Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 11, November 2019, hlm. 10836-10841 http://j-ptiik.ub.ac.id

Optimasi Penjadwalan Perkuliahan menggunakan *Hybrid Discrete Particle Swarm Optimization* (Studi Kasus: STAI Ma'had Aly Al-Hikam Malang)

Achmad Choirur Roziqin[1], Imam Cholissodin[2], Bayu Rahayudi[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]achmadchoirurroziqin@gmail.com, [2]imamcs@ub.ac.id, [3]ubay1@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 permasalah tersebut, dibutuhkan sistem cerdas yang tidak hanya mengotomasi prosesnya, tetapi juga mengoptimasi hasilnya. *Particle Swarm Optimization* (PSO) adalah algoritme metaheuristik yang populer digunakan untuk memecahkan masalah optimasi multiparameter. Algoritme *Discrete* PSO digunakan pada penelitian ini dikarenakan permasalahan yang diangkat merupakan permasalahan kombinatorial. Berbagai strategi juga digunakan dalam metode ini, seperti komposisi data pada partikel, penggunaan metode transposisi dalam perubahan posisi partikel, strategi pengacakan posisi partikel, dan strategi perbaikan posisi partikel. Diharapkan dapat memberikan hasil penjadwalan dan waktu eksekusi yang optimal. Dengan berbagai macam strategi yang digunakan, penelitian ini akan menggunakan pendekatan *Hybrid Discrete* PSO. Hasil pengujian menunjukkan kombinasi parameter yang menghasilkan *fitness* terbaik adalah b_loc=1, b_glob=0,8, b_rand=0, jumlah partikel 200 dan jumlah iterasi 40. *Fitness* yang dihasilkan adalah 0,018896357 dengan waktu eksekusi 34 menit 16 detik 358 milidetik.

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

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. Particle Swarm Optimization (PSO) is a popular metaheuristic algorithm to solve multiparameter optimization problems. Discrete PSO is used in this study because of combinatorics problems. Various strategies are also used in this method such as transposition method for particle movement, guided random strategies, and particle's position repair strategies. The strategies is expected to improve timetabling result. With the various strategies that have been used, this study will use "Hybrid Discrete PSO" approach. The test results showed the combination of parameters that resulting the best fitness are b_loc=1, b_glob=0,8, b_rand=0, number of particle is 200 and number of iteration is 40. The resulting fitness is 0,018896357 with the total execution time is 34 minutes 16 seconds 358 miliseconds.

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

1. PENDAHULUAN

Saat ini, tidak sedikit proses penjadwalan yang masih dilakukan dengan cara konvensial, baik dengan aplikasi *spreedsheet*, seperti Microsoft Excel, maupun tanpa menggunakan bantuan aplikasi. Hal tersebut mengakibatkan banyaknya waktu yang dihabiskan hanya untuk melakukan proses penjadwalan, sedangkan tidak sedikit instansi atau lembaga pendidikan yang memiliki banyak tugas lain yang menyusul. Salah satunya adalah Sekolah Tinggi Agama Islam Ma'had Aly (STAIMA) Al-Hikam Malang.

Dari hasil wawancara, STAIMA saat ini masih menggunakan Microsoft Excel sebagai alat penjadwalannya. Adanya berbagai macam aturan yang ditetapkan, menyebabkan proses

penjadwalan semakin rumit dan membutuhkan waktu hingga satu minggu. Waktu akan semakin terkuras jika tiba-tiba ada kendala mendadak. Banyaknya data yang diterapkan tidak dimungkinkan untuk melakukan pencarian jadwal yang optimal secara *brute force*. Maka dari itu dibutuhkanlah sebuah sistem yang menerapkan optimasi pencarian untuk mempermudah proses penjadwalan yang kompleks, sehingga proses penjadwalan bisa diselesaikan dengan waktu yang lebih singkat Algoritme yang diterapkan dalam penelitian ini adalah *Hybrid Discrete Particle Swarm Optimization* (HDPSO).

Ada beberapa penelitian sebelumnya yang juga telah menggunakan Algoritme HDPSO untuk optimasi penjadwalan. (Tassopoulos & Beligiannis, 2012) dari Agrinio, Yunani menerapkan Algoritme HDPSO untuk memecahkan sebuah masalah penjadwalan kuliah beberapa sekolah menengah di Yunani. Penelitian tersebut menunjukkan bahwa algoritme yang berbasis PSO mencapai hasil yang lebih baik daripada algoritme evolusi dan *simmulated annealing* dengan masalah yang sama. Ada juga penelitian (Shiau, 2011) dari Kaohsiung, Taiwan yang menerapkan HDPSO. Penelitian tersebut menunjukkan bahwa HDPSO menghasilkan sebuah solusi yang efisien dengan secara optimal berhasil memenuhi batasan yang diberikan. Sekaligus menunjukkan bahwa HDPSO bisa mengungguli Algoritme Genetika. Berdasarkan uraian di atas, maka penulis mengambil judul "Optimasi Penjadwalan Perkuliahan *menggunakan Hybrid Discrete Particle Swarm Optimization* (Studi Kasus: Sekolah Tinggi Agama Islam Ma'had Aly Al-Hikam Malang)".

2. METODE

2.1 *Particle Swarm Optimization*

Pada tahun 1995, Kennedy dan Eberhart *Particle Swarm Optimization* (PSO). PSO ini adalah didasarkan dari perilaku sekelompok burung. Seekor burung dalam kawanannya direpresentasikan sebagai sebuah partikel, dan *swarm* terdiri dari sekelompok partikel. Ada empat tahapan utama yang ada pada algoritme dasar PSO, yaitu inisialisasi, *update* kecepatan, update posisi, dan update pembelajaran partikel.

1. Inisialisasi

   Ada 4 variabel yang dibangkitkan pada tahap insialisasi. Pertama adalah parameter yang terdiri dari ukuran populasi (*popSize*), panjang dimensi (d), posisi minimum (x_{min}), posisi maksimum (x_{max}), kecepatan minimum (v_{min}), kecepatan maksimum (v_{max}), koefisien akselerasi (c1 dan c2), bobot inersia (w), dan pengontrol pembelajaran (r1 dan r2). Kedua adalah kecepatan awal yang bernilai 0. Ketiga adalah posisi awal yang formulasinya ditunjukkan pada persamaan 1.

   $x_{i,j}^{0} = x_{min} + rand[0,1] * (x_{max} - x_{min})$(1)

   Keempat adalah p_{best} yang merupakan kumpulan partikel yang saat mereka berada di posisi terbaik mereka masing-masing dan g_{best} yang merupakan salah satu partikel pbest yang berada di posisi terbaik daripada partikel pbest yang lainnya.
2. Hitung *Fitness*

   $$f(x) = \frac{1}{1+\sum_{i=1}^{m} c_i} \quad (2)$$
3. Update Kecepatan

   $$v_{i,j}^{t+1} = w \cdot v_{i,j}^{t} + c_1 \cdot r_1\left(Pbest_{i,j}^{t} - x_{i,j}^{t}\right) + c_2 \cdot r_2\left(Gbest_{g,j}^{t} - x_{i,j}^{t}\right) \quad (3)$$
4. Update Posisi

   $$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \quad (4)$$
5. Update p_{best} dan g_{best}

   Nilai terbesar adalah nilai yang diambil sebagai nilai p_{best}. Sedangkan, Nilai g_{best} dapat dicari dengan membandingkan hasil nilai p_{best} dari setiap partikel yang ada pada iterasi sekarang. Nilai terbesar adalah nilai yang diambil sebagai nilai g_{best}.

2.2 *Hybrid Discrete Particle Swarm Optimization*

Hybrid Discrete Particle Swarm Optimization (HDPSO) adalah algoritme hasil modifikasi dari Algoritme PSO oleh Maurice Clerc pada tahun 2000. PSO dimodifikasi menjadi HDPSO agar dapat solusi yang didapat lebih optimal ketika mengatasi masalah kombinatorial yang bersifat diskrit.

Ada 8 struktur hasil modifikasi Maurice Clerc untuk Algoritme HDPS, yaitu:

- *Position*
  Sama dengan *position* pada PSO
- *Transposition*
  Teknik menukar nilai posisi pada dimensi tertentu.
- *Velocity*
  Penyebab dari terjadinya *transposition*.

- *Opposite of Velocity*
  Velocity yang prosesnya terbalik dari velocity asalnya.
- *Move*
  Cara mengganti *position* karena dipengaruhi oleh *velocity*.
- *Difference*
  Jarak antara dua *position* yang dikonversi menjadi *velocity*.
- *Addition*
  Penjumlahan dua *velocity* untuk mendapatkan *velocity* baru.
- *Multiplication*
  Proses yang diawali dengan mencari panjang *velocity* baru terlebih dahulu. Setelah itu *velocity* baru diisi sesuai dengan *velocity* sebelumnya namun dengan panjang yang berbeda.

Kedelapan struktur diatas digunakan sebagai landasan pada saat melakukan *update* posisi formulasinya ditunjukkan pada persamaan 5.

$$x_i^{t+1} = d_{glob} + \frac{1}{2}\left(d_{loc} - d_{glob}\right) + v_{rand} \quad (5)$$

dimana:

$$d_{glob} = x_i^t + r_{glob} \cdot b_{glob} \cdot (Gbest_g^t - x_i^t) \quad (6)$$

$$d_{loc} = x_i^t + r_{loc} \cdot b_{loc} \cdot (Pbest_i^t - x_i^t) \quad (7)$$

$$v_{rand} = r_{rand} \cdot b_{rand} \cdot (P_{rand} - x_i^t) \quad (8)$$

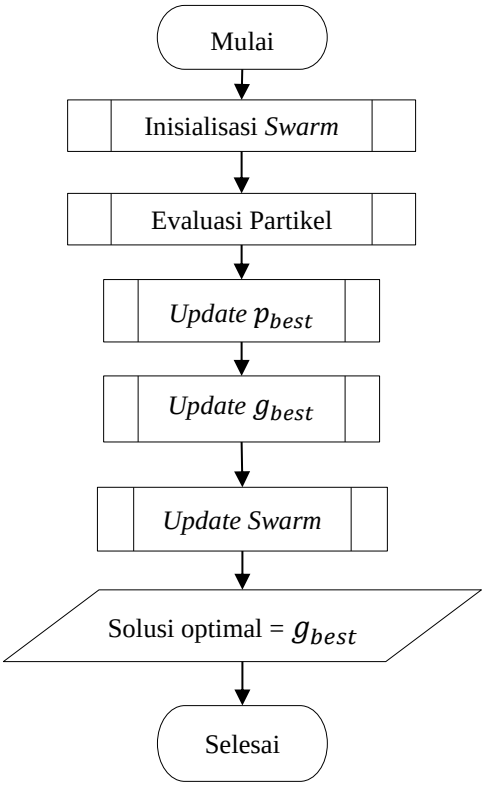
3. PEMBAHASAN

Dalam penelitian ini, algoritme yang digunakan adalah HDPSO. Sebelum masuk ke algoritme HDPSO, terlebih dahulu tentukan metode representasi partikel yang disusul antar urutan data pelajaran. Informasi mengenai pelajaran-pelajaran yang dilaksanakan selama perkuliahan disimpan dalam data pelajaran. Ada lima informasi pada data ini, yaitu nama kelas, nama dosen, jumlah perkuliahan selama seminggu, jumlah SKS, dan nama mata kuliah. Di STAIMA, pada semester ganjil tahun ajaran 2018/2019 ada 68 pelajaran. Tabel 1 menunjukkan sampel data pelajaran yang digunakan.

Gambar 1 menunjukkan alur kerja HDPSO dalam bentuk diagram alir.

Tabel 1. Sampel Data Pelajaran

No	Mata Kuliah	SKS	...	Kelas
1	Pancasila	2	...	PAI-1-A
2	Pancasila	2	...	PAI-1-B
⋮				
68	Takhrijul Hadits	2	...	PAI-3-B



Gambar 1. Alur Kerja HDPSO

1. Inisialisasi *Swarm*

   Inisialisasi *swarm* membutuhkan parameter ukuran *swarm*, yaitu total partikel yang ada di dalam swarm. Parameter tersebut telah didefinisikan sebelumnya. Selain itu juga posisi, sebagai representasi dari total data yang dipakai (data pelajaran ditambah dengan data dummy). Langkah pertama adalah mengecek total partikel yang ada dalam *swarm*. Jika belum sesuai dengan ukuran populasi, maka bangkitkan partikel baru. Kemudian partikel baru tersebut acak posisinya. Selanjutnya masukkan partikel tersebut ke dalam *swarm*. Lakukan proses tersebut terus menerus hingga total partikel yang ada dalam *swarm*. Jika sudah sesuai, maka *swarm* itulah populasi awalnya.
2. Evaluasi Partikel

   Sebelum memulai mengimplementasi-kan fungsi *fitness* yang telah ditentukan, terlebih dahulu lakukan perbaikan posisi agar lembar kerja yang ada dapat terpenuhi dengan setiap pelajaran. Perbaikan posisi dilakukan karena tiap pelajaran memiliki jam pelajaran yang berbeda-beda. Lalu hitung konflik tiap constraint terhadap

masing-masing partikel. Setelah semua partikel dihitung *constraint*-nya, kemudian terapkan fungsi *fitness*.

Daftar dari *hard* dan *soft constraint* yang diterapakan dalam penelitian ini antara lain :

Hard Constraint

a. Tidak ada dosen yang memberikan perkuliahan di periode yang sama meskipun hari dan ruangan yang digunakan berbeda. Hal tersebut dikarenakan tidak akan mungkin bisa dilakukan suatu pembelajaran apabila terdapat dosen yang sama yang mengajar pada waktu yang sama.

b. Tidak ada kelas yang diajar di periode yang sama meskipun hari dan ruangan yang digunakan berbeda. Hal tersebut dikarenakan tidak akan mungkin bisa dilakukan suatu pembelajaran apabila terdapat kelas yang sama yang diajar pada waktu yang sama.

c. Tidak ada perkuliahan yang jumlah SKS-nya lebih dari 3 yang berada pada hari yang berbeda meskipun periode dan ruangan sama. Hal tersebut karena diharuskannya dosen mengajarkan mata kuliahnya di hari yang berbeda.

d. Tidak ada pelajaran yang melanggar *timeoff* ruangan, yaitu pelajaran yang menempati ruangan yang berstatus tak tersedia. Hal tersebut karena ruangan yang berstatus tak tersedia adalah jam ishoma (istirahat, sholat, makan) dan ruangan yang digunakan untuk mahasiswa non mukim.

3. *Update* p_{best}

Saat evaluasi partikel selesai diproses, baik evaluasi partikel sebelum evaluasi *swarm* maupun yang ada di dalam evaluasi *swarm*, selanjutnya lakukan *update* p_{best}. Partikel dengan nilai *fitness* tertinggi hasil perbandingan antara nilai *fitness* pada partikel sebelumnya dengan saat ini dimasukkan ke dalam update p_{best}.

4. *Update* g_{best}

Setelah proses *update* p_{best} selesai dilakukan, selanjutnya lakukan *update* g_{best}. Lakukan *update* g_{best} dengan cara memilih partikel dari p_{best} yang nilai *fitness*-nya paling besar daripada nilai *fitness* dari partikel p_{best} lainnya.. Nilai g_{best} yang dijadikan sebagai solusi paling optimal jika iterasi berakhir.

5. *Update Swarm*

Cara memperbarui swarm adalah mengevaluasi *swarm* itu sendiri. Dalam evaluasi *swarm*, posisi tiap partikel harus di-*update*. Rumus *update* posisi ditunjukkan pada persamaan 5

4. PENGUJIAN

4.1. Parameter Pengujian

Pada penelitian ini, sistem akan diimple-mentasikan menggunakan bahasa pemrograman python dengan masukan berupa *file* excel. Adapun data yang digunakan adalah 5 hari aktif perkuliahan dengan 11 ruangan yang digunakan, masing-masing terdapat 12 periode waktu, 27 dosen, 8 kelas, 68 mata kuliah, dan 3 program studi. Berikut adalah jangkauan parameter HDPSO ditunjukkan pada Tabel 2.

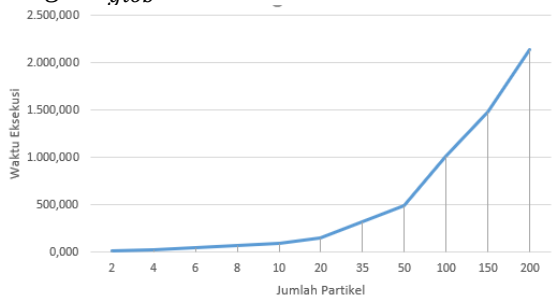
Tabel 2. Parameter HDPSO

Parameter	Jangkauan
b_{loc}	[0...1] dengan peningkatan 0.2
b_{glob}	[0...1] dengan peningkatan 0.2
b_{rand}	[0...1] dengan peningkatan 0.2
limit	100
popSize	200

4.2. Pengujian

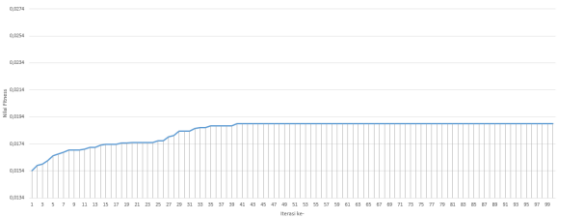
Pengujian yang dilakukan dalam penelitian ini antara lain pengujian parameter, pengujian jumlah partikel, dan pengujian konvergensi. Pengujian parameter digunakan untuk mencari komposisi parameter b_{loc}, b_{glob}, dan b_{rand} yang tepat terhadap *fitness* yang dihasilkan. Pengujian jumlah partikel bertujuan untuk menemukan jumlah partikel yang cocok dan waktu yang dibutuhkan sistem pada saat solusi optimal ditemukan. Pengujian jumlah iterasi dapat dilihat pada Gambar 4. Pengujian konvergensi dilakukan untuk menemukan jumlah iterasi yang tepat agar mendapatkan solusi yang optimal dalam waktu yang lebih cepat. Sistem akan diuji sebanyak 10 kali dengan jumlah iterasi maksimal adalah 100.

Hasil dari pengujian ini menunjukkan bahwa nilai *fitness* tertinggi adalah 0,0169491525423728 dimana nilai parameter b_{loc}, b_{glob}, dan b_{rand} masing-masing adalah 1, 0.8, dan 0. Hal ini menunjukkan bahwa penggunaan parameter b_{loc}, dan b_{rand} mempengaruhi kualitas dari nilai fitness yang dihasilkan dan kedua parameter tersebut berbanding terbalik pengaruhnya. Sedangkan parameter b_{glob} tidak terlalu berpengaruh terhadap nilai fitness meskipun dapat meningkatkan nilai fitness tetapi tidak signifikan. Memberikan nilai parameter b_{loc} sebesar 1 menyebabkan partikel bergerak bebas secara maksimal tetapi juga dipengaruhi social component yang ditandai dengan b_{glob} sebesar 0.8.



Gambar 4. Grafik Pengujian Jumlah Iterasi

Berdasarkan grafik yang dituntukkan pada Gambar 4, menunjukkan bahwa peningkatan jumlah partikel berpengaruh terhadap nilai *fitness* yang dihasilkan walaupun ada beberapa yang saat diberikan jumlah partikel yang lebih besar, nilai *fitness*-nya turun, seperti pada saat jumlah partikelnya 8, 50, dan 100. Hal ini menunjukkan bahwa ada peningkatan seiring dengan banyaknya jumlah partikel, walaupun peningkatannya nilai *fitness*-nya sangat kecil. Meningkatnya nilai *fitness* yang dihasilkan dipengaruhi oleh representasi partikel atau evaluasi partikel yang digunakan bisa mengarungi setiap titik pada ruang pencarian yang ada.



Gambar 5. Grafik Pengujian Konvergensi

Berdasarkan hasil pengujian konvergensi yang ditunjukkan pada Gambar 5, seluruh partikel sudah mengalami konvergensi dimulai pada saat iterasi ke-40 dengan rata-rata *fitness* adalah 0,018896357. Hal tersebut menunjukkan bahwa partikel telah menemukan solusi optimalnya pada iterasi ke-40 dalam waktu kurang dari 34 menit 16 detik 358 milidetik.

4.3. Hasil Analisis

Seluruh *constraint* yang digunakan berpengaruh terhadap hasil yang dihasilkan. Hasil penjadwalan dianggap berhasil jika nilai *fitness* yang dihasilkan menunjukkan bahwa tidak ada satupun *hard constraint* yang dilanggar, yaitu 1. Sistem ini masih belum bisa diterapkan karena rendahnya nilai fitness yang dihasilkan. Hal tersebut dikarenakan :

1. Dalam sistem ini masih banyaknya data pelajaran yang masuk ke ruangan yang berstatus tak tersedia, sehingga mengakibatkan banyaknya konflik yang terjadi.
2. Sistem ini menggunakan teknik pengacakan saat update posisi, tidak adanya pertukaran antara data pelajaran yang konflik dengan data dummy yang tidak konflik, sehingga data pelajaran yang konflik ada kemungkinan akan terjadi konflik lagi pada iterasi selanjunya.
3. Dalam sistem ini, fungsi fitness yang diterapkan menyetarakan soft constraint dengan hard constraint, sehingga nilai fitness cenderung kecil walaupun jika hanya soft constraint yang konflik.

5. KESIMPULAN

Berdasarkan dari perancangan, implementasi, serta pengujian yang telah dilakukan dalam penelitian ini, didapatkan beberapa kesimpulan sebagai berikut:

1. Implementasi metode HDPSO belum dapat diterapkan pada permasalahan penjadwalan perkuliahan STAIMA Al-Hikam Malang semester ganjil tahun ajaran 2018/2019.
2. Parameter yang ideal dan sebanding dengan nilai *fitness* yang dihasilkan terhadap waktu eksekusi yang dibutuhkan adalah sebagai berikut:
   a. b_{loc} = 1
   b. b_{glob} = 0.8
   c. b_{rand} = 0
   d. Jumlah partikel = 200
   e. Jumlah iterasi = 40

Hasil terbaik dicapai oleh sistem dengan *fitness* sebesar 0,018896357 membutuhkan waktu kurang dari 34 menit 16 detik 358 milidetik. 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 menemukan solusi yang lebih optimal.

6. DAFTAR PUSTAKA

Chen, Ruey Maw, dan Hsiao Fang Shih. 2013. "*Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search*." *Algorithms* VI: 227-244.

Hoffmann, Matthias, Moritz Mühlenthaler, Sabine Helwig, dan Rolf Wanka. 2011. "*Discrete Particle Swarm Optimization for TSP: Theoretical Results and Experimental Evaluations.*" *Adaptive and Intelligent Systems* 416-427.

Irene, Ho Sheau Fen, Deris Safaai, Mohd Hashim, dan Siti Zaiton. 2009. "*Solving University Course Timetable Problem Using Hybrid Particle Swarm Optimization.*" *World Summit on Genetic and Evolutionary Computation* 239-245.

Jha, Sujit Kumar. 2014. "*Exam Timetabling Problem Using Genetic Algorithm*." *International Journal of Research in Engineering and Technology* III (5): 649-654.

KBBI, Badan Bahasa Kemendikbud. 2016. KBBI Daring. Diakses February 10, 2019. https://kbbi.kemdikbud.go.id.

Mansur, Toni Prahasto, dan Farikhin. 2014. "*Particle Swarm Optimization* Untuk Sistem Informasi Penjadwalan *Resource* Di Perguruan Tinggi." JSINBIS 11-19.

Montero, Elizabeth, Cristina Maria Riff, dan Leopoldo Altamirano. 2011. "A *Particle Swarm Optimization*." *International Conference on Swarm Intelligence* (ICSI) 24-1 24-8.

Nurcholiq, Mochamad, wawancara oleh Achmad Choirur Roziqin. 2019. Wawancara (9 February).

Perzina, Radomir, dan Jaroslav Ramik. 2013. "*Self-Learning Genetic Algorithm for a Timetabling Problem with Fuzzy Constraints*." *International Journal of Innovative Computing Information and Control* IX (11): 4565-4582.

Qoiriah, Anita. 2014. "Penjadwalan Ujian Akhir Semester dengan Algoritma Genetika (Studi Kasus: Teknik Informatika UNESA)." Jurnal Manajemen Informatika III (2): 33-38.

Shiau, Der Fang. 2011. "*A Hybrid Particle Swarm Optimization for A University Course Scheduling Problem with Flexible Preferences*." *Expert Systems with Applications* XXXVIII (1): 235-248.

Syafiq, Muhammad, Imam Cholissodin, dan Himawat Aryadita. 2017. "Optimasi Penjadwalan Perkuliahan dengan Menggunakan *Hybrid Discrete Particle Swarm Optimization* (Studi Kasus: PTIIK Universitas Brawijaya)." Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer I (4): 249-256.

Tassopoulos, Ioannis X, dan Grigorios N Beligiannis. 2012. "*A Hybrid Particle Swarm Optimization Based Algorithm For High School Timetabling Problems*." *Applied Soft Computing* (12): 3472-3489.

Undang-undang Republik Indonesia Nomor 14 Tahun 2005 tentang Guru dan Dosen. 2005.