BAB II TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

2.1.1 Timetable

Timetable merupakan alokasi subjek yang memiliki kendala untuk ditempatkan pada *ruang waktu* (Gani dkk, 2004). Permasalahan *Timetable* cukup luas. Masalah ini ada pada keseharian kehidupan, pendidikan, kesehatan, transportasi, olahraga dan perusahaan produksi (Chu dkk, 2006, Kumar dkk, 2012, Norberciak, 2006, Mushi, 2006). *Timetable* Universitas merupakan tabel yang digunakan untuk mengkoordinasi siswa, dosen, ruang dan sumberdaya lain (wikipedia, 2013). Proses penyelesaian *Timetable* yang optimal cukup rumit dan memakan waktu jika diselesaikan secara manual (Tariq dkk, 2010, Montero dkk, 2011, Norberciak, 2006). Dari permasalahan tersebut, diperlukan metode yang dapat menyelesaikan *timetable* dengan cepat dan presisi.

Telah banyak upaya peneliti untuk mendesain perangkat lunak yang dapat menciptakan *timetable* secara otomatis. Tetapi, sampai saat ini belum ada yang secara penuh menghasilkan solusi yang optimal. Selain itu juga masih belum memungkinkan untuk menciptakan *timetable* secara otomatis dimana semua kendala terselesaikan dengan baik (Kumar dkk, 2012).

Ada beberapa variasi metode dari tujuan pemecahan masalah dalam pembuatan timetable dan pembuatan *schedule* (Norberciak 2009). Variasi ini dibagi menjadi empat tipe :

a. Metode Sekuensial, metode ini memberi perintah kepada *event* dengan menggunakan domain heuristik dan memasukkan *event* secara berurutan ke dalam periode waktu yang valid atau disebut juga *time slot* dimana tidak ada *event* dalam setiap periode yang saling bermasalah.
b. Metode *cluster*, pada metode ini *event* masuk ke dalam cluster dimana setiap dua *event* masuk ke dalam *cluster* khusus. Kekurangan utama pendekatan ini ialah *cluster* dari *event* terbentuk dari awal dan bersifat fix/tetap dan akan menghasilkan kualitas *timetable* yang kurang baik.
c. Metode berbasis kendala, pada metode ini, kendala yang ada terbentuk dari beberapa variabel domain yang terbatas tergantung dengan sumber daya seperti periode waktu dimana *timetable* ditetapkan untuk memenuhi sejumlah kendala. Pada metode berbasis kendala, dapat terjadi dimana tidak semua kendala dapat terpenuhi.
d. Metode Metaheuristik, variasi dari pendekatan metaheuristik untuk menyelesaikan *timetable* yaitu, *Simulated Annealing (SA)*, *Tabu Search*, *Evolutionary Algorithm*, dan pendekatan lain. Metode metaheuristik dimulai dengan satu atau lebih solusi awal dan akan melakukan strategi pencarian untuk mendapatkan solusi yang optimal. Metode ini juga berupaya untuk menghindari *local optima*.

Pada masalah *timetable* yang lebih spesifik seperti pada perkuliahan, ada beberapa sesi pada satu hari. Ada beberapa hari kuliah, yang berarti ada cukup banyak baris dan kolom tabel. Jika dalam sehari ada empat sesi dan dalam seminggu ada lima hari kuliah, sedangkan kampus memiliki empat

ruang kuliah, maka ada delapan puluh kolom yang akan terbentuk dalam tabel dari *timetable* perkuliahan.

Metode-metode yang pernah digunakan untuk menyelesaikan timetable adalah *Traditional Forward Checking,* Algoritma Genetik (GA) (Rawat dan Rajamani, 2005-2010), PSO (Chu dkk, 2006, Montero dkk, 2011), *Metaheuristic* (Gani dkk, 2004). *Traditional Forward Checking* membutuhkan waktu hitungan jam untuk menyelesaikan *timetable* (Montero dkk, 2011). Pada masalah nyata, metode PSO dapat menyelesaikannya dalam waktu yang cepat (Montero dkk, 2011).

Dalam penerapan PSO untuk masalah *timetable* universitas, urutan pemilihan mata kuliah dianggap sebagai partikel, dan tabel mata kuliah dapat dianggap sebagai suatu ruang waktu yang terdiri atas beberapa dimensi. Tujuan utama pada perhitungan untuk mencari *timetable* yang optimal ialah untuk meminimalkan pelanggaran terhadap kendala (Rawat dan Rajamani, 2005-2010).

Berdasar jenisnya, ada 2 jenis kendala, yaitu kendala yang bersifat tegas dan kendala yang bersifat lemah (Gani dkk, 2004, Rawat dan Rajamani, 2005-2010, Mushi, 2006). Kendala yang bersifat tegas harus dipenuhi, sehingga tidak boleh terjadi pelanggaran terhadap kendala ini. Sedangkan kendala yang bersifat lemah tidak harus dipenuhi. Kendala ini lebih rumit diselesaikan daripada yang bersifat tegas. (Chu dkk, 2006).

2.1.2 Particle Swarm Optimization (PSO)

Teknologi cerdas meniru manusia untuk belajar dari pengalaman mengatasi ketidaktepatan data, respon yang cepat terhadap situasi, dan menambah pengetahuan. Komputasi evolusioner menghasilkan suatu solusi pada ruang pencarian di dalam populasi. Komputasi evolusioner berbasis pada populasi untuk melihat solusi terbaik dari suatu ruang. Particle Swarm Optimization (PSO) merupakan teknik dari komputasi evolusioner yang cukup banyak digunakan (Devi dan Panigrahi, 2011)

PSO adalah salah satu algoritma terbaru dari *Swarm Intelligence* (SI), dibandingkan dengan Algoritma Genetik (GA) dan *Simulated Annealing* (SA) (Hsieh dkk, 2007., Huang dkk, 2011). PSO mirip dengan GA, hanya saja tidak memiliki operator seleksi, crossover dan mutasi (Mozafaria dkk, 2011., Chen, 2010). PSO telah berhasil didemonstrasi untuk optimasi numerik, dan dinyatakan bahwa PSO merupakan alternatif dari global optimization. Evolusi natural pada PSO diterapkan untuk memunculkan generasi baru dari calon solusi.

Keuntungan utama algoritma PSO ialah konsep yang mudah diimplementasikan dan memiliki efisiensi komputasi yang baik (Park dkk, 2005). Selain itu juga PSO cukup cepat dan efektif untuk direalisasikan (Xue-fei, Yun-Xia, 2008). Performa dari PSO lebih baik dari metode pencarian heuristik lain seperti GA (Kumar dan Reddy, 2007). PSO banyak diterapkan dalam memecahkan masalah-masalah optimasi (Mozafaria dkk, 2011).

PSO awalnya dikembangkan oleh Kennedy dan Elberhart pada tahun 1995 dengan meniru perilaku kawanan burung dalam mencari makanan (Shahzad dkk,.

2010., Kanthavel dkk, 2012., Chen dkk, 2006., Pang dkk, 2004, Park dkk, 2005., Kiruthiga dan Senthilkhumar, 2011., Zhang dan Xiong, 2008). PSO didasarkan pada pertukaran informasi antar individu (partikel) dan antar populasi (*swarm*) (Chen, 2010). Pada saat bergerak, masing masing individu membuat keputusan sesuai dengan pengalaman diri sendiri dan individu lain (Niasar dkk, 2009., Zhen dkk, 2009., Pang dkk, 2004).

Optimasi pada PSO seperti dalam skenario dimana kawanan burung mencari makanan. Pada perjalanan yang acak hanya ada satu lokasi tertentu dari setiap wilayah. Meskipun tidak semua burung tahu dimana letak lokasi makanan, tetapi mereka tahu seberapa jauh lokasi mereka sekarang dari makanan. Strategi ini cukup sederhana dan efektif. PSO ditemukan dari model ini dan diaplikasikan dalam permasalahan optimasi (Huang dkk, 2011).

Pada PSO setiap solusi untuk masalah optimasasi dapat diartikan sebagai kawanan mencari solusi dalam *space*. Sekelompok individu disebut juga partikel (Huang dkk, 2011). Setiap partikel merupakan solusi untuk masalah optimasi (Huang dkk, 2011, Kiruthiga dan Senthilkhumar, 2011). Performa dari pergerakan setiap partikel diukur dengan menggunakan fungsi *fitness* yang bervariasi tergantung dari masalah optimasi yang ada (Kiruthiga dan Senthilkhumar, 2011). Semua partikel memiliki nilai *fitness* yang ditentukan melalui fungsi optimasi, dan setiap partikel memiliki kecepatan pencarian sesuai dengan arah dan jarak yang ada (Huang dkk, 2011).

Proses yang terjadi dari algoritma PSO yaitu, semua partikel akan bergerak mengikuti partikel yang optimal dalam ruang untuk mencari solusi optimal (Chen,

2010., Kiruthiga dan Senthilkumar, 2011., Altay dan Kayakutlu, 2011., Niasar dkk, 2009). Posisi baru pada setiap partikel berhubungan dengan kecepatan dan jarak antara posisi yang ada dengan *local best position* serta *global best position*. Algoritma akan beriterasi dan *swarm* akan bergerak ke area yang memiliki solusi yang lebih baik (Mozafaria, 2011., Zhang dan Xiong, 2008., Niasar dkk, 2009).

Setiap partikel mencari vektor lbest terbaik pada setiap topologi lokal daerah partikel. Gbest terbaik ditentukan oleh partikel di seluruh swarm (Kiruthiga dan Senthilkhumar, 2011).

Persamaan yang banyak digunakan dan cukup mendasar dalam PSO :

$$X_i(k+1) = X_i(k) + V_i(k+1) \quad \text{.....(1)}$$

$$V_i(k+1) = wV_i(k) + c_1 r_1 (P_i - x_i(k)) + c_2 r_2 (P_g - x_i(k)) \quad \text{.....(2)}$$

Persamaan 2 jika dituliskan dalam bahasa yang lebih detail ialah sebagai berikut :

$V_{ij} = w * v_{ij} + c1 * rand1(pbest_{ij} - particle_{ij}) + c2 * rand2(gbest_{ij} - particle_{ij})$ .....(3)

Persamaan persamaan tersebut diikuti dengan

$Particle_{ij} = particle_{ij} + v_{ij}$ .....(4)

Berikut ini definisi dari rumus yang telah dijabarkan:

x_{ij} – posisi partikel saat ini

v_{ij} – kecepatan particle saat ini,

$pbest_{ij}$ - posisi terbaik partikel dalam satu kawanan.

$gbest_{ij}$ - posisi terbaik partikel dari semua kawanan.

c_1 = komponen sosial pertama, memiliki nilai positif acak antara 0 dan 1 yang bersifat konstan. c_1 digunakan untuk menentukan kecepatan pergerakan partikel dalam mendekati nilai *Local best*.

c_2 = komponen sosial kedua, memiliki nilai positif acak antara 0 dan 1 yang bersifat konstan. c_2 digunakan untuk menentukan kecepatan pergerakan partikel dalam mendekati nilai Global best.

w = inertia weight particle swarm. w digunakan untuk menentukan kecepatan pergerakan terhadap diri sendiri (v_i).

Rata-rata implementasi menggunakan setting dengan c1 = c2. Penggunaan *inertia weight* 'w' lebih jarang dilakukan. *Inertia weight* memiliki nilai standar 0.4 sampai 0.9 (Kiruthiga dan Senthilkhumar, 2011).

Algoritma PSO memiliki kelemahan yaitu tak dapat digunakan untuk mengatasi data diskret (Sun dan Lei, 2009). Permasalahan data diskret diselesaikan dengan adanya DPSO.

2.1.3 Discrete Particle Swarm Optimization (DPSO)

DPSO digunakan untuk mengatasi *discrete space* dimana particle ter-update (Tasgetiren dkk, 2007., Fan, 2010). Metode DPSO yang merupakan ide dari Shi (Shi dkk, 2007., Venkatesan dkk, 2011). Metode ini cukup efektif untuk memecahkan masalah *Generalized* TSP yang dalam prosesnya melakukan perjalanan melewati setiap titik dengan hasil minimal (Afaq dan Saini, 2011).

Metode DPSO menggunakan konsep permutasi untuk menyelesaikan masalah bertipe diskret. Pada DPSO, $X_i=(x_{i1}, x_{i2}, \ldots, x_{im})$ sebagai posisi partikel ke i pada populasi dari PSO dimana merepresentasikan lingkaran pergerakan dari $x_{i1} \rightarrow x_{i2} \rightarrow \ldots \rightarrow \ldots x_{im} \rightarrow x_{i1}$. Algoritma DPSO yang konvensional didefinisikan melalui persamaan (1)-(2). untuk masalah TSP, persamaan 1 dapat

dijalankan secara formal saat arti item kedua dari bagian lain dari persamaan $V_i(k+1)$berubah. Persamaan (2) dapat didefinisikan kembali. Pertama kali, pembagi dari dua posisi parikel harus didefinisikan ulang.

Rumus DPSO :

$$x_i(t+1) = x_1(t) + wV_i(t) + c_1 r_1 \left(P_i - x_i(t)\right) + c_2 r R_2 \left(P_g - x_i(t)\right) \quad \text{.....(5)}$$

Berikut ini variabel yang digunakan pada rumus (5)

t = waktu saat ini

V_i = kecepatan partikel

x_i = posisi partikel saat ini

P_i = Local best / Pbest

P_g = Global best / Gbest

r = Random

i = partikel

c_1 = komponen sosial pertama, memiliki nilai positif acak antara 0 dan 1 yang bersifat konstan. c_1 digunakan untuk menentukan kecepatan pergerakan partikel dalam mendekati nilai *Local best*.

c_2 = komponen sosial kedua, memiliki nilai positif acak antara 0 dan 1 yang bersifat konstan. c_2 digunakan untuk menentukan kecepatan pergerakan partikel dalam mendekati nilai *Global best*.

w = *inertia weight particle swarm*. w digunakan untuk menentukan kecepatan pergerakan terhadap diri sendiri (V_i).

Rata-rata implementasi menggunakan setting dengan c1 = c2. Penggunaan *inertia weight* 'w' lebih jarang dilakukan.*Inertia weight* memiliki nilai standar 0.4 sampai 0.9 (Kiruthiga dan Senthilkhumar, 2011).

2.2 Landasan Teori

PSO memiliki dasar kecerdasan yang cukup baik. PSO digunakan untuk memecahkan permasalahan kontinyu seperti perbedaan takaran untuk pembuatan beberapa jenis roti dengan jumlah bahan yang sama, permasalahan VRP (Vehicle Routing Problem), dan perhitungan kontinyu lain. Sedangkat untuk permasalahan discrete seperti TSP, pencarian jarak minimum, dapat diselesaikan dengan DPSO. DPSO maupun PSO memiliki dasar rumus yang sama. Rumus DPSO maupun PSO adalah sebagai berikut.

$$X_i(k+1) = x_i(k) + V_i(k+1) \quad \text{.....(6)}$$

$$V_i(k+1) = wV_i(k) + c_1 r_1 (P_i - x_i(k)) + c_2 r_2 (P_g - x_i(k)) \quad \text{.....(7)}$$

Walaupun memiliki rumus yang sama, ada perbedaan penerapan dalam dalam perhitungan. Pada PSO, rumus dapat diterapkan secara langsung. DPSO didasarkan pada konsep permutasi. PSO maupun DPSO menggunakan satu atau lebih swarm, dengan partikel sejumlah n. Posisi terbaik lokal merupakan posisi terbaik dari setiap swarm. Sedangkan posisi terbaik global merupakan solusi terbaik dari keseluruhan solusi. Dimensi dinyatakan dengan x atau salah satu bagian dari solusi. Partikel merupakan nilai solusi dari tabel. Sedangkan satu swarm terdiri atas banyak partikel.

Pada DPSO ada dua jenis subjek yaitu. Subjek pertama yaitu subjek yang menentukan baris dan kolom pada tabel solusi X_i yang terdiri atas x_1, x_2, ... , x_n di dalam tabel. Subjek kedua yaitu subjek yang menentukan nilai solusi. Jika Subjek 1, subjek 2, dan subjek 3 tabel solusi dan subjek selanjutnya merupakan nilai dari x dengan masing masing jumlah elemen dari subjek sebanyak dua, maka akan terlihat pada tabel sebagai berikut.

Tabel 1. Solusi

id	subjek 1	subjek 2	subjek 3	X
1	1	1	1	x_1
2	1	1	2	x_2
3	1	2	1	x_3
4	1	2	2	x_4
5	2	1	1	x_5
6	2	1	2	x_6
7	2	2	1	x_7
8	2	2	2	x_8

Nilai dari x_i diikuti dengan kendala lemah seperti pada tabel berikut.

Tabel 2. Kendala

X	Kendala	Nilai kendala
x_1	jika $x < 40$	-40
x_1	jika $x > 1000$	-50

X_i merupakan partikel yang terdiri atas (x_1, x_2, ... , x_n) yang merupakan Urutan solusi dari jadwal kuliah yang berisi mata kuliah. P_i merupakan solusi terbaik dari swarm tertentu. P_i terdiri atas (x_1, x_2, ... , x_n). P_g merupakan solusi

terbaik dari semua pi Pi. P_g terdiri atas (x_1, x_2, ... , x_n). i dari x_i, merupakan id partikel swarm.

2.2.1 Permutasi

Permutasi ádalah penyusunan kembali kumpulan objek dalam urutan yang berbeda dari urutan semula. Permutasi dapat diilustrasikan pada relasi notasi. Pada DPSO, permutasi dilakukan dengan membandingkan partikel yang dipermutasikan, dengan partikel terbaik lokal dan partikel terbaik global. Ada dua langkah dalam melakukan permutasi, yaitu slide dan transposisi.

a. Slide.

   Slide merupakan proses pergeseran data dimana jumlah pergeseran satu data dengan data yang lain memiliki jumlah yang sama. Untuk *m-ordered sequence* $X = \{x_1, x_2, . . ., x_m\}$, slide operator beraksi pada X dijabarkan pada humus berikut

   $SL(x,k) = \{x_{(1+k)\%m}, X_{(2+k)\%m}, . . .,X_{(m+k)\%m}\}$.....(8)

   Proses slide dilakukan di awal proses pendekatan dimana penentuan dilakukannya proses slide berdasarkan pengecekan if-else.

   Sebagai contoh, transposisi dari X = {1,2,3,4,5} untuk mendekati data tujuan P_{best} = {2,5,4,3,1}. Jika dilakukan slide sebanyak 1 kali atau dapat ditulis slide (x,1) , maka hasilnya ialah [2,3,4,5,1]. Saat X = {1,2,3,4,5}, tidak ada posisi x_i yang sama yang memiliki nilai yang sama dengan P_{best}. Setelah dilakukan slide(x,1), dengan hasil X = {2,3,4,5,1}, pada posisi x_1, x_3 dan x_5 memiliki

nilai yang sama dengan P_{best} yang mengartikan bahwa hasil slide (x,1) memiliki hasil yang lebih baik.

b. Transposisi

Transposisi merupakan fungsi pertukaran posisi data. Sebagai contoh transposisi dari X = {1,2,3,4,5} untuk mendekati data P_{best} = {2,5,4,3,1}. Dilakukan transposisi $t(x_1,x_2)$ menjadi {2.1.3.4.5}. Dilakukan transposisi (x_2,x_5) sehingga menjadi {2,5,3,4,1}. Dilakukan transposisi (x_3,x_4) sehingga menjadi {2,5,4,3,1}. Disini akan terbentuk tiga langkah transposisi untuk proses pendekatan terhadap P_{best}.

Jumlah langkah pendekatan ini dilakukan sebanyak hasil kali nilai c dengan random. Misalkan saja c * random menghasilkan 40%. Langkah yang dilakukan sebanyak langkah transposisi * c * random. Misal 3 * 40%, memiliki hasil 1.2. disini langkah dipandang secara real, bukan pecahan. Maka nilai pecahan 1.2 akan menjadi nilai satu. hasil akhir ialah, dilakukan transposisi pendekatan sebanyak satu kali.

Penjelasan mengenai transposisi di atas hanya merupakan bagian dari proses DPSO. Pendekatan pada DPSO dilakukan secara tiga arah, sesuai dengan pendekatan sebelumnya ($wV_i(k)$), terhadap posisi terbaik lokal ($c_1r_1(P_i - x_i(k))$) dan posisi terbaik global ($c_2r_2(P_g - x_i(k))$. $V_i(k)$ merupakan pendekatan terhadap pengalaman individu dengan jumlah langkah pendekatan dipengaruhi oleh variabel w. $P_i - x_i(k)$ merupakan pendekatan terhadap posisi terbaik lokal

dengan jumlah langkah pendekatan dipengaruhi oleh variabel w Variabel c_1 dan variabel random r_1. $P_g - x_i(k)$ merupakan pendekatan terhadap posisi terbaik global dengan jumlah langkah pendekatan dipengaruhi oleh variabel c_2 dan variabel random r_1. Rata-rata implementasi menggunakan setting dengan c1 = c2. Penggunaan *inertia weight* 'w' lebih jarang dilakukan.*Inertia weight* memiliki nilai standar 0.4 sampai 0.9 (Kiruthiga dan Senthilkhumar, 2011).