BAB III

LANDASAN TEORI

3.1 Optimasi

Menurut Budi Santoso (2011), Optimasi adalah suatu kumpulan formula matematika dan metode numerik untuk menemukan dan mengidentifikasikan kandidat terbaik dari sekumpulan alternative tanpa harus secara ekplisit menghitung dan mengevaluasi semua alternative yang mungkin. Banyak cara yang dapat dilakukan dalam menyelesaikan masalah untuk memberikan hasil terbaik. Cara untuk memberikan hasil terbaik ini disebut sistem optimasi. Sistem optimasi ini umumnya mengacu kepada teknik program matematika yang biasanya membahas atau mengacu kepada jalannya program penelitian (*research programming*) tentang masalah yang sedang dihadapi. Teknik ini diharapkan dapat memberikan solusi yang terbaik dari hasil keputusan yang telah diambil dari permasalahan yang sedang dihadapi tersebut.

Masalah optimasi biasanya dinyatakan dalam bentuk fungsi matematik. Optimasi adalah proses memaksimasi atau meminimasi suatu fungsi tujuan dengan tetap memperhatikan pembatas yang ada. Suatu fungsi didefinisikan sebagai suatu aturan yang menugaskan setiap pilihan nilai x dengan suatu nilai unik $y = f(x)$. Dalam hal ini x adalah variable independent dan y adalah variable *dependent*. Secara sistematis, misalkan kita punya set $S \subset R$, dimana R adalah set dari semua bilangan Riil. Kita bisa mendefinisikan suatu transformasi yang menugaskan satu nilai numerik untuk setiap $x \subset S$. Hubungan seperti ini

sering disebut dengan fungsi sklar *f* yang didefinisikan dalam set *S.* Permasalahan optimasi bisa dibagi menurut beberapa kategori, yaitu :

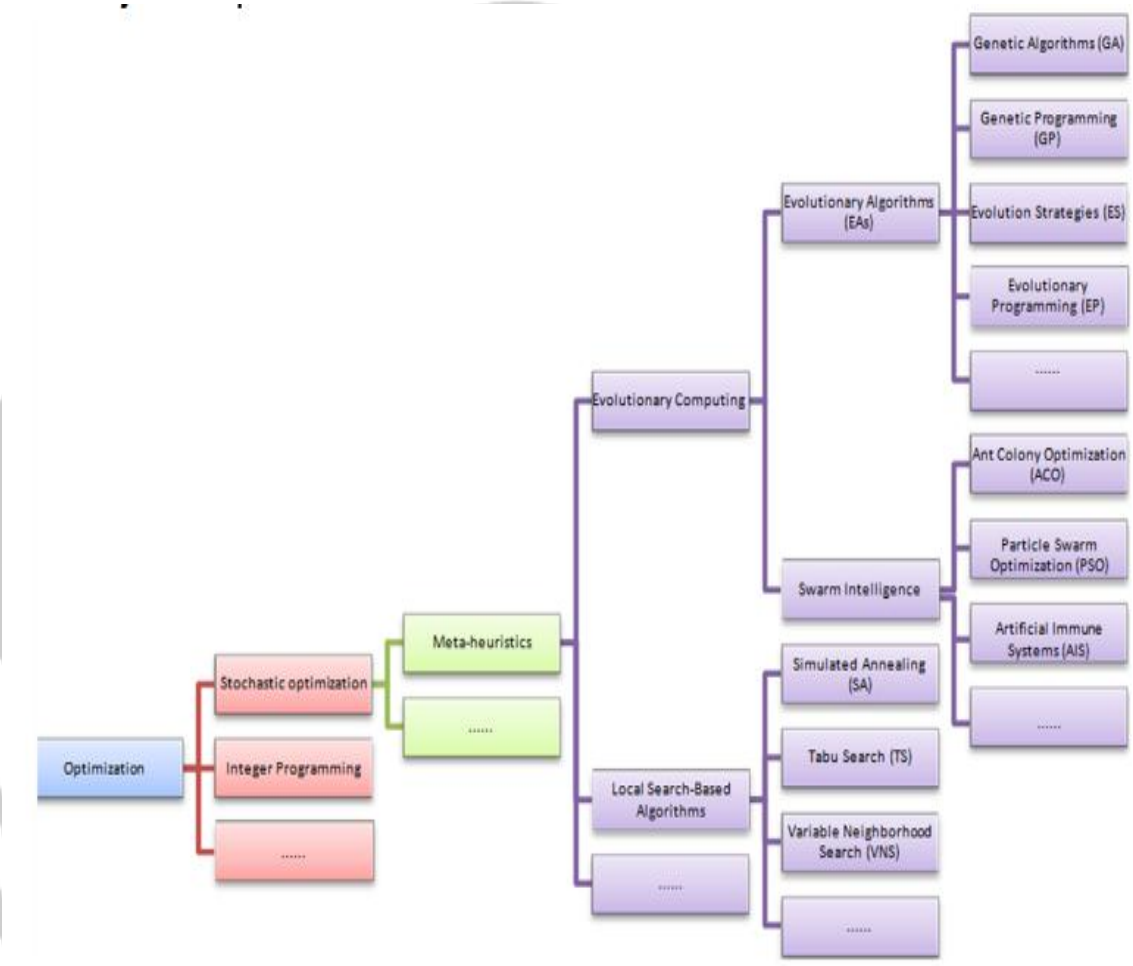
a. Optimasi tanpa batas, jika suatu fungsi *f* berlaku untuk *S* = *R*, maka disebut fungsi satu variabel tanpa batas atau *unconstrained function*.
b. Optimasi dengan pembatas, kalau S adalah subset dari R, maka kita mempunyai fungsi yang didefinisikan dalam daerah yang terbatas atau *constrained region.*

Selain itu masalah optimasi bisa juga dilihat dari nilai varibaelnya. Pengelompokan masalah optimasi berdasarkan nilai varibaelnya, yaitu :

a. Optimasi dengan variable kontinyu, masalah optimasi dengan nilai *x* bisa berapa saja dalam daerah *feasible*.
b. Optimasi diskrit, masalah optimasi dengan nilai solusi terbatas pada nilai-nilai tertentu yang biasanya bilangan bulat.

Untuk permasalahan optimasi seperti pembuatan jadwal kuliah yang mencakup ketersediaan dosen dan ruangan, persoalan transportasi yang mencakup pendistribusian suatu komoditas atau produk dari sejumlah sumber kepada sejumlah tujuan, penentuan rute terpendek, penentuan komposisi makanan ternak dengan biaya minimum yang harus memenuhi batasan minimal untuk setiap komponen nutrisi, dan lain-lain. Permasalahan-permasalahan tersebut dapat diselesaikan dengan menggunakan metoda-metoda optimasi. Berikut cara mengelompokkan masalah optimasi

dan metoda yang digunakan, untuk menyelesaikan masalah optimasi, disajikan dalam gambar 3.1 dibawah ini :



Gambar 3.1 Mengelompokkan masalah optimasi dan metoda yang digunakan dalam menyelesaikan masalah optimasi

3.2 Metaheuristik

Untuk penyelesaikan kasus khusus seperti diatas dapat digunakan metode heuristik, yaitu suatu metode pencarian yang didasarkan atas intuisi atau aturan-aturan empiris untuk memperoleh solusi yang lebih baik daripada solusi yang telah dicapai sebelumnya. Didalam ilmu komputer, metode heuristik merupakan suatu proses yang mungkin dapat menyelesaikan suatu masalah tetapi tidak ada jaminan bahwa solusi yang dicari selalu dapat ditemukan. Heuristik hanyalah sebuah cara menerka langkah berikutnya yang harus diambil dalam memecahkan suatu persoalan berdasarkan informasi yang tersedia atau suatu metode penyelesaian yang menggunakan konsep pendekatan.

Menurut Talbi (2009), *meta-heuristic* dapat didefinisikan sebagai metode lanjut berbasis *heuristic* untuk menyelesaikan persoalan optimisasi secara efisien. Metode ini mampu menghasilkan penyelesaian yang baik dalam waktu yang cepat, tetapi tidak menjamin bahwa penyelesaian yang dihasilkan merupakan penyelesaian terbaik (optimal). Dalam mencari solusi *meta-heuristic* memadukan interaksi antara procedur pencarian lokal dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik lokal optimal. Istilah lain dalam *meta-heuristic* adalah *derivative-free*, *direct search*, *black-blox*, atau *heuristic optimizer*. Beberapa karakteristik umum yang biasa dimiliki oleh pendekatan *meta-heuristic*, yaitu :

a. Dipakai dalam masalah optimasi *stokhastic*
b. Tidak menggunakan perhitungan gradient dari fungsi tujuan

c. Diinspirasi dari analogi fisik, seperti : ethology yaitu: *ant colony optimization, particle swarm optmization, evolutionary algorithms*.

d. Kesulitan mengatur nilai parameter dan komputasi yang lama.

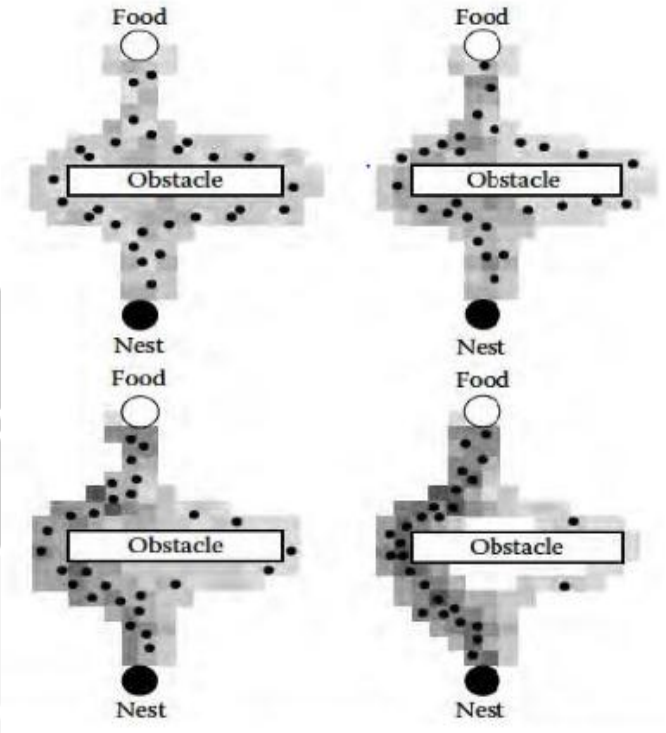
3.2.1 Ant Colony Optimization

Ant Colony Optimization (ACO) termasuk dalam kelompok *Swarm Intelligence*, yang merupakan salah satu jenis pengembangan paradigma yang digunakan untuk menyelesaikan masalah optimasi dimana inspirasi yang digunakan untuk memecahkan masalah tersebut berasal dari perilaku kumpulan atau kawanan (*swarm*) serangga. ACO biasanya digunakan untuk menyelesaikan *discrete optimization problems* dan persoalan yang kompleks dimana terdapat banyak variabel. Hasil yang diperoleh dengan menggunakan ACO, walaupun tidak optimal namun mendekati optimal.

Ant based techniques pertama kali digunakan oleh Dorigo et. al Dorigo et al. [1996] dengan menggunakan ACO untuk menyelesaikan *Traveling Salesman Problem* (TSP). Konsep *Ant System* sendiri terinspirasi dari pengamatan terhadap tingkah laku semut. Semut merupakan serangga sosial yang hidup dalam koloni-koloni dan berperilaku berdasarkan kepentingan koloni. Misalkan ada segerombolan semut yang berada di depan harus memilih lintasan tertentu untuk dilewati. Pada saat semut pertama berjalan, semut tersebut meninggalkan hormon pheromone yang dapat dicium oleh semut berikutnya, sehingga semut-semut berikutnya tahu apakah tempat

tersebut sudah dilewati atau belum. Dengan demikian, setiap semut yang berjalan akan meninggalkan pheromone pada jalur yang dilaluinya.

Semut yang melewati lintasan yang pendek akan meninggalkan aroma *pheromone* yang lebih tajam daripada yang menempuh lintasan yang lebih panjang. Hal ini terjadi karena *pheromone* yang ditinggalkan dapat menguap. Saat semut-semut yang di belakang mengikuti semut-semut yang di depannya, semut-semut tersebut akan memilih lintasan berdasarkan kuatnya aroma *pheromone* dan jarak lintasan. Semakin banyak semut yang menempuh suatu lintasan tertentu, maka aroma *pheromone* pada lintasan tersebut akan semakin kuat sehingga semut-semut berikutnya akan mengikuti lintasan yang sama, sehingga jalur terpendek akan ditemui karena lebih banyak semut yang akan melewati jalur tersebut. Berikut perubahan konsentrasi *pheromone*, disajikan pada Gambar 3.2, sebagai berikut :



Gambar 3.2 Perubahan konsentrasi *pheromone*

Proses dalam ACO bisa dijelaskan sebagai berikut. Misalkan ada N semut dalam satu koloni. Semut-semut itu akan memulai gerakan dari sarang mereka menuju tujuan akhir melalui beberapa simpul dan berakhir pada simpul tujuan di akhir setiap siklus atau iterasi. Kalau semua semut sudah menyelesaikan lintasannya, jumlah pheromone pada lintasan terbaik secara global akan diperbarui. Lintasan global terbaik

artinya terbaik diantara semua semut. Pada awal proses, yaitu pada iterasi 1, semua ruas dari simpul awal akan diberi jumlah pheromone yang sama. Sehingga pada iterasi 1, semua semut akan mulai dari simpul awal dan berakhir pada simpul tujuan

dengan cara memilih simpul-simpul antara secara random. Proses optimasi berakhir jika jumlah iterasi maksimum sudah tercapai atau tidak ada lagi solusi yang lebih baik yang bisa didapat dalam beberapa iterasi yang berturutan.

3.2.1.1 Perilaku Semut

Seekor semut *k* pada simpul *i* akan memilih simpul *j* yang dituju pada pada tingkatan berikutnya dengan probabilitas :

$$p_{i,j} = \begin{cases} \dfrac{\tau_{i,j}^{\alpha}}{\sum_{j \in N_i^{(k)}} \tau_{i,j}^{\alpha}} & , \; jika\; j \in N_i^{(k)} \\ 0 \; , \quad jika\; j \in N_i^{(k)} & \end{cases} \qquad (3.1)$$

Dimana :

P = probalitas

α = derajat kepentingan *pheromone*

$N_i^{(k)}$ = pilihan yang dipunyai semut *k* pada saat berada pada simpul *i* tetangga

3.2.1.2 Penambahan Pheromon

Seekor semut *k* ketika melewati ruas akan meninggalkan pheromone. Jumlah pheromone yang terdapat pada ruas *ij* setelah dilewati semut *k* diberikan dengan formula sbagaicberikut :

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^{k} \qquad (3.2)$$

Dimana:

τ = pheromone

i,j = simpul

dengan meningkatnya nilai *pheromone* pada ruas *i,j* maka kemungkinan ruas ini akan dipilih kembali kembali pada iterasi berikutnya semakin besar.

3.2.1.3 Penguapan Pheromon

Setelah sejumlah simpul dilewati maka akan terjadi penguapan *pheromone* dengan formula sebagai berikut:

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j};\ \forall (i,j) \in A \qquad (3.3)$$

Dimana:

$\rho =$ parameter tingkat penguapan

A = ruas yang telah dilalui olek semut k sebagai bagian dari lintasan dari sarang menuju makanan

Penguapan *pheromone* memungkinkan semut untuk mengekplorasi lintasan yang berbeda selama proses pencarian. Hal ini akan menghilangkan kemungkinan memilih lintasan yang kurang bagus. Jumlah *pheromone* yang ditambahkan pada ruas *i,j* oleh semut dibuat formula sebagai berikut:

$$\Delta\tau_{ij}^{(k)} = \frac{Q}{L_k} \qquad (3.4)$$

Dimana:

Q = konstanta

L = lintasan terpendek

3.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) merupakan salah satu metode *meta-huristik* yang diperkenalkan oleh Kennedy dan Eberhart pada tahun 1995. Algoritma ini mengadopsi perilaku sosial dari perilaku kumpulan atau kawanan (*swarm*) hewan seperti kawanan burung atau ikan. Perilaku sosial yang dimaksud disini adalah tindakan dari individu dari kumpulan atau kawanan tersebut dan pengaruh dari individu-individu lain dalam kumpulan atau kawanan tersebut. Di dalam PSO, kumpulan atau kawanan hewan ini diumpamakan sebagai kumpulan partikel yang memiliki posisi dan kecepatan. Posisi dan kecepatan inilah yang akan selalu diperbarui di setiap iterasi PSO hingga ditemukan posisi partikel yang optimal.

Pencarian solusi pada PSO dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara random dengan batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi terbaik dari partikel tersebut (*local best*) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. Berikut formulasi yang menggambarkan posisi dan kecepatan partikel pada suatu dimensi ruang tertentu :

$$X_i(t) = x_{i1}(t), x_{i2}(t), \dots x_{iN}(t) \quad (3.5)$$

$$V_i(t) = v_{i1}(t), v_{i2}(t), v_{i3}(t), \dots v_{iN}(t) \quad (3.6)$$

Dimana :

X = posisi partikel

V = kecepatan partikel

i = indek partikel

t = iterasi ke-t

N = ukuran dimensi ruang

Untuk menghitung kecepatan partikel yang baru berdasarkan kecepatan sebelumnya, dihitung dengan mekanisme *updating* status partikel diformulasikan dengan persamaan (3.7) dan untuk memperbaharui posisi setiap partikel berdasarkan dari kecepatan baru yang telah didapat menggunakan formulasi seperti dibawah ini (3.8):

$$v_i(t) = v_i(t-1) + c_1 r_1 \left(X_i^L - X_i(t-1)\right) + c_2 r_2 (X^G - X_i(t-1)) \qquad (3.7)$$

$$X_i(t) = V_i(t) + X_i(t-1) \qquad (3.8)$$

Dimana:

X_i^L= merepresentasikan *Local best* dari partikel

X^G= merepresentasikan *Global best* dari seluruh kawanan

$c_1$dan c_2 = konstanta yang bernilai positif, yang disebut sebagai *learning factor*

r_1 dan r_2 = suatu bilangan random yang bernilai antara 0 dan 1

Kecepatan partikel pada saat iterasi *i*, dan kecepatan untuk setiap partikel *j*, di hitung dengan formula sebagai berikut:

$$v_j(i) = v_j(i-1) + c_1 r_1[P_{best,j} - x_j(i-1)] + c_2 r_2\left[G_{best} - x_j(i-1)\right] \quad (3.9)$$

3.2.2.1 Modifikasi Particle Swarm Optimization

Pada kenyataannya ditemukan bahwa kecepatan partikel dalam PSO standart diupdate terlalu cepat dan nilai minimum fungsi tujuan yang dicari sering terlewati. Perbaikan dilakukan pada penambahan suatu term inersia θ untuk mengurangi kecepatan pada formula update kecepatan. Perbaikan dengan menambahkan θ untuk meredam kecepatan selama iterasi, yang memungkinkan kawanan burung menuju konvergen titik target secara akurat dan efisien, dibuat formula seperti dibawah ini:

$$v_j(i) = \theta v_j(i-1) + c_1 r_1[P_{best,j} - x_j(i-1)] + c_2 r_2\left[G_{best} - x_j(i-1)\right] \quad (3.10)$$

Dimana:

θ = inersia

3.3 Algoritma Hybrid

Pengertian *Hybrid* sendiri adalah penggabungan dua unsur yang berlawanan tetapi tetap mempertahankan karakter unsur - unsur tersebut. Konsep *hybrid* sendiri baru mulai dikenal oleh masyarakat umum sejak diterapkannya pada konsep mobil *hybrid.* Konsep *hybrid* ini tidak hanya bisa diterapkan pada mobil saja, namun bisa

juga diterapkan dalam algoritma. Algoritma *hybrid* adalah algoritma yang menggabungkan dua atau lebih algoritma lain yang memecahkan masalah yang sama, baik memilih satu (tergantung pada data), atau beralih di antara keduanya selama algoritma tersebut berlangsung. Hal ini umumnya dilakukan untuk menggabungkan fitur yang diinginkan masing-masing, sehingga keseluruhan algoritma lebih baik daripada komponen individual. Algoritma *Hybrid* tidak mengacu pada hanya menggabungkan beberapa algoritma untuk memecahkan masalah yang berbeda, banyak algoritma dapat dianggap sebagai kombinasi potongan yang lebih sederhana, namun hanya untuk menggabungkan algoritma yang memecahkan masalah yang sama, namun berbeda dalam karakteristik lain, terutama kinerja.

3.4 Penjadwalan

Menurut Rosani Ginting (2000), penjadwalan adalah pengalokasian sumber daya pada objek-objek yang ada pada ruang waktu dan tergantung pada kendala-kendala yang sedemikian sehingga memenuhi sekumpulan sasaran yang diinginkan, secara sederhana, penjadwalan dapat diartikan sebagai pengalokasian sumber-sumber daya yang tersedia pada ruang waktu yang ada sehingga memenuhi kondisi-kondisi tertentu. Tujuannya adalah untuk memaksimalkan suatu proses dengan tetap menjaga agar tidak melanggar *constraint* yang berlaku pada proses yang bersangkutan.

Penjadwalan perkuliahan atau mata pelajaran merupakan kegiatan yang sangat penting untuk proses belajar mengajar. Proses belajar mengajar dilaksanakan oleh seluruh mahasiswa dan dosen atau siswa dan guru, sehingga jadwal perkuliahan yang disusun harus dapat memfasilitasi kepentingan dosen dan mahasiswa atau guru dan siswa-siswi. (Sugiarto 2007;90).

Didalam penjadwalan perkuliahan dikenal dua macam *constraint* (persyaratan) yaitu *Hard Constraint* dan *Soft Constraint*. *Constraint* penjadwalan perkuliahan yang berlaku tentunya berbeda tergantung dari regulasi dari Perguruan Tinggi tersebut dan tata cara Perguruan Tinggi yang bersangkutan. *Hard Constraint* adalah memiliki prioritas lebih tinggi dibanding *Soft constraint,* dan akan menjadi pertimbangan utama untuk bisa dipenuhi. Kenyataanya, penjadwalan biasanya hanya akan mempertimbangkan sebagian atau keseluruhan *hard constraints* dari suatu masalah yang telah dipenuhi. Sedangkan *soft constraints*, merupakan *constraint* atau pertimbangan yang bisa dilaksanakan jika memungkinan dan biasanya menjelaskan apa saja yang bisa dipertimbangkan disesuaikan dengan kebijakan Perguruan Tinggi tersebut.