai jumlah maksimum atau sistem ini menghasilkan perilaku stagnasi dimana sistem ini berhenti untuk mencari solusi alternatif.

Seiring waktu, bagaimanapun juga jejak *pheromone* akan menguap dan akan mengurangi kekuatan daya tariknya. Lebih cepat setiap semut pulang pergi melalui jalur tersebut, maka *pheromone* yang menguap lebih sedikit. Begitu pula sebaliknya jika semut lebih lama pulang pergi melalui jalur tersebut, maka *pheromone* yang menguap lebih banyak

2.1.2.1 Aturan Transisi Status

Aturan transisi status ini diambil jika semut harus mengeksplorasi simpul yang belum pernah dikunjungi sama sekali. Kemungkinan ini merupakan probabilitas dari semut *m* pada titik i yang memilih untuk menuju ke titik j.[6]

$$P_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\Sigma[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & Jika\ j \in\ diijinkan \\ j \epsilon\ diijinkan & \\ 0 & jika\ j \notin\ diijinkan \end{cases} \quad \text{..... (2.1)}$$

Dimana τ_{ij} adalah jumlah *pheromone* yang terdapat pada edge antara titik *i* dan titik *j*, $\eta_{ij} = 1/d_{ij}$ adalah *visibility* (invers dari jarak d_{ij}) dimana $d_{ij} : \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$

(jika hanya diketahui koordinat titiknya saja). α adalah sebuah parameter yang mengontrol bobot (*weight*) relatif dari *pheromone* dan β adalah parameter pengendali jarak ($\alpha \geq 0$ dan $\beta > 0$).

pada persamaan (2.1) kita mengalikan *pheromone* pada edge (i,j) dengan nilai *visibility* yang sesuai (η_{ij}). Dengan cara ini kita memilih edge yang lebih pendek dan memiliki jumlah *pheromone* yang lebih besar.

2.1.2.2 Update Pheromone Trail

Setelah semua semut menyelesaikan tour-nya masing – masing maka *pheromone* di-*update*. Dalam *Ant System*, jumlah *pheromone* (τ_{ij}) akan berubah sesuai dengan persamaan 2.2 dibawah ini :

$$\tau_{ij} \leftarrow (1-\alpha) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \text{.....(2.2)}$$

dengan $\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L^k} & jika\ (i,j) \epsilon\ tour\ yang\ dilakukan\ oleh\ semut\ k \\ 0 & sebaliknya \end{cases}$ ..... (2.3)

Dimana L^k, panjang tour yang dilalui oleh semut k. $0 \leq \alpha \leq 1$ adalah sebuah parameter tingkat *evaporasi pheromone*, dan *m* adalah jumlah dari semut.

Untuk memastikan bahwa semut mengunjungi n titik yang berbeda, diberikan *tabu list* pada masing – masing semut, yaitu sebuah struktur data yang menyimpan titik – titik yang telah dikunjungi semut dan melarang semut mengunjungi kembali titik – titik tersebut sebelum mereka menyelesaikan sebuah tour. Ketika sebuah tour selesai, *tabulist* digunakan untuk menghitung solusi yang ditemukan semut pada tour tersebut. *Tabulist* kemudian dikosongkan dan semut kembali bebas memilih titik tujuannya pada tour berikutnya. $Tabu_k$ adalah *tabu list* untuk semut k. $Tabu_k$ (i) adalah elemen ke-i dari $Tabu_k$, yaitu titik ke-i yang dikunjungi semut k pada suatu tour.

Meskipun AS bermanfaat dalam menemukan solusi-solusi yang optimal ataupun bagus untuk TSP dengan jumlah kota yang sedikit (sampai dengan 30 kota), waktu yang dibutuhkan untuk mendapatkan hasil tersebut membuatnya menjadi tidak mungkin lagi untuk diterapkan pada masalah yang lebih besar.

2.1.3 Ant Colony System (ACS)

Algoritma *Ant Colony System* (ACS) merupakan pengembangan dari AS selanjutnya dengan maksud untuk meningkatkan efisiensinya ketika diterapkan pada TSP yang lebih kompleks. ACS memiliki perbedaan dengan *ant system* karena tiga aspek utama

- Aturan transisi status pada sistem ini memberikan suatu cara langsung untuk menyeimbangkan antara penjelajah (*exploration*) ruas – ruas yang baru dengan eksploitasi (*exploitation*) dari sebuah *priori* dan pengetahuan yang dihimpun mengenai masalah tersebut.
- Aturan pembaruan *pheromone* global hanya dilakukan pada ruas – ruas yang merupakan bagian dari tour terbaik.
- Disaat *ants* membangun sebuah solusi, diterapkan suatu aturan pembaruan *pheromone* lokal (*local pheromone updating rule*).

Secara informal, ACS bekerja sebagai berikut: pertama kali, sejumlah m *ants* ditempatkan pada sejumlah n kota berdasarkan beberapa aturan inisialisasi (misalnya, secara acak). Setiap semut membuat sebuah tour (yaitu, sebuah solusi TSP yang mungkin) dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun *tour*nya, seekor semut juga memodifikasi jumlah *pheromone* pada ruas-ruas yang dikunjunginya dengan menerapkan aturan pembaruan *pheromone* lokal yang telah disebutkan tadi. Setelah semua *ants* mengakhiri *tour* mereka, jumlah *pheromone* yang ada pada ruas – ruas dimodifikasi kembali (dengan menerapkan aturan pembaruan *pheromone* global). Seperti yang terjadi pada *ant system*, dalam membuat tour, *ants* ‘dipandu’ oleh informasi heuristik (mereka lebih memilih ruas – ruas yang pendek) dan oleh informasi *pheromone*. Sebuah ruas dengan jumlah *pheromone* yang tinggi

merupakan pilihan yang sangat diinginkan. Sebuah ruas dengan jumlah *pheromone* yang tinggi merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan *pheromone* itu dirancang agar semut cenderung untuk memberi lebih banyak *pheromone* pada ruas – ruas yang harus mereka lewati. Berikutnya akan dibahas mengenai tiga karakteristik utama dari ACS, yaitu aturan transisi status, aturan pembaharuan *pheromone* global, dan aturan pembaharuan *pheromone* lokal.

2.1.3.1 Aturan Transisi Status

Aturan transisi status yang berlaku pada ACS ditunjukkan pada persamaan 2.4. Semut m yang berada di titik *i*, akan memilih titik berikutnya *j*, menurut persamaan berikut :[6]

$$J = \begin{cases} \arg\max_{u \in j_i^m} \left\{ \tau_{ij} [\eta_{ij}]^{\beta} \right\}, & jika\ q \leq q_0 (exploitasi) \\ s, & jika\ tidak\ (explorasi) \end{cases} \quad \text{.....(2.4)}$$

Dimana q adalah bilangan random dalam [0...1], q_0 ($0 \leq q_0 \leq 1$) adalah sebuah parameter pembanding bilangan random, dan s = P_{ij}^m probabilitas dari semut *m* pada titik *i* yang memilih untuk menuju ke titik *j* (persamaan 2.1).

Aturan transisi status yang dihasilkan dari persamaan (2.4) dan (2.1) dinamakan aturan *random-proportional* semu (*pseudorandom-proportional rule*). Aturan transisi status ini, sebagaimana dengan aturan *random-proportional* terdahulu, mengarahkan *ants* untuk bertransisi ke kota – kota yang dihubungkan dengan ruas-ruas yang pendek dan memiliki jumlah *pheromone* yang besar. Setiap kali seekor semut yang ada pada kota *i* harus memilih kota *j* sebagai tujuan berikutnya, ia membangkitkan sebuah bilangan acak antara 0 s.d. 1 ($0 \leq q_o \leq 1$). Jika $q \leq q_o$ maka semut tersebut akan memanfaatkan pengetahuan yang ada mengenai masalah tersebut, yaitu pengetahuan heuristik tentang jarak antara kota tersebut dengan kota – kota lainnya dan juga pengetahuan yang telah didapat dan disimpan dalam bentuk *pheromone trail*. Hal ini mengakibatkan ruas terbaik

(berdasarkan persamaan (2.4)) dipilih (*exploitation*). Jika sebaliknya maka sebuah ruas dipilih berdasarkan persamaan (2.1) (*biased exploration*).

2.1.3.2 Aturan Pembaruan *Pheromone* Global

Pada sistem ini, pembaruan *pheromone* secara global hanya dilakukan oleh semut yang membuat tour terpendek sejak permulaan percobaan. Pada akhir sebuah iterasi, setelah semua *ants* menyelesaikan tour mereka, sejumlah *pheromone* ditaruh pada ruas – ruas yang dilewati oleh seekor semut yang telah menemukan tour terbaik (ruas-ruas yang lain tidak diubah). Tingkat *pheromone* itu diperbarui dengan menerapkan aturan pembaruan *pheromone* global yang ditunjukkan oleh persamaan 2.6.[6]

$$\tau(i,j) \longleftarrow (1-\alpha)\cdot\tau(i,j) + \alpha\cdot\Delta\tau(i,j) \quad \text{..... (2.5)}$$

Dimana

$$\Delta\tau(i,j) = \begin{cases} (L_{gb}), & jika\ (i,j) \in global\ best\ tour \\ \square & \\ 0 \quad , & sebaliknya \end{cases} \quad \text{..... (2.6)}$$

$0 \leq \square \leq 1$ adalah peristiwa penguapan *pheromone*, dan L_{gb} adalah panjang dari tour terbaik secara global sejak permulaan percobaan. Seperti yang terjadi pada *ant system*, pembaruan *pheromone* global dimaksudkan untuk memberikan *pheromone* yang lebih banyak pada tour yang lebih pendek. Persamaan *update* jejak *pheromone* ini, dilakukan pada akhir sebuah iterasi algoritma, saat semua semut telah menyelesaikan sebuah tour. Persamaan diaplikasikan ke ruas yang digunakan semut menemukan lintasan keseluruhan yang terbaik sejak awal percobaan. Tujuan pemberian nilai ini adalah memberi sejumlah jejak *pheromone* pada lintasan terpendek, dimana tour terbaik (lintasan dengan panjang terpendek) mendapat penguatan. Bersama dengan *pseudo-randomproportional rule* dimaksudkan untuk membuat pencarian lebih terarah.

2.1.3.3 Aturan Pembaruan *Pheromone* Lokal

Selagi melakukan tour untuk mencari solusi dari TSP, *ants* mengunjungi ruas-ruas dan mengubah tingkat *pheromone* pada ruas-ruas tersebut dengan menerapkan aturan pembaruan *pheromone* lokal yang ditunjukkan oleh persamaan 2.7.[6]

$$\tau(i,j) \leftarrow (1 - \rho) \cdot \tau(i,j) + \rho \cdot \Delta\tau(i,j) \quad \text{..... (2.7)}$$

dimana $0 < \rho < 1$ adalah sebuah parameter.

Peranan dari aturan pembaruan *pheromone* lokal ini adalah untuk mengacak arah tour yang sedang dibangun, sehingga kota – kota yang telah dilewati sebelumnya oleh tour seekor semut mungkin akan dilewati kemudian oleh tour *ants* yang lain. Dengan kata lain, pengaruh dari pembaruan lokal ini adalah untuk membuat tingkat ketertarikan ruas – ruas yang ada berubah secara dinamis: setiap kali seekor semut menggunakan sebuah ruas maka ruas ini dengan segera akan berkurang tingkat ketertarikannya (karena ruas tersebut kehilangan sejumlah *pheromone*-nya), secara tidak langsung *ants* yang lain akan memilih ruas – ruas lain yang belum dikunjungi. Konsekuensinya, *ants* tidak akan memiliki kecenderungan untuk berkumpul pada jalur yang sama.

2.2 Algoritma Genetika

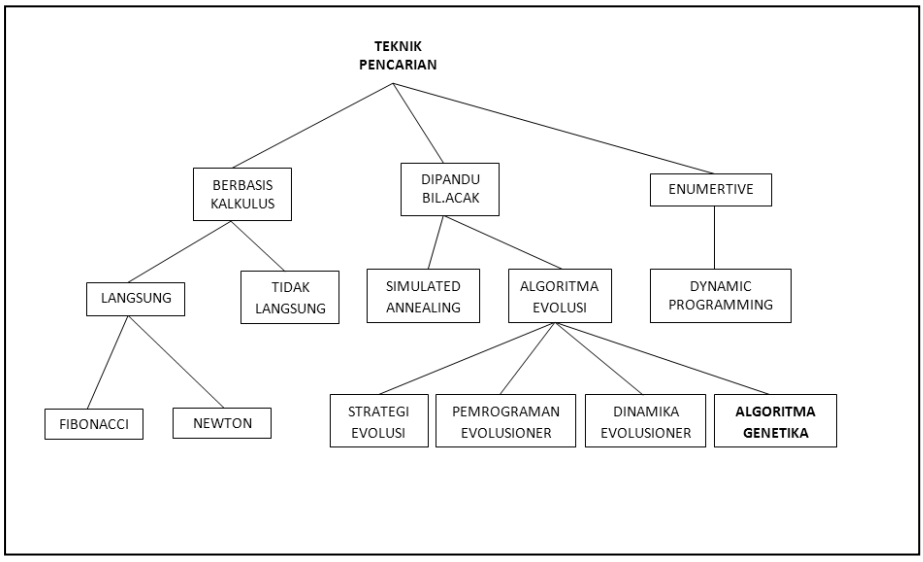
Algoritma Genetika (AG) adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis, sehingga banyak istilah dalam konsep biologi yang digunakan dalam AG. Semula ini merupakan hasil samping dari suatu kerjasama penelitian antara ahli biologi dengan ahli komputer. Ahli biologi yang ingin menyimulasikan proses evolusi, yang kemudian mengajak ahli komputer untuk merealisasikannya. Ternyata kemudian disadari proses evolusi merupakan proses ciptaan Allah SWT untuk mengoptimasi suatu persoalan.

Kenyataan ini dapat diamati dengan mudah, misalnya para ahli pertanian mampu menghasilkan buah-buahan baru yang merupakan hasil perkawinan antara buah yang besar tetapi pahit dengan buah yang kecil tetapi manis. Hasilnya, bisa

jadi adalah buah besar dan manis. Namun bisa saja karena 'kecelakaan' - yang kemudian diistilahkan dengan mutasi -, hasilnya adalah buah kecil dan pahit. Itu adalah proses yang kelihatanya acak dan tidak terpola tetapi sering kali menghasilkan sesuatu yang lebih baik dari indukya.

AG diusulkan pertama kali oleh John Holland dan kolega-koleganya di Universitas Michigan untuk aplikasi *cellular automata* pada tahun 1960-an. John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. AG adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom. Teknik ini menjadi populer di kalangan ilmuwan dan rekayasawan seluruh dunia untuk memecahkan masalah optimasi mereka. Aplikasinya antara lain meliputi *job shop scheduling*, pembelajaran pengendali *neuro-fuzzy*, pemrosesan citra dan optimasi kombinatorial.

AG adalah teknik optimasi yang terkenal. Secara umum ada tiga golongan besar teknik optimasi, yang berbasis pada kalkulus, enumeratif dan pencarian acak terarah (*guide random search*). Seperti dapat dilihat pada gambar 2.2 berikut ini.



Gambar 2.2 diagram pengelompokan dalam teknik pencarian

Setelah John Holland memaparkan idenya tentang AG maka banyak peneliti yang mengusulkan beberapa variasi dari algoritma dasar AG. Tetapi ciri penting

dari seluruh AG adalah teknik penanganan populasinya. Ide awal AG mengadopsi kebijakan penggantian umum dimana keseluruhan populasi diganti pada setiap generasi. Di pihak lain, kebijakan keadaan tunak (*steady-state policy*) menerapkan penggantian selektif bagi populasi. Hal ini lebih alami karena di alam sudah biasa orang tua dan anak hidup berdampingan pada saat yang bersamaan. Kebijakaan ini banyak diaplikasikan pada banyak variasi AG.

Penyajian individu dari AG biasanya menggunakan sekumpulan bilangan biner. Tetapi penyajian bilangan biner ini memiliki beberapa kelemahan bila diaplikasikan pada masalah multidimensional dan persoalan numerik persisi. Karena itu, D.E Golberg mengusulkan pemakain bilangan nyata atau floating-point. Z. Mischalewisz menunjukkan bahwa penyajian dengan cara demekian mampu megalahkan AG dengan penyajian biner, dalam hal kebugaran (*fitness*) dan waktu komputasi CPU (*Central Processing Unit*). Sejak itu banyak operator khusus seperti *arithmatical crossover* dan mutasi tidak seragam diusulkan.

Pada dasarnya ada empat kondisi yang sangat mempengaruhi proses evolusi, yaitu :

1. Kemampuan organisme untuk melakukan reproduksi.
2. Kebeeradaan populasi organisme yang bisa melakukan reproduksi.
3. Keberagaman organisme dalam suatu populasi.
4. Perbedaan kemampuan untuk bertahan hidup.

Individu yang lebih kuat akan memiliki tingkat *survival* dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang lebih lemah. Pada kurun waktu tertentu (sering dikenal dengan istilah generasi), populasi secara keseluruhan akan lebih banyak memuat organisme yang kuat.

2.3 Teori Graf

Untuk mendiskusikan *ant colony system*, lingkungan yang akan kita gunakan adalah sebuah graf yang *fully connected* (setiap *node* memiliki busur ke *node* yang lain) dan *bidirectional* (setiap jalur bisa ditempuh bolak - balik dua

arah). Setiap busur memiliki bobot yang menunjukkan jarak antara dua buah *nodes* yang dihubungkan oleh busur tersebut.

Algoritma ini menggunakan sistem multi agen, yang berarti kita akan mengerahkan seluruh koloni semut yang masing-masingnya bergerak sebagai agen tunggal. Setiap semut menyimpan daftar tabu yang memuat node yang sudah pernah ia lalui, dimana ia tidak diijinkan untuk melalui node yang sama dua kali dalam satu kali perjalanan (daftar ini disebut juga sebagai jalur Hamilton, yaitu jalur pada graf dimana setiap node hanya dikunjungi satu kali).

Sebuah koloni semut diciptakan, dan setiap semut ditempatkan pada masing-masing node secara merata untuk menjamin bahwa tiap node memiliki peluang untuk menjadi titik awal dari jalur optimal yang dicari. Setiap semut selanjutnya harus melakukan tour semut, yaitu perjalanan mengunjungi semua node pada graf tersebut.

2.3.1 Definisi Graf

Graf adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lain melalui sisi/busur (*edges*). Suatu Graf G terdiri dari dua himpunan yaitu himpunan V dan himpunan E.

a. Verteks (simpul):V = himpunan simpul yang terbatas dan tidak kosong
b. Edge (sisi/ruas):E = himpunan busur yang menghubungkan sepasang simpul.

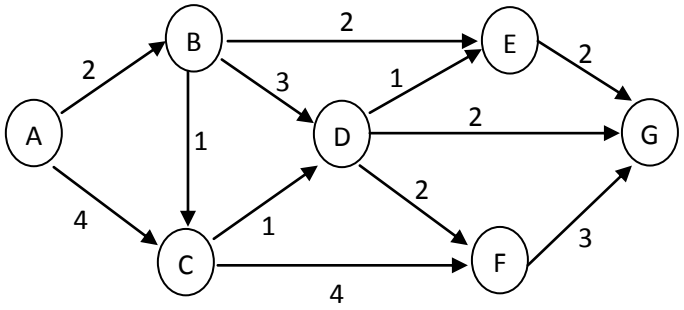
Simpul-simpul pada graf dapat merupakan obyek sembarang seperti kota, atom-atom suatu zat, nama anak, jenis buah, komponen alat elektronik dan sebagainya. Busur dapat menunjukkan hubungan (relasi) sembarang seperti rute penerbangan, jalan raya, sambungan telepon, ikatan kimia, dan lain-lain. Notasi graf: G(V,E) artinya graf G memiliki V simpul dan E busur.

2.3.2 Macam-macam Graf

Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu :

1. Graf berarah dan berbobot

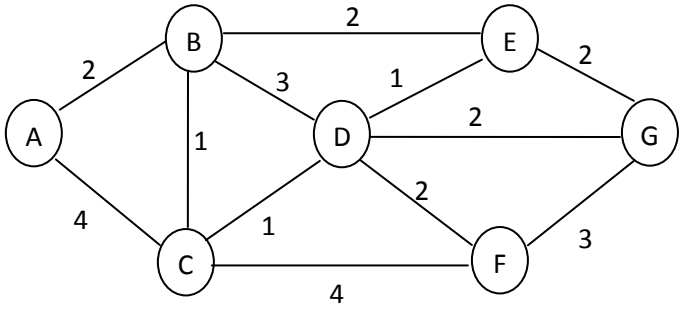
Tiap busur mempunyai anak panah dan bobot. Gambar 2.3 menunjukkan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menujukkan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan titik C, dan seterusnya. Bobot antar titik A dan titik B pun telah di ketahui.



Gambar 2.3 *Graf berarah dan berbobot*

2. Graf tidak berarah dan berbobot

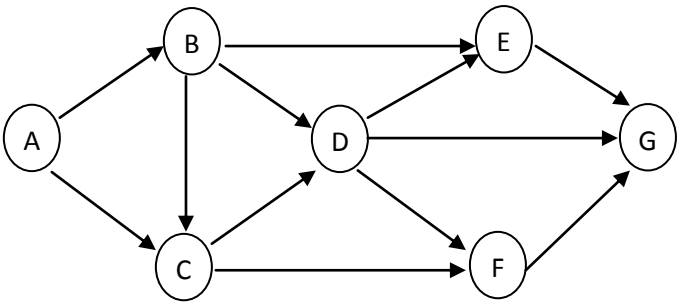
   Tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.4 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



Gambar 2.4 *Graf tidak berarah dan berbobot*

3. Graf berarah dan tidak berbobot

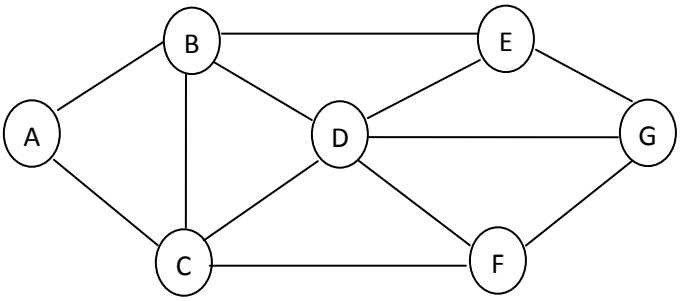
   Tiap busur mampunyai anak panah yang tidak berbobot. Gambar 2.5 menunjukan graf berarah dan tidak berbobot.



Gambar 2.5 Graf berarah dan tidak berbobot

4. Graf tidak berarah dan tidak berbobot

   Tiap busur tidak mempunyai anak panah dan tidak berbobot. Gambar 2.6 menunjukan graf tidak berarah dan tidak berbobot.



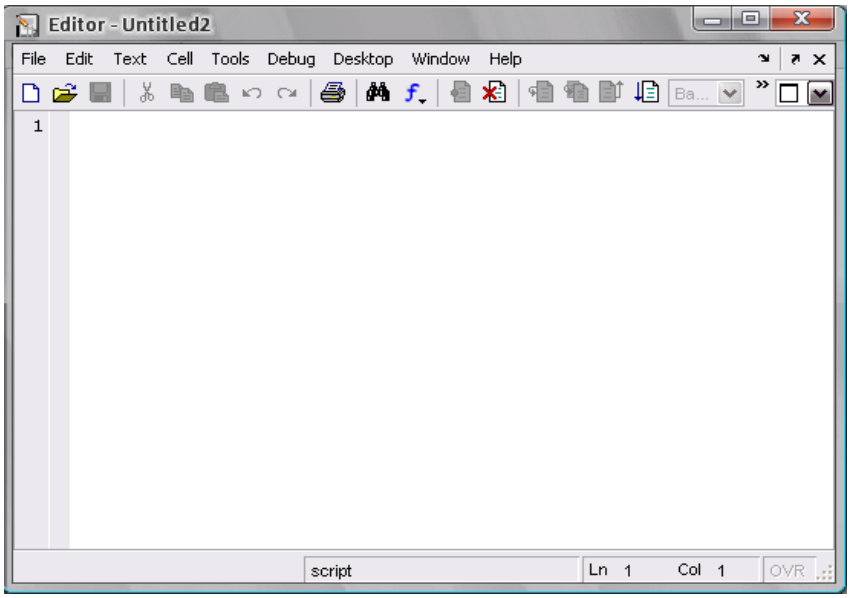
Gambar 2.6 *Graf tidak berarah dan tidak berbobot*

2.4 Tentang MATLAB

MATLAB adalah sebuah bahasa tingkat tinggi untuk komputasi teknis. MATLAB merupakan singkatan dari Matrix Laboratory. MATLAB mangintegrasikan perhitungan, visualisasi, dan pemrograman dalam suatu lingkungan yang mudah digunakan di mana permasalahan dan solusi dinyatakan dalam notasi secara matematis yang dikenal umum.

MATLAB merupakan bahasa pemrograman, yang sering digunakan pada teknik-teknik komputasi. MATLAB memiliki beberapa *feature* yang dikelompokan berdasarkan aplikasi tertentu yang dikenal dengan nama

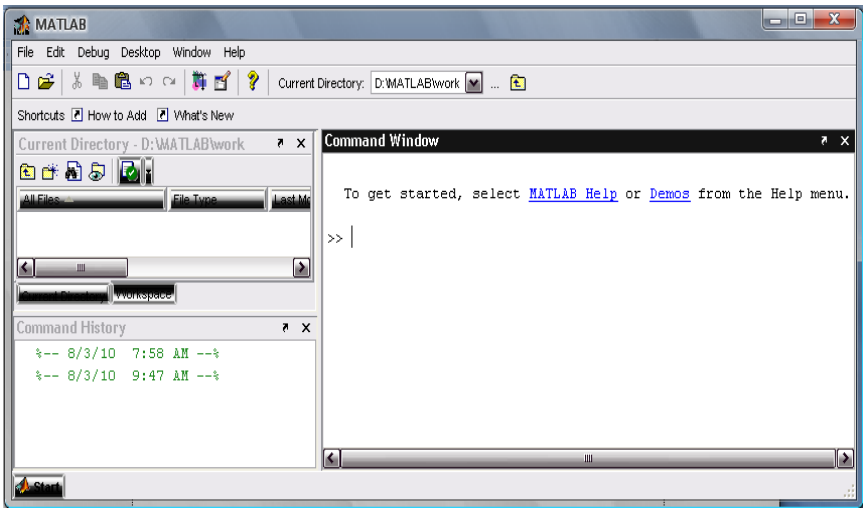
TOOLBOX. Dengan toolbox ini para pengguna diharapkan dapat mempelajari dan mengaplikasikan teknologi pada bidang kajian tertentu. Program yang ditulis dengan menggunakan MATLAB memiliki ekstensi m (.m). Untuk membuat program MATLAB dapat dilakukan dengan denggunakan MATLAB EDITOR seperti pada gambar 2.7 berikut.



Gambar 2.7 MATLAB Editor

2.4.1 Memulai MATLAB

Ketika pertama kali membuka MATLAB, akan muncul beberapa form seperti pada gambar 2.8



Gambar 2.8 Form MATLAB

Secara keseluruhan MATLAB memiliki beberapa buah jendela:

a. Jendela perintah (*command window*) seperti yang tampak di sebelah kanan gambar 2.8. Jendela perintah maupun tempat untuk memasukan perintah-perintah yang diinginkan. Baik perintah maupun keluaran MATLAB muncul dalam jendela perintah. Apabila terlalu lama dipakai, biasanya jendela perintah akan terasa sangat panjang. Untuk menghapus semua teks dalam jendela perintah, anda dapat memilih c*lear command window* dari menu edit.
b. Jendela daftar perintah (*command history*) seperti yang tampak di kiri bawah gambar 2.8. Jendela ini memuat daftar perintah yang pernah diketikan dalam jendela perintah. Untuk mengeksekusi kembali perintah yang dipakai, *drag* perintah tersebut ddari jendela daftar perintah ke jendela perintah.
c. Jendela *Help* yang dipakai jika mengalami kesulitan sewaktu memilih perintah atau formatnya.
d. Jendela *direktori*
e. Jendela *workspace*

2.5 Permasalahan Optimasi

2.5.1 Definisi Masalah Optimasi

Optimisasi adalah suatu proses untuk mencapai hasil yang optimal (nilai efektif yang dapat dicapai). Dalam disiplin matematika optimisasi merujuk pada studi permasalahan yang mencoba untuk mencari nilai minimal atau maksimal dari suatu fungsi riil. Untuk dapat mencapai nilai optimal baik minimal atau maksimal tersebut, secara sistematis dilakukan pemilihan nilai variabel integer atau riil yang akan memberikan solusi optimal.

2.5.2 Macam – Macam Permasalahan Optimasi

Permasalahan yang berkaitan dengan optimisasi sangat kompleks dalam kehidupan sehari-hari. Nilai optimal yang didapat dalam optimisasi dapat berupa

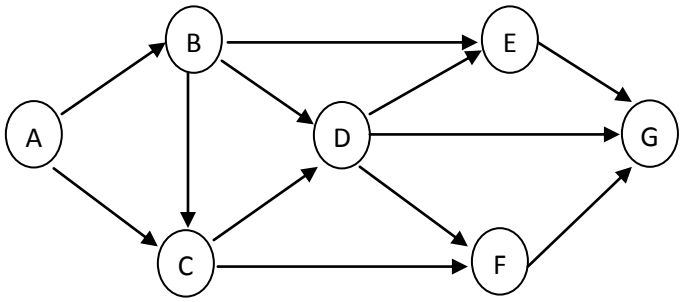
besaran panjang, waktu, jarak, dan lain-lain. Berikut ini adalah termasuk beberapa persoalan optimisasi :

1. Menentukan lintasan terpendek dari suatu tempat ke tempat yang lain.
2. Menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimalkan dan hasil produksi tetap maksimal.
3. Mengatur rute kendaraan umum agar semua lokasi dapat dijangkau.
4. Mengatur *routing* jaringan kabel telepon agar biaya pemasangan kabel tidak terlalu besar dan penggunaannya tidak boros.

Selain beberapa contoh di atas, masih banyak persoalan lainnya yang terdapat dalam berbagai bidang.

2.5.3 Permasalahan Jalur Terpendek

Jalur terpendek adalah suatu jaringan pengarahan perjalanan dimana sesorang pengarah jalan ingin menentukan jalur terpendek antara dua kota, berdasarkan beberapa jalur alternatif yang tersedia, dimana titik tujuan hanya satu. Gambar 2.9 menunjukkan suatu graf ABCDEFG yang berarah dan tidak berbobot.



Gambar 2.9 Graf ABCDEFG

Pada gambar di atas, misalkan kita dari titik A ingin menuju titik G. Untuk menuju titik G dapat dipilih beberapa jalur yang tersedia:

A→ B→ C→ D→ E→ G

A→ B→ C→ D→ F→ G

A → B→ C→ D→ G

A → B→ C→ F → G

A → B→ D→ E → G

A → B→ D→ F→ G

A → B→ D→ G

A → B→ E→ G

A → C→ D→ E → G

A → C→ D→ F → G

A → C→ D→ G

A → C→ F→ G

Berdasarkan data diatas, dapat dihitung jalur terpendek dengan mencari jarak antara jalur-jalur tersebut. Apabila jarak antar jalur belum diketahui, jarak dapat dihitung berdasarkan koordinat kota-kota tersebut, kemudian menghitung jalur terpendek yang dapat dilalui.

2.5.4 Penyelesaian Masalah Optimasi

Secara umum, penyelesaian masalah pencarian jalur terpendek dapat dilakukan dengan menggunakan dua metode, yaitu metode konvensional dan metode heuristik. Metode konvensional diterapkan dengan perhitungan matematis biasa, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan, dengan menentukan basis pengetahuan dan perhitungannya.

1. Metode Konvensional

   Metode konvensional berupa algoritma yang menggunakan perhitungan matematis biasa. Ada beberapa metode yang biasa digunakan untuk melakukan pencarian jalur terpendek, diantaranya algoritma Djikstraa, algoritma Floyd-Warshall, dan algoritma Bellman-Ford.

2. Metode Heuristik

   Metode Heuristik adalah sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan optimasi. Ada beberapa algoritma pada metode heuristik yang biasa digunakan dalam permasalahan optimasi, diantaranya algoritma genetika, algoritma semut, logika fuzzy, jaringan syaraf tiruan, pencarian tabu, *simulated annealing*, dan lain-lain.

2.6 Travelling Salesman Problem (TSP)

Permasalahan matematik yang berkaitan dengan *Traveling Salesman Problem* mulai muncul sekitar tahun 1800-an. Masalah ini dikemukakan oleh dua orang matematikawan, yaitu Sir William Rowan Hamilton yang berasal dari Irlandia dan Thomas Penyngton Kirkman yang berasal dari Inggris. Diskusi mengenai awal studi dari persoalan TSP ini dapat ditemukan di buku *Graph Theory* 1736-1936 by N.L. Biggs, E.K. LLoyd, and R.J. Wilson, Clarendon Press, Oxford, 1976. Bentuk umum dari persoalan TSP pertama kali dipelajari oleh para matematikawan mulai tahun 1930-an Karl Menger di Vienna dan Harvard. Persoalan tersebut kemudian dikembangkan oleh Hassler Whitney dan Merril Flood di Princeton.

Persoalan TSP merupakan salah satu persoalan *kombinatorial*. Sebagai permasalahan *kombinatorial,* permasalahan ini tergolong memiliki kemungkinan jawaban yang sangat banyak. Persoalan ini sendiri menggunakan representasi graf untuk memodelkan persoalan yang diwakili sehingga lebih memudahkan penyelesaiannya. Banyak permasalahan yang dapat direpresentasikan dalam

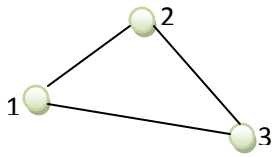
bentuk TSP. Diantara permasalahan yang dapat direpresentasikan dengan TSP adalah masalah transportasi, efisiensi pengiriman surat atau barang, perancangan pemasangan pipa saluran, proses pembuatan PCB (*Printed Cirtcuit Board*) dan lain-lain.

Berdasarkan kesesuaian dengan nama, deskripsi persoalan adalah sebagai berikut: diberikan sejumlah kota, tentukan sirkuit terpendek yang harus dilalui oleh seorang pedagang bila pedagang itu berangkat dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan. Kota dapat dinyatakan sebagai sebuah simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antara dua kota. Bobot pada sisi menyatakan jumlah antara dua buah kota. Persoalan ini adalah persoalan yang menentukan sirkuit Hamilton dengan sisi memiliki bobot minimum pada suatu graf terhubung.

Jika setiap simpul pada graf bobot mempunyai sisi ke simpul lain maka, graf tersebut adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul (n>2), jumlah sirkuit Hamilton yang berbeda adalah (n-1)!/2.

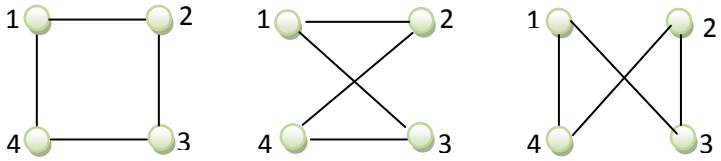
Persoalan TSP adalah persoalan yang sulit jika dipandang dari segi komputasi. Secara teoritis, TSP dapat diselesaikan dengan mengenumerasi seluruh kemungkinan sirkuit Hamilton yang ada lalu menghitung bobot total dari seluruh sirkuit Hamilton.

Berikut beberapa contoh penanganan permasalahan TSP:



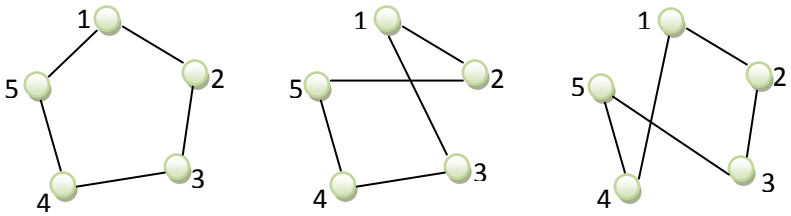
Gambar 2.10 TSP dengan 3 kota

Permasalahan 3 kota tidak memerlukan komputasi karena jumlah kemungkinannya solusinya hanya 1.



Gambar 2.11 TSP dengan 4 kota

Permasalahan 4 kota tidak memerlukan komputasi karena jumlah kemungkinannya solusinya hanya (4-1)!/2 = 3.



Gambar 2.12 TSP dengan 5 kota

Permasalahan 5 kota mulai memerlukan komputasi karena jumlah kemungkinan (5-1)!/2 = 12.

Permasalahan dengan jumlah kota yang semakin banyak akan menghasilkan semakin banyak kemungkinan. Untuk 20 kota akan menghasilkan 19!/2 kemungkinan atau sekitar 6,08 x 10^{16}. Namun, tidak semua travelling salesman problem melibatkan graf lengkap. Pada pemodelan kota, kasus ini dapat terjadi ketika ada kota yang tidak memiliki jalan penghubung langsung.

Persoalan yang muncul adalah bagaimana cara mengunjungi simpul (*node*) pada graf dari titik awal ke setiap titik-titik lainnya dengan bobot minimum (biaya paling murah). Bobot atau biaya ini sendiri dapat mewakili berbagai hal, seperti biaya, jarak, bahan bakar, waktu, kenyamanan dan lain-lain.

Pada TSP, jumlah jalur yang mungkin diperoleh dengan menggunkan rumus permutasi berikut ini:

$$P_n^k = \frac{n!}{(n-k)!} \quad \text{.....(2.8)}$$

dimana *n* adalah jumlah seluruh node dan *k* adalah jumlah node yang diseleksi. Terdapat dua jenis TSP, yaitu asimetris dan simetris. Pada TSP asimetris, biaya dari node 1 ke node 2 tidak sama dengan biaya dari node 2 ke node 1. Sedangkan pada TSP simetris, biaya dari node 1 ke node 2 sama dengan biaya dari node 2 ke node 1.

Untuk TSP asimetris, jumlah jalur yang mungkin adalah permutasi dari jumlah node dibagi dengan jumlah node. Hal ini dapat dipahami karena secara siklus, sebuah jalur dengan urutan 1-2-3 adalah sama dengan jalur 2-3-1 dan 3-1-2. Tetapi jalur dengan urutan 1-2-3 tidak sama dengan 3-2-1. Jadi apabila terdapat 10 node, maka jalur yang mungkin untuk TSP asimetris adalah:

$$P_{10}^{10} = \frac{10!}{10 \times (10-10)!} = 362.88 \quad \text{.....(2.9)}$$

Sedagkan untuk TSP simetris, jumlah jalur yang mungkin adalah permutasi dari jumlah node dibagi dengan dua kali jumlah node. Hal ini dapat dipahami karena secara siklus sebuah jalur dengan urutan 1-2-3 adalah sama dengan jalur 2-3-1 dan 3-1-2. Karena biaya dari node 1 ke node 2 sama dengan biaya dari node 2 ke node 1, maka jalur dengan urutan 1-2-3 sama dengan jalur 3-2-1. Jadi apabila terdapat 10 node, maka jalur yang mungkin untuk TSP simetris adalah:

$$P_{10}^{10} = \frac{10!}{2 \times 10 \times (10-10)!} = 181.440 \quad \text{.....(2.10)}$$

2.5.1 Contoh Perkembangan Masalah yang Muncul

Kode program komputer yang dibuat untuk menyelesaikan persoalan TSP telah berkembang semakin baik dari tahun ke tahun. Tanda yang paling mencolok dari perkembangan metode untuk menyelesaikan persoalan TSP adalah bertambahnya jumlah simpul (*node*) yang dapat diselesaikan, mulai dari solusi persoalan 49 kota yang dikembangkan oleh Dantzig, Fulkerson, dan Johnson's pada tahun 1954 sampai kepada solusi persoalan 24.978 kota pada tahun 2004, data-data tersebut didapat dari National Imagery and Mapping Agency, sebuah *database* nasional yang menyimpan nama-nama fitur geografi.