OPTIMASI PENJADWALAN CERDAS MENGGUNAKAN ALGORITMA *BEES* DAN *HARMONY SEARCH*

Muhamad Sauky[1], Galih Hermawan, S.Kom., M.T.[2]

[1,2] Universitas Komputer Indonesia
Jl. Dipatiukur No. 112-116, Bandung 40132
E-mail : ukkyms@gmail.com[1], galih.hermawan@ gmail.com[2]

ABSTRAK

Penjadwalan dapat didefinisikan sebagai proses untuk menempatkan *resources* atau *events* ke dalam *timeslots* sesuai dengan *constraints* yang sudah ditetapkan. Masalah ini membutuhkan waktu komputasi yang cukup tinggi untuk pencarian solusinya, terlebih lagi jika ukuran permasalahan semakin besar dengan bertambahnya jumlah komponen dan syarat yang ditentukan. Untuk itu algoritma *Bees* dan *Harmony Search* diterapkan pada penelitian ini, karena algoritma tersebut dapat diimplementasikan kedalam optimasi penjadwalan.

Adapun tahapan yang dianalisis adalah proses inisialisasi solusi awal, pencarian lokal dan global pada algoritma *Bees*, inisialisasi *Harmony Memory*, improvisasi *Harmony*, dan pembandingan *Harmony* pada algoritma *Harmony Search*. Berdasarkan hasil penelitian, algoritma *Bees* dan *Harmony Search* dapat digunakan untuk menyelesaikan masalah penjadwalan. Berdasarkan hasil pengujian, parameter *Bees* dengan nilai n=20, m=5, e=4, nep=4, nsp=2, ngh=5 dan parameter *Harmony Search* dengan nilai HMCR=0.8, PAR=0.3 akan mengoptimalkan proses pencarian. Berdasarkan hasil pengujian *soft constraints* yang sudah dilakukan, semakin banyak jumlah slot waktu tidak bersedia yang ditetapkan maka semakin banyak pula jumlah rata-rata pelanggaran *soft constraints* yang terjadi. Dapat diketahui pula bahwa batas aman jumlah slot waktu tidak bersedia adalah antara 30 sampai dengan 60 slot waktu dan jumlah slot waktu tidak bersedia tidak terlalu berpengaruh terhadap rata-rata waktu yang digunakan untuk menemukan solusi.

Kata kunci : Penjadwalan, *Bees*, *Harmony Search*.

1. PENDAHULUAN

Penjadwalan dapat didefinisikan sebagai proses untuk menempatkan *resources* atau *events* ke dalam *timeslots* yang terbatas sesuai dengan *constraints* yang sudah ditetapkan. Masalah penjadwalan merupakan persoalan dari masalah optimasi yang ditemukan pada situasi nyata. Masalah ini membutuhkan waktu komputasi yang cukup tinggi untuk pencarian solusinya, terlebih lagi jika ukuran permasalahan semakin besar dengan bertambahnya jumlah komponen dan tetapan atau syarat yang ditentukan [1]. Selama proses optimasi, banyak aspek yang harus dipertimbangkan untuk memperoleh jadwal yang optimal, dan seringkali tidak dapat memberikan hasil yang memuaskan karena tidak semua kebutuhannya terpenuhi. Oleh karena itu, perlu ditetapkan suatu batasan dalam penyusunan jadwal yang bersifat *hard constraints* dan *soft constraints*.

Algoritma *Bees* merupakan algoritma optimasi yang terinspirasi dari perilaku alami lebah madu saat mencari makan untuk menemukan solusi optimal [2]. Algoritma *Bees* mirip dengan algoritma *Artificial Bee Colony* (ABC) dalam artian memiliki proses pencarian lokal dan pencarian global. Namun ada perbedaan di antara kedua algoritma tersebut selama proses pencarian lokal. Algoritma *Artificial Bee Colony* (ABC) menggunakan pendekatan probabilistik, sedangkan algoritma *Bees* tidak menggunakan pendekatan probabilistik, melainkan menggunakan evaluasi nilai *fitness* untuk mendorong pencariannya [3]. Algoritma *Harmony Search* adalah algoritma yang terinspirasi dari proses improvisasi suatu musik di mana sekelompok musisi akan saling mengimprovisasi pitch instrumen Mereka masing-masing untuk mencari harmoni yang sempurna sesuai standar estetika [4]. Algoritma *Harmony Search* menangani intensifikasi dan diversifikasi. Diversivikasi akan mempertahankan solusi lokal yang baik, mengeksplorasi ruang pencarian global yang lebih luas dan efisien, dan membuat solusi baru yang cukup baik mendekati solusi terbaik yang ada. Sedangkan intensifikasi akan memimpin proses pencarian menuju ruang pencarian solusi yang baik.

Menurut Baris Y., Michael S. P., Ernesto M., Duc T. P., dan Alfredo L. [3] Algoritma *Bees* memiliki kemampuan pencarian lokal dan global, dapat diimplementasikan kedalam beberapa masalah optimasi, mudah digunakan, dapat digabungkan dengan algoritma lain, dan memiliki parameter yang dapat diatur. Menurut Juliana W. [5] *Harmony Search* dapat digunakan untuk memecahkan masalah penjadwalan, namun masih dperlukan perbaikan untuk meningkatkan kinerja dari optimasi yang dilakukan, ada beberapa masalah yang perlu dipertimbangkan seperti pada bagian *memory consideration*, dan juga pada proses perbaikan

setelah improvisasi. Menurut Hadwan M., Masri A., Nasser R. S., dan Roug Q. [6] nilai pada *parameter harmony memory size* yang tinggi akan berkontribusi pada titik awal dari proses improvisasi yang lebih baik untuk mendapatkan solusi yang layak.

Gabungan dari algoritma *Bees* dan algoritma *Harmony Search* menyatukan kemampuan optimasi dari keduanya. Algoritma *Bees* menggunakan pencarian lokal dan pencarian global, sedangkan algoritma *Harmony Search* menggunakan *memory consideration*, *random selection* dan *pitch adjustment* [7].

Berdasarkan permasalahan yang sudah dijelaskan, maka tujuan yang ingin dicapai dari penelitian ini adalah untuk mengetahui tingkat performansi pada algoritma *Bees* dan *Harmony Search* pada penyelesaian kasus penjadwalan.

2. ISI PENELITIAN

2.1 Landasan Teori

2.1.1 Penjadwalan

Penjadwalan adalah pengalokasian sumber daya pada objek-objek yang ada terhadap ruang waktu yang tersedia sedemikian rupa sehingga sedapat mungkin memenuhi kondisi-kondisi tertentu [8]. Dalam proses penjadwalan, sumber daya yang ada harus dialokasikan secara optimal, efektif dan efisien, serta sedapat mungkin memenuhi kebutuhan yang ada. Fang [9] berpendapat bahwa penjadwalan merupakan masalah yang menantang di bidang Artificial Inteligence, hal ini disebabkan oleh beberapa hal yang di antaranya adalah:

1. Penjadwalan merupakan masalah *NP-Complete (Non-deterministic Polynomial Complete)*, yaitu suatu permasalahan dimana dengan bertambahnya variable secara linear maka waktu yang diperlukan untuk memecahkan masalah itu bertambah secara ekponensial. Hal ini sering disebut juga *Combinatorial Explosion*, selain itu ruang solusi yang mungkin untuk kebanyakan masalah riil terlalu besar untuk dilakukan *brute-force search* atau pencarian yang tidak terarah.
2. Teknik pencarian tingkat tinggi dengan menggunakan berbagai *heuristic* untuk memangkas *search space* tidak menjamin dihasilkannya suatu solusi yang optimal dan sangat sulit untuk mendesain suatu *heuristic* yang efektif.
3. Masalah penjadwalan sering menjadi rumit karena adanya *special constraints*.
4. Penjadwalan di dunia nyata sering melibatkan *constraint* yang tidak dapat dipresentasikan atau bahkan tidak dapat dinyatakan dengan tepat.

2.1.1.1 Constraints Penjadwalan

Di dalam penjadwalan dikenal dua jenis *constraint* (persyaratan), yaitu *hard constraints* dan *soft constraints*. *Hard constraints* adalah constraint yang tidak boleh dilanggar sama sekali sedangkan *soft constraints* adalah *constraints* yang sedapat mungkin dipenuhi. Pelanggaran terhadap satu *hard constraints* saja akan menyebabkan jadwal tersebut menjadi tidak berguna sedangkan pelanggaran terhadap *soft constraints* masih bisa diterima namun akan menimbulkan ketidaknyamanan pihak yang memakai jadwal tersebut [9].

2.1.2 Algoritma *Bees*

Menurut Pham [2] algoritma *Bees* adalah algoritma yang termasuk dalam bidang algoritma *Swarm Intelligence*, algoritma ini masih berhubungan dengan algoritma lain yang terinspirasi dari lebah seperti algoritma *Artificial Bee Colony*, dan algoritma *Swarm Intelligence* lain seperti *Ant Colony Optimization* dan *Particle Swarm Optimization*. Algoritma *Bees* terinspirasi dari perilaku lebah madu saat mencari makan. Lebah madu mengumpulkan nektar dari area yang luas di sekitar sarang Mereka. Suatu koloni mengirimkan sejumlah lebah untuk mengumpulkan nektar dari patch bunga. Lebah saling berkomunikasi menggunakan *waggle dance* untuk menginformasikan arah, jarak, dan penilaian kualitas dari sumber makanan kepada lebah lain di sarang. Tujuan pengolahan informasi dari algoritma *Bees* adalah untuk mencari dan mengeksplorasi *best patch* dalam ruang pencarian masalah. *Scout bee* dikirim keluar untuk mencoba ruang masalah secara acak dan menemukan *best patch*. *Best patch* dieksploitasi menggunakan *local search*, di mana sejumlah *best patch* akan lebih dieksplorasi dari yang lain. *Best patch* akan terus dieksploitasi, meskipun banyak *scout bee* dikirim keluar setiap iterasi untuk mencari *best patch* baru.

Algoritma *Bees* mirip dengan algoritma *Artificial Bee Colony* dalam artian memiliki proses pencarian lokal dan pencarian global. Namun ada perbedaan di antara kedua algoritma tersebut selama proses pencarian lokal. Algoritma *Artificial Bee Colony* menggunakan pendekatan probabilistik selama tahap pencarian lokal, sedangkan algoritma *Bees* tidak menggunakan pendekatan probabilistik, melainkan menggunakan evaluasi nilai *fitness* untuk mendorong pencariannya [3].

Tabel 1. *Parameter* Algoritma *Bees*

Parameter	**Simbol**
Jumlah *scout bees* pada *patches* yang terpilih	*n*
Jumlah *patches* terbaik dari *patches* yang terpilih	*m*
Jumlah *elite patches* pada *patches* terbaik yang terpilih	*e*
Jumlah *bees* yang direkrut pada *elite patches*	*nep*
Jumlah *bees* yang direkrut pada *non-elite patches* terbaik	*nsp*
Ukuran pencarian untuk setiap *patch*	*ngh*

2.1.3 Algoritma *Harmony Search*

Algoritma *Harmony Search* adalah algoritma yang terinspirasi dari proses improvisasi musisi, secara spesifik yaitu pada saat para musisi (yang mungkin sebelumnya belum pernah bermain bersama) dengan cepat berimprovisasi untuk menghasilkan *harmony* yang sesuai perkiraan estetika. Dengan menggunakan analogi tersebut, maka algoritma *Harmony Search* melakukan proses optimasi untuk mendapatkan keadaan terbaik dengan cara melakukan evaluasi terhadap nilai *fitness*. Seperti halnya perkiraan estetika yang ditentukan menggunakan himpunan *pitches* yang dikeluarkan alat musik, nilai *fitness* pada algoritma *Harmony Search* dihitung menggunakan himpunan nilai-nilai pada setiap variabel keputusan. Perbaikan nilai *fitness* pada algoritma *Harmony Search* menerapkan improvisasi yang terus ditingkatkan dari tiap iterasi ke iterasi, sama seperti perbaikan kualitas estetika yang diperbaiki dengan cara melakukan latihan demi latihan. Jika *harmony* baru yang diciptakan lebih baik dibandingkan dengan *harmony* sebelumnya, maka *harmony* baru ini akan tetap dipertahankan.

Berdasarkan konsep tersebut, secara garis besar algoritma Harmony Search terdiri dari lima tahapan, yaitu:

1. Identifikasi masalah dan parameter algoritma Harmony Search, dengan parameter yang dibutuhkan adalah sebagai berikut.

Tabel 2. *Parameter* algoritma *Harmony Search*

Parameter	Simbol
Harmony Memory Size, adalah banyaknya vektor solusi yang terbentuk	HMS
Harmony Memory Consider Rate, adalah probabilitas pemilihan komponen dari Harmony Memory yang ada, bernilai $0 \leq HMCR \leq 1$. Umumnya berkisar antara 0.7 sampai 0.99	HMCR
Pitch Adjustment Rate, adalah probabilitas suatu anggota Harmony Memory untuk dimutasi bernilai $0 \leq PAR \leq 1$. Umumnya berkisar antara 0.1 sampai 0.5.	PAR

2. Inisialisasi Harmony Memory, harmony memory awal didapatkan dari membangkitkan solusi secara acak
3. Improvisasi Harmony Baru, setiap komponen dari solusi ini didapatkan berdasarkan HMCR. HMCR didefinisikan sebagai probabilitas pemilihan komponen dari Harmony Memory yang ada. Jika x_1^j diambil dari Harmony Memory maka dipilih secara acak dari dimensi ke-j anggota Harmony Memory, dan dapat dimutasikan berdasarkan PAR. PAR adalah probabilitas suatu anggota Harmony Memory untuk dimutasi. Proses improvisasi ini mirip dengan proses pembuatan offspring pada algoritma genetika dengan proses mutasi dan crossover.
4. Memperbarui Harmony Memory. Solusi dari tahap ketiga akan dievaluasi, jika nilai fitness dari vektor solusi baru nilainya lebih baik daripada vektor terburuk yang ada pada Harmony Memory, maka vektor solusi baru akan digunakan, sebaliknya jika lebih buruk maka akan dieliminasi.
5. Kriteria berhenti, menentukan banyanya proses pembangkitan solusi baru dan memperbarui Harmony Memory. Apabila kriteria berhenti terpenuhi maka proses iterasi akan berhenti.

2.2 Analisis Masalah

Analisis masalah merupakan gambaran masalah yang diangkat dalam penelitian mengenai optimasi penjadwalan menggunakan algoritma *Bees* dan *Harmony Search*. Masalah penjadwalan kuliah membutuhkan waktu komputasi yang cukup tinggi untuk pencarian solusinya, hal ini disebabkan karena adanya berbagai komponen dan batasan yang ditentukan oleh pihak instansi yang akan menggunakan jadwal tersebut. Selama proses, banyak aspek yang harus dipertimbangkan sehingga menghasilkan jadwal yang optimal dan dapat digunakan. Oleh karena itu perlu ditetapkan suatu batasan dalam penyusunan jadwal yang bersifat harus dipenuhi (*hard constraints*) dan tidak harus dipenuhi tetapi tetap menjadi acuan dalam proses pembuatan jadwal (*soft constraints*).

Batasan yang harus dipenuhi (*hard constraints*) dalam penjadwalan yang akan dibuat adalah sebagai berikut.

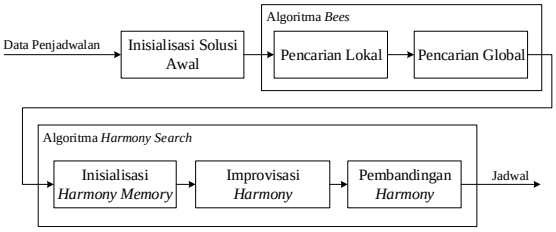
1. Satu ruang pada suatu slot waktu hanya dapat berlangsung satu kegiatan mata kuliah.
2. Satu dosen pada suatu slot waktu hanya dapat mengajar satu mata kuliah.
3. Tidak ada kegiatan kuliah selama waktu sholat Jumat.

Batasan yang tidak harus dipenuhi (*soft constraints*) dalam penjadwalan yang akan dibuat adalah sebagai berikut.

1. Waktu mengajar dosen sebaiknya disesuaikan dengan ketersediaan waktu dosen tersebut.

2.3 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan serta hambatan yang ada sehingga dapat diusulkan perbaikan-perbaikannya dalam membantu proses perancangan sistem tersebut.

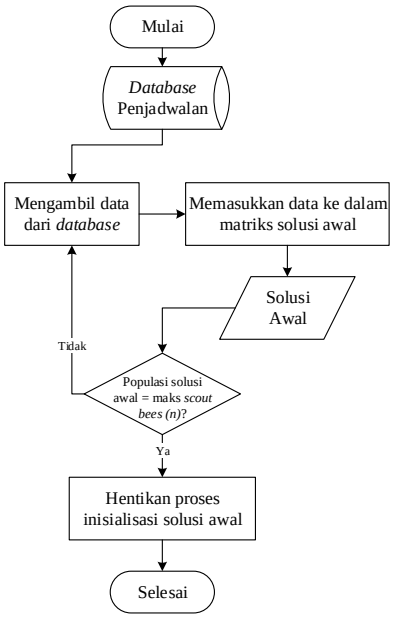


Gambar 1. Blok Diagram Sistem

2.4 Analisis Proses

2.4.1 Inisialisasi Solusi Awal

Proses inisialisasi solusi awal merupakan proses yang dilakukan untuk membangkitkan solusi awal yang nantinya akan digunakan sebagai dasar untuk pencarian solusi yang optimal. Jumlah populasi solusi awal yang dibangkitkan akan disesuaikan dengan nilai dari parameter jumlah populasi *scout bees* (n) yang diinputkan oleh *user*. Setiap data kelas, data jam, data hari, dan data ruang ditetapkan ke dalam matriks solusi awal, di mana data kelas ditetapkan secara berurut sedangkan data jam, data hari, dan data ruang ditetapkan secara acak.



Gambar 2. Inisialisasi Solusi Awal

2.4.1.1 Evaluasi Nilai Fitness Solusi Awal

Seperti yang sudah disebutkan pada analisis masalah, penjadwalan yang diteliti memiliki tiga *hard constraints* dan satu *soft constraints*, setiap *constraints* memiliki bobot nilai penalti=+1 untuk dievaluasi menggunakan rumus sebagai berikut.

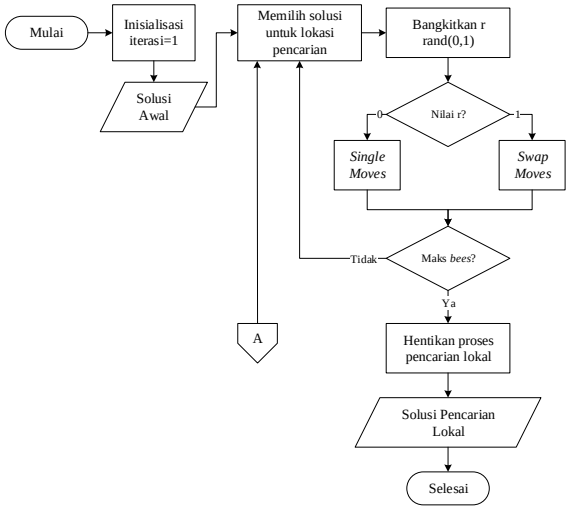
$$fitness = \frac{1}{(1+(penalty[1]+penalty[2]+penalty[3]+penalty[4]))} \quad (1)$$

Dengan batasan yang dianalisis adalah sebagai berikut.

1. *Constraint* 1 (*penalty*[1]), yaitu satu ruang pada suatu slot waktu hanya dapat berlangsung satu kegiatan mata kuliah. *Constraint* ini dapat dibagi lagi menjadi:
   a. Ketika sks >= 1 dan jam pertama = jam pertama mata kuliah lain pada hari dan ruangan yang sama.
   b. Ketika sks >= 2 dan jam kedua = jam pertama mata kuliah lain pada hari dan ruangan yang sama.
   c. Ketika sks >= 3 dan jam ketiga = jam pertama mata kuliah lain pada hari dan ruangan yang sama.
   d. Ketika sks >= 4 dan jam keempat = jam pertama mata kuliah lain pada hari dan ruangan yang sama.
2. *Constraint* 2 (*penalty*[2]), yaitu Satu dosen pada suatu slot waktu hanya dapat mengajar satu mata kuliah. *Constraint* ini dapat dibagi lagi menjadi:
   a. Ketika sks >= 1 dan jam pertama = jam pertama mata kuliah lain pada hari dan dosen yang sama.
   b. Ketika sks >= 2 dan jam kedua = jam pertama mata kuliah lain pada hari dan dosen yang sama.
   c. Ketika sks >= 3 dan jam ketiga = jam pertama mata kuliah lain pada hari dan dosen yang sama.
   d. Ketika sks >= 4 dan jam keempat = jam pertama mata kuliah lain pada hari dan dosen yang sama.
3. *Constraint* 3 (*penalty*[3]), yaitu tidak ada kegiatan kuliah selama waktu shalat Jumat. *Constraint* ini dapat dibagi lagi menjadi:
   a. Ketika sks >= 1 dan jam pertama = jam shalat jumat.
   b. Ketika sks >= 2 dan jam kedua = jam shalat jumat.
   c. Ketika sks >= 3 dan jam ketiga = jam shalat jumat.
   d. Ketika sks >= 4 dan jam keempat = jam shalat jumat.
4. *Constraint* 4 (*penalty*[4]), yaitu waktu mengajar dosen sebaiknya disesuaikan dengan ketersediaan waktu dosen tersebut.

2.4.2 Pencarian Lokal Algoritma *Bees*

Pada fase lgoritma *Bees*, pencarian lokal dibentuk berasarkan jumlah *bees* pada *best patch* (m) dengan melakukan perpindahan elemen dari kolom hari, jam, ataupun ruang pada solusi yang dipilih sebagai lokasi pencarian, perpindahan yang dilakukan dapat berupa *single moves* atau *swap moves*. Perpindahan ini dilakukan pada lokasi yang dipilih untuk mencari solusi alternatif dari calon jadwal suatu kelas matakuliah yang masih memiliki pelanggaran *constraint*. Pemilihan *single moves* atau *swap moves* dilakukan dengan membangkitkan angka *random* (r), jika bernilai 0 maka lakukan *single moves*, sedangkan jika bernilai 1 maka lakukan *swap moves*.



Gambar 3. Proses Pencarian Lokal

2.4.2.1 *Single Moves*

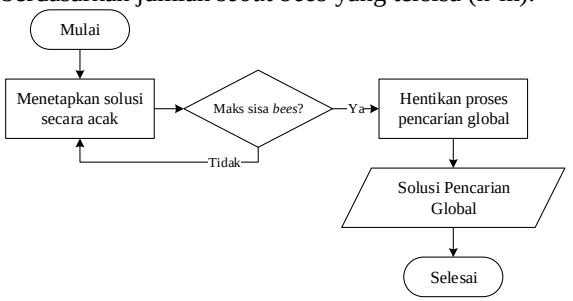
Perpindahan yang dilakukan pada single moves akan mengubah antara hari, jam atau ruang dari calon jadwal suatu kelas matakuliah. *Single moves* dilakukan terhadap kelas matakuliah yang masih memiliki pelanggaran *constraints* pada solusi yang ada. Pertama-tama, pilih kolom yang akan dilakukan *single moves* secara acak, jika kolom hari yang terpilih, maka lakukan pengubahan terhadap nilai pada kolom hari, jika kolom jam yang terpilih, maka lakukan pengubahan terhadap nilai pada kolom jam, jika kolom ruang yang terpilih, maka lakukan pengubahan terhadap kolom ruang.

2.4.2.1 *Swap Moves*

Perpindahan yang dilakukan pada *swap moves* akan mengubah hari, jam, dan ruang dari suatu kelas matakuliah. Setiap *swap move* menggunakan dua komponen, yaitu kelas matakuliah (ki) dan kelas matakuliah (kj). Pertama-tama, untuk kelas matakuliah (ki), pilih secara acak kelas matakuliah kj yang akan ditukar hari, jam, dan ruangnya dengan ki. Kemudian, lakukan swap move antara (ki, kj).

2.4.3 Pencarian Global Algoritma *Bees*

Pencarian global dibentuk dengan cara menetapkan solusi secara acak untuk mengeksplorasi potensi solusi yang baru berdasarkan jumlah *scout bees* yang tersisa (n-m).

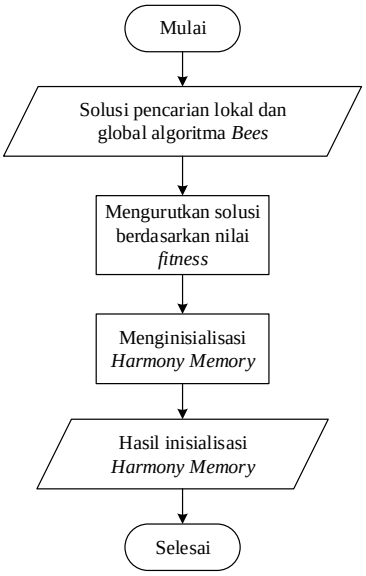


Gambar 4. Proses Pencarian Global

Pencarian global menetapkan solusi secara acak dari ruang pencarian, proses penetapan solusi ini akan terus dilakukan hingga mencapai jumlah maksimal scout bees yang tersisa.

2.4.4 Inisialisasi *Harmony Memory*

Pada tahap ini, solusi pencarian lokal dan global dari algoritma *Bees* akan diurutkan berdasarkan nilai *fitness*. Kemudian *Harmony Memory* diinisialisasi berdasarkan dari *Harmony Memory Size* (HMS).



Gambar 5. Inisialisasi *Harmony Memory*

2.4.5 Improvisasi *Harmony*

Pada algoritma *Harmony Search*, tiga langkah operasi akan diterapkan pada setiap improvisasi dari harmony baru, yaitu *Memory Consideration*, *Random Selection*, dan *Pitch Adjustment*. Tahap awal dari improvisasi *harmony* adalah membangkitkan angka acak r untuk dibandingkan dengan parameter HMCR yaitu probabilitas dilakukannya pemilihan komponen dari *Harmony Memory*. dengan rentang nilai HMCR 0-1.

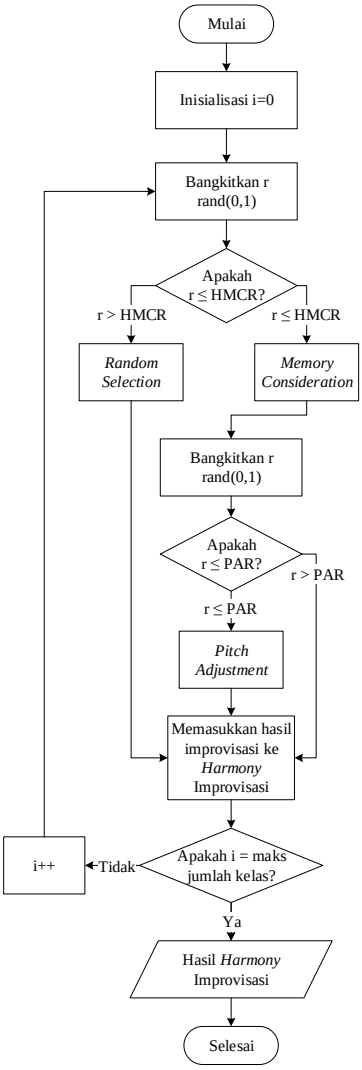
Jika r ≤ HMCR, maka lakukan *Memory Consideration.*

Jika r > HMCR, maka lalukan *Random Selection*.

Jika r ≤ HMCR, maka setelah tahap *Memory Consideration* akan dibangkitkan kembali angka acak r untuk dibandingkan dengan *parameter* PAR dengan rentang nilai PAR 0-1.

Jika r ≤ PAR, maka lakukan *Pitch Adjustment*.

Jika r > PAR, maka tidak dilakukan *Pitch Adjustment*.



Gambar 6. Improvisasi *Harmony*

2.4.5.1 *Memory Consideration*

Operasi pada *Memory Consideration* akan menetapkan hari, jam, dan ruang dari setiap kelas matakuliah pada *harmony* improvisasi berdasarkan solusi yang disimpan pada *Harmony Memory* dengan menggunakan perbandingan nilai r (yang didapat dari pembangkitan r baru dengan angka acak 0-1) dengan HMCR.

Jika r ≤ HMCR, maka lakukan *Memory Consideration*.

Jika r > HMCR, maka lewati tahap ini dan lakukan *Random Selection*.

2.4.5.2 *Pitch Adjustment*

Operasi *Pitch Adjustment* akan diterapkan terhadap hasil dari operasi *Memory Consideration* dengan menggunakan nilai PAR. Nilai jam, hari, dan ruang dari kelas matakuliah yang ditetapkan pada *harmony* improvisasi akan diambil dari *harmony* terbaik di *Harmony Memory* dengan menggunakan perbandingan nilai r dengan PAR.

Jika r ≤ PAR, maka lakukan *Pitch Adjustment*.

Jika r > PAR, maka tidak dilakukan *Pitch Adjustment*.

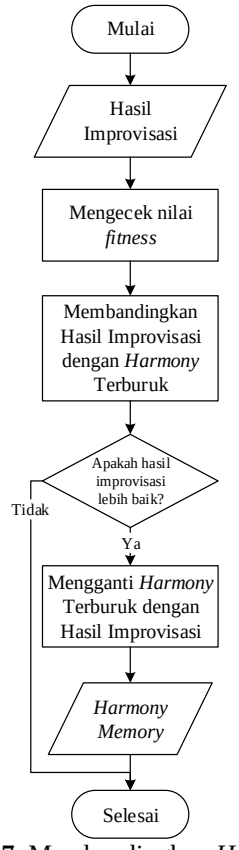
2.4.5.3 *Random Selection*

Pada tahap ini, jam, hari, dan ruang dari setiap kelas matakuliah pada *harmony* improvisasi akan dipilih secara acak dari kemungkinan solusi yang ada.

Jika r ≥ HMCR, maka lakukan *Random Selection*.

2.4.6 Pembandingan *Harmony*

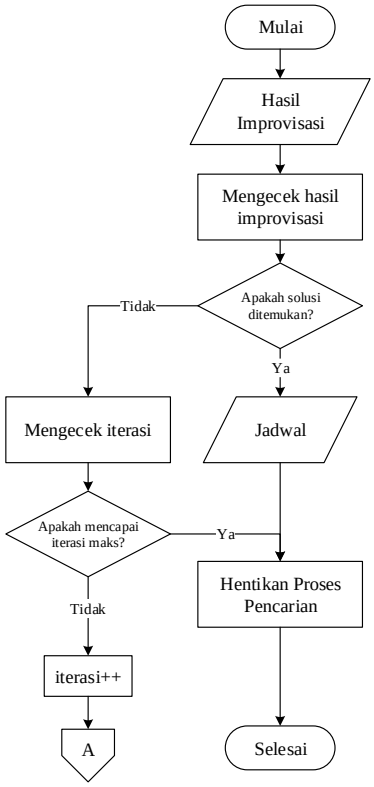
Pada tahap ini, *harmony* hasil improvisasi akan dibandingkan dengan *harmony* urutan terendah (nilai *fitness* terburuk) yang ada di *Harmony Memory*. Pembandingan ini dilakukan berdasarkan nilai *fitness*. Jika nilai *fitness harmony* improvisasi lebih baik daripada *harmony* terburuk yang ada di *Harmony Memory*, maka simpan *harmony* improvisasi ke *Harmony Memory* dan buang *harmony* terburuk dari *Harmony Memory*, begitupun sebaliknya.



Gambar 7. Membandingkan *Harmony*

2.4.6 Penghentian Proses

Proses akan berhenti jika memenuhi kriteria berhenti, yaitu mencapai jumlah maksimal iterasi atau menemukan solusi optimal, yaitu saat nilai *fitness* dari hasil improvisasi mencapai 1 (satu) yang artinya sudah tidak ada constraints yang dilanggar.



Gambar 8. Pengecekan Kriteria Berhenti

2.5 Pengujian

Sebelum dilakukan pengujian performansi akan dilakukan pengujian parameter dengan tujuan mendapatkan nilai parameter yang paling sesuai untuk digunakan dalam menemukan solusi dari penjadwalan yang diteliti. Nilai parameter yang paling sesuai ini didapatkan dengan mengamati jumlah rata-rata iterasi, waktu, dan nilai fitness yang diperlukan untuk menemukan solusi. Pengujian nilai dari setiap parameter ini dilakukan hingga menemukan solusi sebanyak lima kali.

Adapun nilai *parameter* dari algoritma *Bees* yang digunakan dalam pengujian *parameter* adalah sebagai berikut.

Tabel 3. *Parameter Bees* yang diuji

Parameter	**Nilai**
n	[20, 30]
m	[4, 5]
e	[3, 4]
nep	[4, 5]
nsp	[2, 3]
ngh	[5, 10]

Sedangkan nilai *parameter* dari algoritma *Harmony Search* yang diuji adalah sebagai berikut.

Tabel 4. *Parameter Harmony Search* yang diuji

Parameter	**Nilai**
HMCR	[0.7, 0.8, 0.95]
PAR	[0.3, 0.4, 0,5]

Untuk pengujian performansi, akan dilakukan dengan mengamati jumlah pelanggaran terhadap *soft constraints* yaitu ketersediaan dosen dengan prioritas normal, di mana *constraints* tersebut adalah *soft constraints* yang dapat dilanggar dan apabila terjadi pelanggaran maka pelanggaran tersebut masih bisa diterima. Pengujian ini menggunakan nilai *parameter* algoritma *Bees* dan *Harmony Search* berdasarkan hasil pengujian *parameter* yang sudah dilakukan sebelumnya. Adapun rencana jumlah slot waktu tidak bersedia yang akan digunakan dalam pengujian adalah sebagai berikut

Tabel 5. Slot waktu tidak bersedia yang diuji

Jumlah slot waktu tidak bersedia
30
60
90
120
150

2.5.1 Hasil Pengujian

Berdasarkan hasil pengujian *parameter*, didapatkan parameter yang optimal untuk menyelesaikan *masalah* penjadwalan yang diteliti adalah sebagai berikut.

Tabel 6. Parameter Optimal

Parameter	**Nilai**
n	20
m	5
e	4
nep	4
nsp	2
ngh	5
HMCR	0.8
PAR	0.3

Berdasarkan dari hasil pengujian performansi terhadap *soft constraints* yang sudah dilakukan, didapatkan hasil sebagai berikut.

Tabel 7. Hasil Pengujian Performansi

Jumlah slot waktu tidak bersedia	**Rata-rata pelanggaran *soft constraints***	**Rata-rata waktu**	**Persentase pelanggaran *soft constraints***
30	0	3:11:14	0%
60	5	3:08:15	8%
90	21	3:14:35	23%
120	37	3:11:02	31%
150	88	3:11:22	59%

Dapat disimpulkan bahwa semakin banyak jumlah slot waktu tidak bersedia yang ditetapkan maka semakin banyak pula jumlah rata-rata pelanggaran soft constraints yang terjadi.

Berdasarkan pengujian tersebut, dapat disimpulkan pula bahwa batas aman jumlah slot waktu tidak bersedia adalah antara 30 sampai dengan 60 slot waktu, dan banyaknya jumlah slot waktu tidak bersedia tidak terlalu berpengaruh terhadap rata-rata waktu yang digunakan untuk menemukan solusi.

3. PENUTUP

Berdasarkan hasil penelitian yang telah dilakukan, maka dapat diperoleh kesimpulan bahwa algoritma *Bees* dan *Harmony Search* dapat digunakan untuk menyelesaikan masalah penjadwalan dengan hasil yang optimal (tidak ada hard constraints yang dilanggar). Berdasarkan pengujian terhadap parameter didapatkan bahwa nilai parameter algoritma Bees dengan nilai n=20, m=5, e=4, nep=4, nsp=2, ngh=5 dan Harmony Search dengan nilai HMCR=0.8, PAR=0.3 akan mengoptimalkan proses pencarian solusi akhir. Berdasarkan dari hasil pengujian terhadap *soft constraints* yang sudah dilakukan, dapat disimpulkan bahwa semakin banyak jumlah slot waktu tidak bersedia yang ditetapkan maka semakin banyak pula jumlah rata-rata pelanggaran *soft constraints* yang terjadi, *persentase* pelanggaran *soft constraints* juga semakin meningkat bersamaan dengan semakin banyaknya jumlah slot waktu tidak bersedia yang ditetapkan. Diperoleh kesimpulan pula bahwa batas aman jumlah slot waktu tidak bersedia adalah antara 30 sampai dengan 60 slot waktu, dan banyaknya jumlah slot waktu tidak bersedia tidak terlalu berpengaruh terhadap rata-rata waktu yang digunakan untuk menemukan solusi.

3.1 Saran

Berdasarkan hasil dari penelitian yang telah dilakukan, masih diperlukan beberapa kajian lebih lanjut. Adapun saran untuk penelitian lebih lanjut adalah sebagai berikut.

1. Memperbaiki penanganan terhadap pelanggaran soft constraints agar jadwal yang dihasilkan menjadi lebih optimal.

DAFTAR PUSTAKA

[1] Suhartono E., 2015 "*Optimasi Penjadwalan Mata Kuliah Dengan Algoritma Genetika (Studi Kasus di AMIK JTC Semarang).*" Jurnal Ilmiah INFOKAM. No II, Th. XI Sept 15.

[2] Duc T. P., Ghanbarzadeh A., Koc E., Otri S., Rahim S., dan Zaidi M., 2006 "*The Bees Algorithm - A Novel Tool for Complex Optimisation Problems.*" Proceedings of IPROMS 2006 Conference, pp. 454-461.

[3] Baris Y., Michael S. P., Ernesto M., Duc T. P., dan Alfredo L. 2013 "*Honey Bees Inspired Optimization Method: The Bees Algorithm.*" Insects 4, no. 4: 646-662.

[4] Geem Z. W., Kim J. H., dan Loganathan G. V., 2001 "*A New Heuristic Optimization Algorithm: Harmony Search. Simulation.*" SIMULATION: Transactions of The Society for Modeling and Simulation International 76(2), pp. 60–68.

[5] Juliana W., dan Naimah M. H., 2013 "*Harmony Search Algorithm for Curriculum-Based Course Timetabling Problem.*" International Journal of Soft Computing and Software Engineering, Vol. 3, No. 3, pp. 365-371.

[6] Hadwan M., Ayob M., Nasser R. S., dan Roug Q., 2013. "*A harmony search algorithm for nurse rostering problems.*" Information Sciences Vol. 233, pp. 126–140.

[7] Khang N., Phuc N., dan Nuong T., 2012 "*A hybrid algorithm of Harmony Search and Bees Algorithm for a University Course Timetabling Problem.*" International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1.

[8] Anthony W., 1996 "*Scheduling, timetabling and rostering – A special relationship?*" LNCS 1153, 46–75.

[9] Fang H. F., 1994 "*Genetic Algorithms in Timetabling and Scheduling.*" Ph.D. Thesis, Department of Articial Intelligence, University of Edinburgh.