Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 4, April 2017, hlm. 249-256 http://j-ptiik.ub.ac.id

Optimasi Penjadwalan Perkuliahan Dengan Menggunakan *Hybrid Discrete Particle Swarm Optimization* (Studi Kasus: PTIIK Universitas Brawijaya)

Muhammad Syafiq[1], Imam Cholissodin[2], Himawat Aryadita[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]id.muhammad.syafiq@gmail.com, [2]imamcs@ub.ac.id, [3]himawat@ub.ac.id

Abstrak

Penjadwalan pada umumnya dilakukan secara manual dengan menggunakan tabel konvensional atau *spreadsheet*. Akibatnya berdampak pada kualitas hasil penjadwalan dan dapat menguras waktu dan tenaga apabila jadwal yang dipertimbangkan mencapai ribuan. Berdasarkan permasalahan tersebut, dibutuhkan sistem cerdas yang tidak hanya mengotomasi prosesesnya, tetapi juga mengoptimasi hasilnya. PSO adalah metode optimasi yang terbukti efektif digunakan untuk memecahkan masalah optimasi multidimensi dan multi-parameter dibandingkan dengan metode yang lain. Algoritme DPSO digunakan pada penelitian ini dikarenakan permasalahan yang diangkat merupakan permasalahan kombinatorial. Berbagai strategi juga digunakan dalam penggunaan DPSO ini seperti *clustering* komposisi data pada partikel, penggunaan metode transposisi dalam perubahan posisi partikel, penggunaan *time-variant*, strategi pengacakan posisi partikel, strategi perbaikan posisi partikel serta penggunaan *multithreading*. Diharapkan dapat memberikan hasil penjadwalan dan waktu eksekusi yang optimal. Dengan berbagai macam strategi yang digunakan, penelitian ini akan menggunakan pendekatan *Hybrid Discrete Particle Swarm Optimization*. Hasil pengujian menunjukkan kombinasi parameter yang menghasilkan *fitness* terbaik adalah: $b_{loc}^{min} = 0{,}6$, $b_{loc}^{max} = 1$, $b_{glob}^{min} = 0{,}6$, $b_{glob}^{max} = 1$, $b_{rand}^{min} = 0$,$b_{rand}^{max} = 0{,}002$, jumlah partikel 2 dan jumlah iterasi 50.000. *Fitness* yang dihasilkan adalah 248.515,76 dengan waktu eksekusi 1 jam 46 menit 14 detik dan 600 milidetik.

Kata kunci: *penjadwalan, PSO, diskrit, kombinatorial, transposisi, time variant.*

Abstract

Generally, timetabling is done by using conventional tables or spreadsheets. As a result, its affect the quality of timetable and it could drain time and energy if data is considered in thousands. Based on these problems, it requires an intelligent system that not only automates its process but also optimizes the result. PSO is proved effective to solve multidimensional and multiparameter optimization problems compared with other methods. DPSO is used in this study because of combinatorics problems. Various strategies are also used in this method such as clustering particle representation, transposition method for particle movement, time-variant approach, guided random strategies, particle's position repair strategies, and multithreading. The strategy is expected to improve timetabling result. With the various strategies that have been used, this study will use "Hybrid Discrete Particle Swarm Optimization" approach. The test results showed the combination of parameters that resulting the best fitness is: $b_{loc}^{min} = 0.6$, $b_{loc}^{max} = 1$, $b_{glob}^{min} = 0.6$, $b_{glob}^{max} = 1$, $b_{rand}^{min} = 0$,$b_{rand}^{max} = 0.002$, *particle total 2 and iteration total 50,000. The resulting fitness is 248,515.76 with the total execution time is 1 hour 46 minutes 14 seconds and 600 milliseconds.*

Keywords: *timetabling, PSO, discrete, combinatorial, transposition, time variant.*

1. PENDAHULUAN

Penjadwalan berkaitan dengan alokasi sumber daya yang langka pada suatu kegiatan tertentu dengan tujuan untuk mengoptimalkan satu atau lebih ukuran kinerja, seperti meminimalisasi *makespan*, mengurangi jumlah pekerjaan yang terlambat, memampatkan pekerjaan dan lain sebagainya dikarenakan

kompleksitas masalah yang melekat dan banyaknya variabilitas (Jakobovi'c dan Budin, 2006). Masalah penjadwalan masih menjadi isu menarik yang diteliti dan dikembangkan oleh banyak peneliti (Yang dan Jat, 2011), salah satunya adalah penjadwalan perkuliahan. Permasalahan Penjadwalan Perkuliahan atau bisa disebut *University Course Timetabling Problem* (UCTP) merupakan suatu perencanaan untuk mengalokasikan sejumlah mata kuliah kedalam kumpulan periode (*timeslot*) dan ruangan tanpa melanggar aturan yang ada (Irene *et al.*, 2009). UCTP dianggap menjadi salah satu masalah yang sulit dihadapi oleh perguruan tinggi (A.Wasfy, 2007, Yang dan Jat, 2011).

Berbagai macam metode telah digunakan untuk menyelesaikan masalah penjadwalan seperti *Particle Swarm Optimization* (Chen dan Shih, 2013, Irene *et al.*, 2009, Kumar *et al.*, 2013, Sharma dan Singhal, 2014), dan *Genetic Algorithms* (Thanh, 2007, Wall, 1996, Yang dan Jat, 2011). *Particle Swarm Optimization* (PSO) adalah metode optimasi yang terbukti efektif digunakan untuk memecahkan masalah optimasi multidimensi dan multi-parameter (Wall, 1996). Ketika diterapkan pada berbagai masalah PSO menghasilkan kualitas yang lebih baik dan kinerja yang lebih cepat bila dibandingkan dengan algoritme genetika. PSO juga lebih baik dalam menangani konvergensi dibandingkan dengan algoritme genetika (Abdelhalim dan Habib, 2009, Jiang *et al.*, 2013).

Algoritme PSO menghasilkan solusi yang *powerful* bila diaplikasikan pada permasalahan dengan domain kontinyu, tetapi tidak pada domain diskrit, sehingga Kennedy dan Eberhart pada tahun 1997 mengusulkan optimasi algoritme PSO pada domain diskrit. Terciptalah metode baru yaitu *Discrete Particle Swarm Optimization* DPSO yang merupakan pengembangan dari algoritme PSO untuk domain diskrit. Beberapa pengembangan dari algoritme DPSO juga digunakan untuk menyelesaikan permasalahan pada penelitian ini seperti penggunaan *time-variant* (Ratnaweera *et al.*, 2004), *particle encoding* (Chen dan Shih, 2013), *guided random method*, serta *repair method*, harapannya dapat menyelesaikan permasalahan penjadwalan dengan lebih baik. Berdasarkan pengembangan tersebut dalam penelitian ini menggunakan pendekatan *Hybrid Discrete Particle Swarm Optimization* dalam menyelesaikan permasalahan penjadwalan perkuliahan.

2. METODE

2.1. *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) adalah teknik optimasi stokastik berbasis populasi yang dikembangkan oleh Dr. Eberhart dan Dr. Kennedy pada tahun 1995 terinspirasi oleh perilaku sosial berkelompok burung (Kennedy dan Eberhart, 1995). Algoritma dasar PSO terdiri dari tiga tahap, yaitu pembangkitan posisi dan kecepatan partikel, *update* kecepatan dan *update* posisi. Posisi partikel berubah setiap iterasinya berdasarkan variabel *update* kecepatan. Langkah Pertama, posisi x_k^i, dan kecepatan v_k^i dari kumpulan partikel dibangkitkan secara acak menggunakan batas atas (x_{max}) dan batas bawah (x_{min}) dari desain variable ditunjukkan pada Persamaan 1 dan 2.

$$x_0^i = x_{min} + rand(x_{max} - x_{min}) \quad (1)$$

$$v_0^i = x_{min} + rand(x_{max} - x_{min}) \quad (2)$$

Langkah kedua adalah melakukan *update* kecepatan terbaru (v_{k+1}) pada masing-masing partikel pada waktu $k+1$ berdasarkan kecepatan sebelumnya (v_k) dan kedua posisi terbaik yang telah dicari (p_{best} dan g_{best}). Perumusan *update velocity* mencakup beberapa parameter *random* (rnd), *inertia factor* (w), *self-confidence* (c_1), *swarm confidence* (c_2) ditunjukkan pada Persamaan 3 dan 4.

$$v_{k+1}^i = w * v_k^i + c1 * rnd(1) * (p_k^i - x_k^i) + c2 * rnd(1) * (p_{g,k} - x_k^i) \quad (3)$$

$$w = (w_{max} + w_{min}) + \frac{iter_{max}+iter}{iter_{max}} + w_{max} \quad (4)$$

Langkah ketiga adalah melakukan *update* posisi partikel (x_{k+1}^i) berdasarkan kecepatan yang dimilikinya (v_{k+1}^i). Diharapkan dengan berubahnya posisi partikel dapat menemukan solusi optimal. *Update* posisi partikel ditunjukkan pada Persamaan 5.

$$x_{k+1}^i = x_k^i + v_{k+1}^i \quad (5)$$

2.2. *Discrete Particle Swarm Optimization*

Tahun 1997 versi diskrit/biner dari algoritme PSO diusulkan oleh Kennedy dan Eberhart untuk masalah optimasi diskrit (Kennedy dan Eberhart, 1997). Setiap partikel terdiri dari D elemen yang menunjukkan solusi potensial. Setiap partikel dianggap sebagai posisi di ruang dimensi D dan setiap elemen dari posisi partikel dapat bernilai 0 atau 1. Persamaan yang digunakan untuk memperbarui kecepatan

dan posisi vektor dari masing-masing partikel DPSO ditunjukkan pada Persamaan 6, 7 dan 8.

$$V_i^{(t+1)}(j) = V_i^t(j) + c_1 r_1 \left(p_{best_i}{}^t(j) - X_i^t(j)\right) \\ c_2 r_2 \left(n_{best}{}^t(j) - X_i^t(j)\right) \quad (6)$$

$$X_i^{(t+1)}(j) = \begin{Bmatrix} 1 & sig\left(V_i^{(t+1)}(j)\right) > r_{ij} \\ 0 & lainnnya \end{Bmatrix} \quad (7)$$

dimana:

$$sig\left(V_i^{(t+1)}(j)\right) = \frac{1}{1+\exp\left(-V_i^{(t+1)}(j)\right)} \quad (8)$$

2.3. *Time Varying Particle Swarm Optimization*

Tahun 2004 Ratnaweera meneliti bahwa pada algoritme optimasi berbasis populasi, individu diharapkan bergerak untuk mencari posisi seluas-luasnya pada awal pencarian dan melakukan pencarian menuju global optimum pada akhir pencarian (Ratnaweera *et al.*, 2004). Berdasarkan penelitian itu, Ratnaweera mengenalkan pendekatan baru menggunakan kecepatan yang berubah secara konstan seiring bertambahnya iterasi (*time-variant*).

Pendekatan tersebut dilakukan dengan cara menurunkan pengaruh g_{best} dan meningkatkan pengaruh p_{best} dengan cara merubah komposisi parameter c_1 dan c_2 dari waktu ke waktu. Tingginya pengaruh p_{best} dan rendahnya pengaruh g_{best}pada awal pencarian membuat partikel bergerak secara bebas pada swarm daripada bergerak menuju global optimum. Sedangkan menurunkan pengaruh p_{best}dan meningkatkan pengaruh g_{best} pada akhir pencarian menyebabkan partikel bergerak menuju global optimum. Secara matematis perubahan nilai c_1 dan c_2 ditunjukkan pada Persamaan 9 dan 10. Penelitian tersebut menyatakan bahwa solusi yang dihasilkan dengan menggunakan *time-variant* lebih bagus daripada menggunakan PSO biasa maupun dengan menggunakan faktor inersia dan *constriction factor* (Ratnaweera *et al.*, 2004).

$$c_1 = \left(c_1^{min} - c_1^{max}\right)\left(\frac{t}{t^{max}}\right) + c_1^{max} \quad (9)$$

$$c_2 = \left(c_2^{max} - c_2^{min}\right)\left(\frac{t}{t^{max}}\right) + c_2^{min} \quad (10)$$

2.4. *Hybrid Discrete Particle Swarm Optimization*

2.4.1. Representasi Partikel

Partikel disusun atas urutan data pelajaran yang telah ditentukan sebelumnya. Data pelajaran merupakan data yang menyimpan informasi pelajaran selama penjadwalan tersebut akan dilaksanakan. Data ini memuat enam informasi yaitu mata kuliah, jumlah sks, jumlah perkuliahan selama seminggu, dosen, kelas, dan kemungkinan ruangan yang dapat ditempati. Berikut adalah contoh daftar pelajaran ditunjukkan pada Tabel 1.

Tabel 1. Data Pelajaran

No	Mata Kuliah	SKS	⋯	Kelas
1	Pemrograman Dasar	2	⋯	INF-1-A
2	Pemrograman Dasar	2	⋯	INF-1-B
3	Pemrograman Dasar	2	⋯	INF-1-C
⋮	⋮	⋮	⋱	⋮
1244	*Dummy*	1	⋯	Semua

Data *dummy* pada daftar pelajaran digunakan sebagai pelengkap data agar semua pelajaran dapat ditempatkan di seluruh kemungkinan ruangan yang ada. Strategi untuk memaksimalkan hasil *fitness*, masing-masing pelajaran akan dikelompokkan berdasarkan kemungkinan ruangan dari pelajaran tersebut sehingga apabila posisi partikel berubah, pelajaran tersebut akan berpindah posisi hanya dalam satu *cluster* sehingga tidak ada pelajaran yang ditempatkan di luar ruangan yang telah ditetapkan. Representasi partikel ditunjukkan pada Gambar 1.

1	3	4	5	6	7	⋯	995
2	18	23	42	53	67	⋯	1244

Gambar 1. Representasi Partikel

Gambar 1 menunjukkan pelajaran 1 dan 2 berada pada *cluster* yang berbeda. Hal ini dikarenakan kedua pelajaran tersebut memiliki kemungkinan penempatan di ruangan yang berbeda. Dengan demikian pelajaran 1 tidak dapat menempati ruangan yang dimiliki oleh pelajaran 2.

2.4.2. *Update* Posisi

Update posisi yang digunakan berdasarkan penelitian dari Hoffmann *et al.* (2011) yang mengadopsi penelitian Clerc (2000). Update posisi yang di kenalkan oleh Clerc sangat cocok bila diimplementasikan menggunakan representasi partikel pada penelitian ini tetapi masih terdapat banyak kendala sehingga Hoffmann memperbaiki update posisi tersebut. Berikut adalah update posisi yang telah

dimodifikasi oleh Hoffmann ditunjukkan pada Persamaan 11 sampai 14.

$$x_i^{t+1} = d_{glob} + \frac{1}{2}\left(d_{loc} - d_{glob}\right) + v_{rand} \quad (11)$$

dengan

$$d_{loc} = x_i^t + r_{loc} \cdot b_{loc}(p_i^t - x_i^t) \quad (12)$$

$$d_{glob} = x_i^t + r_{glob} \cdot b_{glob}\left(p_g - x_i^t\right) \quad (13)$$

$$v_{rand} = r_{rand} \cdot b_{rand}(p_{rand}^t - x_i^t) \quad (14)$$

2.4.3. Fungsi *Fitness*

Fungsi *fitness* dijadikan sebagai alat ukur untuk memilih objek terbaik dari sekumpulan objek yang ada (Tang dan Lee, 2006). Dalam algoritme evolusi fungsi *fitness* bertanggung jawab dalam menentukan solusi yang terbaik dari sekumpulan solusi yang ada (Nelson *et al*., 2009). Permasalahan penjadwalan merupakan permasalahan yang digunakan untuk meminimalisasi konflik waktu antara mata kuliah, kelas, ruangan serta dosen pengajar dan juga meminimalkan konflik dari aturan lain (*constraint*) yang telah ditentukan. Sehingga dalam hal ini semakin besar nilai yang dihasilkan maka semakin sedikit konflik yang terjadi dari solusi tersebut. Pada permasalahan terdapat 2 tipe constraint yaitu *hard constraint* dan *soft constraint*. *Hard Constraint* merupakan suatu aturan yang ditetapkan dan harus dipenuhi (Burke dan Petrovic, 2002). *Soft Constraint* merupakan suatu aturan yang ditetapkan tetapi sedapat mungkin harus dipenuhi (Burke dan Petrovic, 2002). Adapun fungsi *fitness* yang digunakan pada penelitian ini ditunjukkan pada Persamaan 15.

$$f(x) = \sum_{i=1}^{m}\left(\frac{k_{i1}}{1+c_{i1}} + \frac{k_{i2}}{1+c_{i2}} + \cdots + \frac{k_{in}}{1+c_{in}}\right) \quad (15)$$

Dimana; x adalah partikel yang akan dilakukan perhitungan nilai fitness, k_i adalah konstanta bobot constraint-i, c_i adalah nilai beban constraint-i, $1,2,\ldots,n$ adalah total constraint, m adalah total periode pada semua ruangan dan semua hari dan c merupakan kumpulan dari *constraint* pada penjadwalan baik *hard constraint* maupun *soft constraint*. Berikut adalah daftar *hard constraint* dan *soft constraint* pada permasalahan ini.

Hard Constraint:

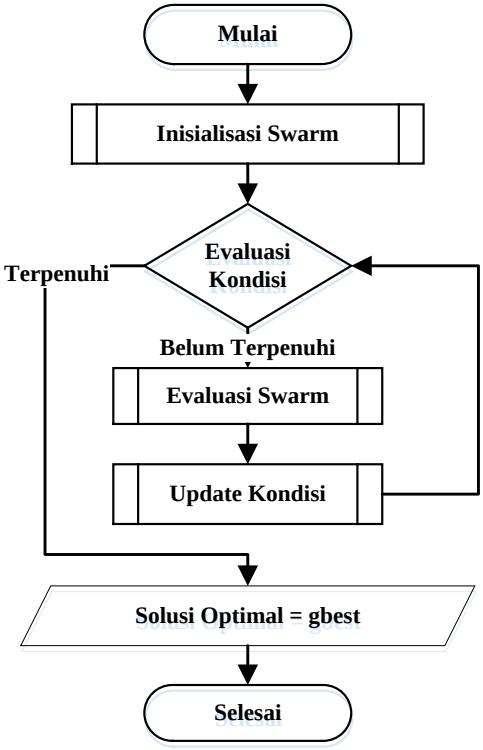
1. Tidak ada dosen yang mengajar di waktu yang sama pada hari yang sama. Meskipun ruangan yang digunakan berbeda.
2. Tidak ada kelas yang sama pada waktu yang sama meskipun di ruangan yang berbeda.
3. Tidak ada kelas yang perulangan mata kuliah melebihi satu yang mana penempatan mata kuliahnya berada pada hari yang sama.
4. Tidak ada pelajaran yang penempatan ruangannya tidak termasuk dalam kemungkinan ruangan yang telah ditentukan.

Soft Constraint:

1. Pelajaran akan melanggar *timeoff* ruangan apabila pelajaran tersebut menempati ruangan yang berstatus *not available*.
2. Pelajaran akan melanggar *timeoff* dosen apabila pelajaran tersebut diletakkan dimana dosen pengajar pelajaran tersebut memiliki status *not available*.
3. Pelajaran akan melanggar *timeoff* kelas apabila pelajaran tersebut diletakkan pada waktu kelas tersebut *not available*.
3. Pelajaran akan melanggar *timeoff* mata kuliah apabila pelajaran tersebut ditempatkan dimana mata kuliah pada pelajaran tersebut berstatus *not available*.

2.4.4. Alur HDPSO

Berikut adalah alur dari algoritme HDPSO ditunjukkan dalam diagram alir Gambar 2.



Gambar 2. *Flowchart* Metode HDPSO

1. Inisialisasi *Swarm*

Inisialisasi *swarm* digunakan untuk meng-inisialisasi seluruh partikel pada *swarm* dan menem-patkannya secara acak pada *swarm space*. Strategi pengacakan yang dengan

menempatakan data pelajaran yang memiliki SKS tinggi terlebih dahulu diikuti dengan SKS yang lebih sedikit dengan harapan seluruh pelajaran dapat ditempatkan. Pelajaran juga ditempatkan di ruangan yang memang dapat ditempati oleh pelajaran tersebut. Strategi ini dinamakan *guided random*.

2. Evaluasi Kondisi

Evaluasi Kondisi digunakan untuk mengevaluasi apakah kondisi yang digunakan telah memenuhi kriteria atau tidak. Apabila kondisi tersebut telah memenuhi maka proses akan berhenti dan menjadikan g_{best} terakhir sebagai solusi optimal Sedangkan bila kondisi yang didefinisikan masih belum memenuhi kriteria maka akan dilakukan lagi proses evaluasi *swarm*.

3. Evaluasi *Swarm*

Evaluasi *swarm* merupakan serangkaian prosedur bagaimana sebuah *swarm* dapat dievaluasi. Bagaimana masing-masing partikel akan mengevaluasi posisi terbaik selama ini (p_{best}). Bagaimana posisi terbaik pada *swarm* (g_{best}) dievaluasi. Bagaimana masing-masing partikel memperbarui posisinya berdasarkan gaya tarik yang mempengaruhi partikel tersebut. Bagaimana posisi terbaru dapat diperbaiki. Bagaimana setiap partikel mengevaluasi *fitness* terhadap posisinya yang terbaru. Adapun rangkaian prosedur yang dilakukan pada saat evaluasi *swarm* yaitu

   a. *Update* p_{best}.
   b. *Update* g_{best}.
   c. *Update* Posisi.
   d. Perbaikan Posisi.
   e. Perhitungan *Fitness*.

Perbaikan posisi dilakukan untuk menukar data pelajaran yang mengalami *overflow* dalam penempatannya maupun pelajaran yang ditempatkan pada ruangan yang bukan pada tempatnya

4. *Update* Kondisi

Pada proses update kondisi, kondisi akan diperbarui berdasarkan kondisi yang telah didefinisikan.

3. PENGUJIAN

3.1. Parameter Pengujian

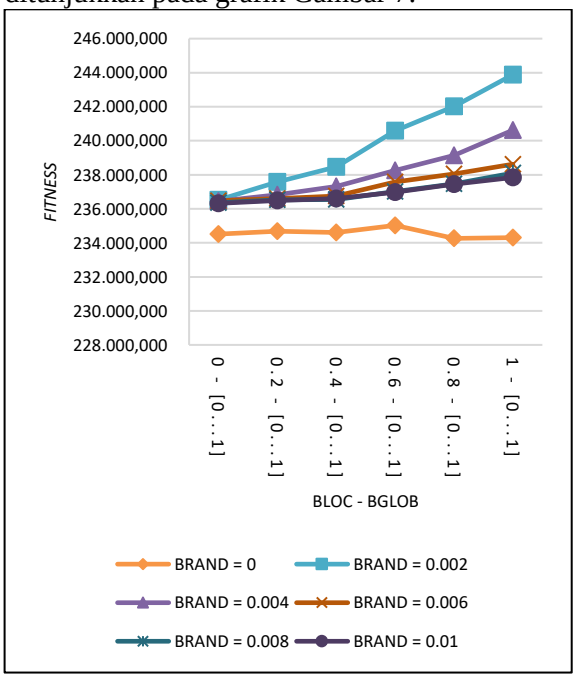
Pada penelitian ini sistem akan diimplementasikan menggunakan bahasa pemrograman java dengan *database* sqlite. Adapun data yang digunakan adalah data 5 hari aktif perkuliahan dengan 38 ruangan yang digunakan, masing-masing hari terdapat 17 periode waktu (*timeslot*), 109 dosen, 111 kelas, 70 mata kuliah, dan 3 jurusan masing-masing 4 angkatan. Berikut adalah jangkauan parameter HDPSO ditunjukkan pada Tabel 2.

Tabel 2. Parameter HDPSO

Parameter	Jangkauan
b_{loc}^{min}	b_{loc}^{max} – 0.4
b_{loc}^{max}	[0…1] dengan peningkatan 0.2
b_{glob}^{min}	b_{glob}^{max} – 0.4
b_{glob}^{max}	[0…1] dengan peningkatan 0.2
b_{rand}^{min}	b_{rand}^{max} – 0.002
b_{rand}^{max}	[0…0.01] dengan peningkatan 0.002
Jumlah iterasi	50 – 500.000
Jumlah Partikel	2 – 200
Jumlah percobaan	4 kali

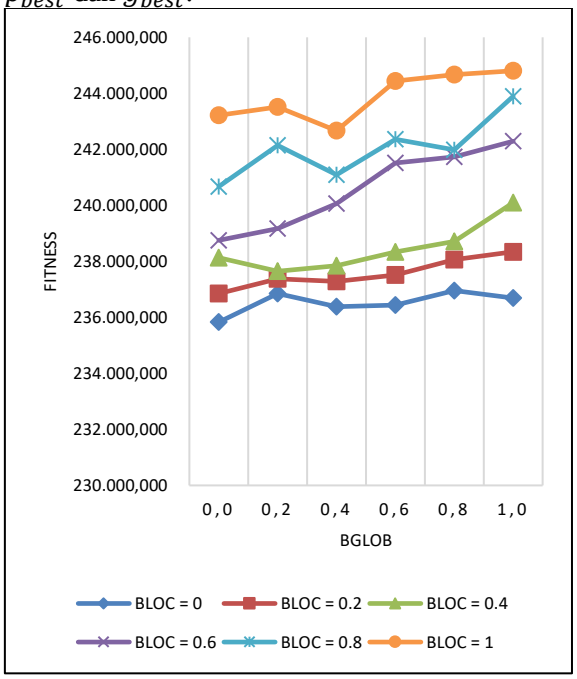
3.2. Pengujian

Pengujian yang akan dilakukan meliputi pengujian parameter, pengujian jumlah iterasi, jumlah partikel. Pengujian parameter digunakan untuk mencari komposisi parameter b_{rand}, b_{loc} dan b_{glob} yang tepat terhadap *fitness* yang dihasilkan. Pengujian ditunjukkan pada grafik Gambar 3 dan 4. Pengujian iterasi digunakan untuk memilih komposisi iterasi yang tepat, tidak menguras waktu eksekusi terlalu lama dan menghasilkan fitness yang maksimal. Pengujian tersebut ditunjukkan pada grafik Gambar 5 dan 6. Pengujian jumlah partikel digunakan untuk memilih komposisi partikel yang tepat dengan memberikan fitness yang terbaik tanpa menguras waktu eksekusi terlalu lama. Pengujian tersebut ditunjukkan pada grafik Gambar 7.

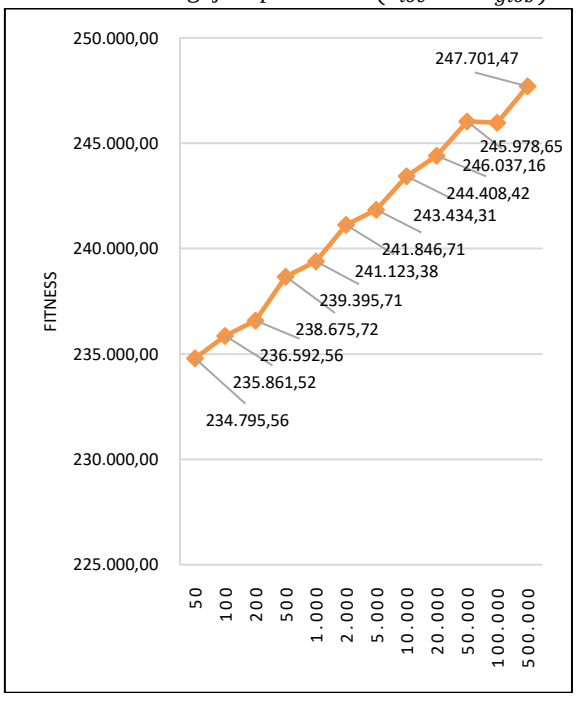


Gambar 3. Pengujian parameter (b_{rand})

Berdasarkan grafik yang ditunjukkan pada Gambar 3, *fitness* yang dihasilkan apabila $b_{rand} = 0$ rendah dibanding penggunaan lebih dari 0, Hal ini dikarenakan peran b_{rand} untuk menjaga keberagaman posisi partikel. *Fitness* terbaik dicapai ketika nilai $b_{rand} = 0{,}002$ dibanding penggunaan lebih dari itu dikarenakan keberagamannya masih mendekati posisi dari p_{best} dan g_{best}.



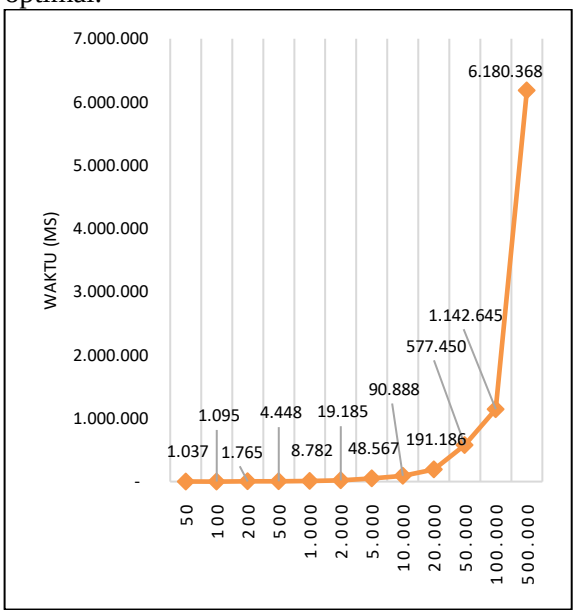
Gambar 4. Pengujian parameter $(b_{loc}$ dan $b_{glob})$



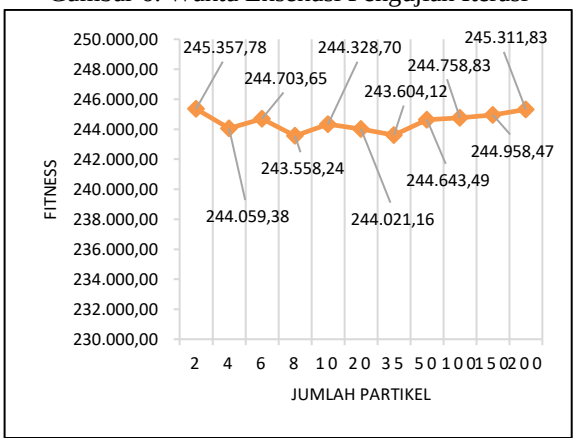
Gambar 5. Pengujian Iterasi

Gambar 4 data menunjukkan bahwa pemilihan b_{loc}^{max} dan b_{glob}^{max} terbaik ketika bernilai 1. Hal ini dikarenakan partikel akan secara sempurna bergerak secara bebas pada awal pencarian dan bergerak secara sempura menuju global optimum di akhir pencarian. Pemilihan yang tidak tepat menyebabkan partikel bergerak menuju satu arah saja sehingga memungkinkan terjadi konvergensi dini.

Berdasarkan grafik yang ditunjukkan pada Gambar 5, semakin tinggi iterasi yang digunakan semakin optimal solusi dari dihasilkan oleh sistem dengan ditandainya hasil *fitness* yang semakin meningkat. Hal ini disebabkan semakin banyaknya iterasi yang digunakan membuat partikel bergerak untuk menemukan solusi optimal semakin banyak sehingga memungkinkan partikel menemukan solusi yang optimal.



Gambar 6. Waktu Eksekusi Pengujian Iterasi



Gambar 7. Pengujian Jumlah Partikel

Namun semakin banyak iterasi yang digunakan berdampak pada waktu eksekusi dikarenakan sistem akan melakukan evaluasi masing-masing partikel lagi dan lagi yang mana

memerlukan waktu. Hal ini ditunjukkan pada Gambar 6. Peningkatan jumlah iterasi juga tidak menjamin meningkatkan nilai *fitness* secara signifikan. Hal ini ditunjukkan pada peningkatan iterasi dari 50.000 – 100.000. Berdasarkan analisa tersebut jumlah iterasi yang dipilih yang menghasilkan fitness yang mendekati optimal dan memiliki waktu ideal adalah 50.000.

Grafik yang ditunjukkan pada Gambar 7 menunjukkan pada permasalahan ini peningkatan jumlah partikel tidak mempengaruhi nilai *fitness* yang dihasilkan. Tidak meningkatnya nilai *fitness* yang dihasilkan dapat disebabkan oleh representasi partikel maupun evaluasi partikel seperti strategi pengacakan dan strategi perbaikan yang digunakan mampu menjelajahi seluruh *swarm space* yang ada, atau dimungkinkan *swarm space* yang ada pada permasalahan ini memiliki lingkup yang kecil. Berdasarkan analisa tersebut jumlah partikel yang dipilih yang menghasilkan fitness yang mendekati optimal dan memiliki waktu ideal adalah 2.

4. KESIMPULAN

Penelitian ini telah mengimplementasikan penggunaan HDPSO dalam menyelesaikan permasalahan penjadwalan perkuliahan. Hasil terbaik dicapai oleh sistem dengan *fitness* sebesar 248.515,76 membutuhkan waktu selama 1 jam 46 menit 14 detik dan 600 milidetik dengan komposisi parameter $b_{loc}^{min} = 0{,}6$, $b_{loc}^{max} = 1$, $b_{glob}^{min} = 0{,}6$, $b_{glob}^{max} = 1$, $b_{rand}^{min} = 0$,$b_{rand}^{max} = 0{,}002$, jumlah partikel 2 dan jumlah iterasi 50.000. Hasil tersebut masih terdapat beberapa constraint yang masih mengalami konflik. Hal ini dikarenakan ketidakmampuan partikel dalam menemukan posisi terbaik atau representasi partikel yang digunakan tidak mampu menggerakkan partikel dengan baik sehingga sulit untuk mencapai konvergensi.

Usulan dalam penelitian selanjutnya, peneliti menyarankan untuk menerapkan *time window* pada masing-masing objek yang dilakukan penjadwalan, memampatkan hasil penjadwalan, dan menyeimbangkan komposisi mata kuliah perharinya. Pada penggunaan metode butuh perbaikan pada penggunaan fungsi repair. Fungsi repair dapat dihibridasi menggunakan Local Search(Chen dan Shih, 2013, Yang dan Jat, 2011) atau Constraint Based Reasoning (Irene *et al.*, 2009). Merumuskan representasi partikel baru yang lebih sederhana dan lebih powerful sehingga partikel akan lebih mudah untuk mencapai tingkat konvergensi pada global optimum.

5. DAFTAR PUSTAKA

A.Wasfy, F.A. 2007. Solving the University Class Scheduling Problem Using Advanced ILP Techniques. *IEEE*.

Abdelhalim, M.B. and Habib, S.E.D. 2009. Particle Swarm Optimization for HWSW Partitioning

Burke, E.K. and Petrovic, S. 2002. Recent research directions in automated timetabling. *European Journal of Operational Research* 140(2) 266-280.

Chen, R.-M. and Shih, H.-F. 2013. Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. *Algorithms* 6(2) 227.

Clerc, M. 2000. Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem.

Hoffmann, M., Mühlenthaler, M., Helwig, S. and Wank, R. 2011. Discrete Particle Swarm Optimization for TSP: Theoretical Results and Experimental Evaluations. *In* A. Bouchachia ed. *Adaptive and Intelligent Systems: Second International Conference, ICAIS 2011, Klagenfurt, Austria, September 6-8, 2011. Proceedings*. Berlin, Heidelberg, Springer Berlin Heidelberg. pp. 416-427.

Irene, H.S.F., Deris, S. and Hashim, S.Z.M. 2009. University course timetable planning using hybrid particle swarm optimization. *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. Shanghai, China, ACM. 239-246.

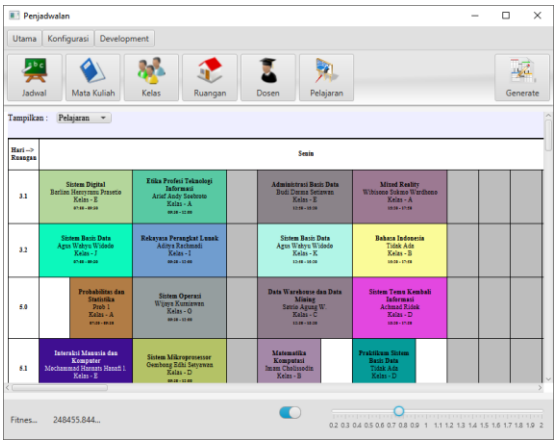
Jakobovi'c, D. and Budin, L. 2006. Dynamic Scheduling with Genetic Programming. *Proceedings of the 9th European Conference on Genetic Programming* 3905 73--84.

Jiang, M. et al. 2013. Study on Parameter Optimization for Support Vector Regression in Solving the Inverse ECG Problem. *Computational and Mathematical Methods in Medicine* 2013 9.

Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization. Neural Networks, 1995. Proceedings., IEEE International

Conference on.
Kennedy, J. and Eberhart, R.C. 1997. A discrete binary version of the particle swarm algorithm.
Kumar, A., Singh, K. and Sharma, N. 2013. Automated Timetable Generator Using Particle Swarm Optimization. *International Journal on Recent and Innovation Trends in Computing and Communication* 1(9) 6.
Nelson, A.L., Barlow, G.J. and Doitsidis, L. 2009. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems* 57(4) 345-370.
Ratnaweera, A., Halgamuge, S.K. and Watson, H.C. 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation* 8(3) 240-255.
Sharma, J. and Singhal, R.S. 2014. Genetic Algorithm and Hybrid Genetic Algorithm for Space Allocation Problems-A Review
Tang, P. and Lee, G.K. 2006. An Adaptive Fitness Function for Evolutionary Algorithms Using Heuristics and Prediction. 2006 World Automation Congress.
Thanh, N.D. 2007. Solving Timetabling Problem Using Genetic and Heuristic Algorithms. Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007).
Wall, M.B. 1996. A Genetic Algorithm for Resource-Constrained Scheduling.
Yang, S. and Jat, S.N. 2011. Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41(1) 93-106.

6. LAMPIRAN



Gambar 8. Implementasi Sistem