Penerapan *Harmony Search Algorithm* dalam Permasalahan Penjadwalan *Flow Shop*

[1]Indra Aulia, [1]Erna Budhiarti Nababan, [1]M. Anggia Muchtar

[1]Program Studi S1 Teknologi Informasi
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara
E-mail: indra@students.usu.ac.id | ernabrn@usu.ac.id | anggia.muchtar@usu.ac.id

***Abstrak*— Dunia manufaktur dan industri jasa memiliki permasalahan dibidang penjadwalan. Penjadwalan adalah proses pengambilan keputusan yang berkaitan tentang pengalokasian sejumlah resource dan tugas dalam waktu tertentu. Oleh karena itu, dibutuhkan suatu teknik untuk mendapatkan penjadwalan yang efektif dan efisien. Flow shop merupakan salah satu permasalahan penjadwalan yang dikategorikan dalam permasalahan NP-hard. Permutation akan menjadi kendala dalam lingkungan flow shop dimana mesin-mesin diatur secara seri dengan menyusun antrian job mengikuti aturan FIFO. Harmony search algorithm salah satu teknik metaheuristik yang terinspirasi dari permainan musik. Algoritma tersebut dapat digunakan untuk menyelesaikan permasalahan permutation flow shop untuk minimize makespan. Teknik penyelesaian dengan menggunakan algoritma tersebut menerapkan inisialisasi dengan variabel diskrit. Berdasarkan hasil pengujian yang telah dilakukan, harmony search algorithm efektif menghasilkan makespan yang lebih baik dari makespan yang terdapat pada benchmark data.**

Kata Kunci*— Flow shop, *harmony search algorithm*, makespan, penjadwalan, *permutation

I. PENDAHULUAN

Dunia manufaktur dan industri jasa memiliki permasalahan dibidang penjadwalan. Permasalahan tersebut berkaitan tentang pengalokasian sejumlah *resource* dan tugas selama waktu tertentu. Penjadwalan sebagai proses pengambilan keputusan memiliki peranan penting dalam berbagai sistem manufaktur dan produksi serta lingkungan pemrosesan informasi [1]-[2]. Oleh sebab itu, dibutuhkan pengembangan dan pendekatan untuk mendapatkan penjadwalan yang efektif dan efisien [3].

Flow shop merupakan lingkungan dimana mesin-mesin diatur secara seri dan semua *job* memiliki operasi yang dikerjakan dalam urutan atau rute yang sama [1]-[2]. *Flow shop* adalah salah satu masalah penjadwalan yang dikategorikan dalam permasalahan *NP-hard* [4].

Permasalahan penjadwalan flow shop yang kompleks sudah dilakukan pendekatan teknik matematis seperti *branch and bound* dan *mathematical programming*, namun teknik tersebut hanya berlaku efektif pada permasalahan dengan skala kecil [5].

Lebih dari lima dekade, permasalahan penjadwalan *flow shop* sudah menjadi penelitian para ahli. Penelitian-penelitian yang telah dilakukan, diantaranya menggunakan *genetic algorithm* [4], *ant colony algorithm* [6], *simulated annealing* [6], *particle swarm optimization* [7], *hybrid harmony search* [8] dan *discrete harmony search algorithm* [9]. Seiring dengan berkembangnya ilmu pengetahuan, pengembangan dan pengkajian terus dilakukan sehingga ditemukan beberapa metode baru salah satunya metode *harmony search algorithm*.

Harmony Search algorithm (*HSA*) adalah salah satu algoritma metaheuristik yang diusulkan oleh Zong Woo Geem pada tahun 2001. Algoritma tersebut terinspirasi oleh proses pertunjukan musik [11]. Dalam proses tersebut, dianalogikan seorang musisi mengimprovisasi *pitch instrument* dimana proses tersebut bertujuan untuk mendapatkan keadaan terbaik berdasarkan perkiraan estetika. Harmoni dalam musik tersebut merupakan representasi dari vektor solusi sedangkan proses improvisasinya merepresentasikan pencarian global atau lokal dalam teknik optimisasi [10]. *Harmony search algorithm* memiliki struktur yang relatif mudah karena tidak perlu melibatkan kalkulasi matematika yang kompleks [12].

Pada jurnal ini permasalahan penjadwalan *permutation flow shop* dengan kriteria makespan akan diselesaikan dengan menggunakan diskrit HSA. Diskrit HSA tersebut merepresentasikan *job* yang akan diproses dan dalam algoritma ini menjadi himpunan variabel keputusan yang tersedia.

II. IDENTIFIKASI MASALAH

Permasalahan penjadwalan *permutation flow shop* merupakan permasalahan yang memiliki n *job* dengan waktu proses pada m mesin. Urutan operasi *job* pada semua mesin adalah searah untuk masing-masing *job* dan sama pada setiap mesin [13].Secara sederhana dapat diartikan apabila suatu *job* berada pada posisi ke-i pada mesin 1, maka *job* tersebut akan berada di posisi ke-i pada semua mesin. Sehingga persoalan tersebut dianalogikan sama sebagai antrian FIFO (*first in first out*) [1]-[2].

Permasalahan tersebut dalam dunia penjadwalan dapat dinotasikan *Fm | prmu | Cmax*. Fungsi objektif pada permasalahan ini pada (1).

$$Minimize\ F(x) = Minimize\ Cmax \quad (1)$$

dimana

$$\sum_{q=1}^{n} y_{iq} = 1 \qquad i = 1,2,\ldots,n \quad (2)$$

$$\sum_{i=1}^{n} y_{iq} = 1 \qquad q = 1,2,\ldots,n \tag{3}$$

$$h_{kq} + \sum_{q=1}^{n} y_{ik} y_{iq} \le h_{k+1,q} \tag{4}$$

$k = 1,2,\ldots,m-1; q = 2,3,\ldots,n;$

$$h_{kq} + \sum_{q=1}^{n} y_{ik} y_{iq} \le h_{k,q-1} \tag{5}$$

$k = 1,2,\ldots,m; q = 2,3,\ldots,n-1;$

Persamaan (2) dan (3) menjelaskan bahwa setiap job hanya diproses pada satu mesin dan setiap urutan mesin diisi dengan satu job. Persamaan (4) mengharuskan waktu awal setiap pekerjaan pada mesin k+1 tidak lebih awal dari waktu akhir proses di mesin k. Sedangkan persamaan (5) mengharuskan *job* j dapat diproses pada mesin k+1 jika *job* tersebut telah selesai diproses pada mesin k.

III. PENELITIAN TERDAHULU

Permasalahan penjadwalan *flow shop* telah menjadi fokus penelitian di bidang optimasi. Dalam menyelesaikan permasalahan tersebut para peneliti telah banyak menggunakan algoritma-algoritma yang dapat memberikan hasil yang sesuai harapan.

Etiler et al[4] telah mencoba menyelesaikan permasalahan penjadwalan *flow shop* dengan menggunakan algoritma genetika. Mereka meneliti permasalahan *flow shop* dengan fungsi objektifnya adalah makespan. Algoritma genetika akan dibandingkan dengan algoritma NEH yang merupakan metode heuristik yang populer digunakan dalam menyelesaikan permasalahan tersebut. Hasilnya algoritma ini mudah dalam pengimplementasian pada permasalahan *flow shop* dengan kinerja yang efektif.

Permasalahan *flow shop* dengan kendala permutasi diselesaikan dengan menggunakan algoritma ant-colony [6]. Dalam penelitian ini, mereka mengembangkan dua jenis varian dari algoritma *ant-colony* yakni M-MMAS dan PACO. Hasilnya PACO lebih baik dari M-MMAS dalam permasalahan *permutation flow shop* dengan ukuran permasalahan yang besar dibandingkan permasalahan dengan ukuran permasalahan yang kecil.

Algoritma *simulated annealing* (SAA) dikembangkan untuk menyelesaiakan permasalahan penjadwalan *permutation flow shop* [7]. Namun pada penelitian ini melahirkan hybrid SAA. Hybrid SAA menghasilkan nilai makespan dengan pencarian yang cepat dibandingkan dengan algoritma metaheuristik yang lain seperti algoritma genetik dan algoritma SA.

Penelitian *flow shop* dengan menggunakan algoritma *particle swarm* dikembangkan dengan variabel diskrit [8]. PSO dengan pencarian lokal menberikan kinerja yang baik dalam penjadwalan *flow shop* dengan fungsi objektif *total flow time* namun membutuhkan waktu komputasi yang lebih.

Penelitian dibidang *flow shop* dengan menggunakan algoritma *hybrid harmony search* menerapkan inisialisasi pada HM digunakan teknik heuristik dengan variabel kontinu [9]. Hasilnya algoritma ini dapat menyelesaikan permasalahan ini dengan baik.

Penelitian dibidang *flow shop* dengan menggunakan *discrete harmony search algorithm* merupakan kelanjutan dari penelitian yang dilakukan oleh Ling et al[9] namun menggunakan variabel diskrit dengan kendala *no-wait flow shop* [10]. Hasilnya algoritma ini efektif dalam menyelesaikan permasalahan tersebut.

Penelitian-penelitian terdahulu yang membahas tentang permasalahan penjadwalan *flow shop* yang dapat dirangkum dalam tabel 1.

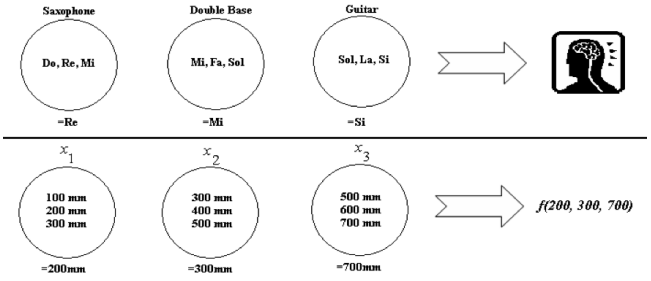
Tabel 1.
Penelitian Terdahulu Flow Shop Scheduling

No	Nama (Tahun)	Penelitian	Algoritma
1	Etiler et al (2004)	Flow Shop	Membandingkan algoritma genetika dengan NEH
2	Rajendran dan Ziegler (2004)	Permutation Flow Shop	Membandingkan dua varian algoritma ant-colony
3	Nearchou (2004)	Permutation Flow Shop	Hybrid simulated annealing
4	Ching et al (2007)	Flow Shop	Dicrete particle swarm
5	Ling et al (2010)	Blocking Flow Shop	Hybrid harmony search algorithm dengan inisialisasi NEH
6	Kin et al (2011)	No-Wait Flow Shop	Discrete harmony search algorithm dengan inisialisasi NEH

IV. METODOLOGI

Harmony Search algorithm (HSA) adalah salah satu algoritma metaheuristik yang diusulkan oleh Zong Woo Geem pada tahun 2001. Algoritma tersebut terinspirasi oleh proses pertunjukan musik ketika musisi mencari harmoni yang lebih baik [11]. Pencarian harmoni pada proses improvisasi musik bertujuan untuk mendapatkan keadaan terbaik berdasarkan perkiraan estetika.

Dengan analogi tersebut, HSA melakukan proses optimasi untuk mendapatkan keadaan terbaik dengan cara mengevaluasi fungsi objektif. Seperti halnya perkiraan estetika yang ditentukan menggunakan himpunan *pitches* yang dikeluarkan alat musik, fungsi objektif pada HSA dihitung menggunakan himpunan nilai-nilai pada setiap variabel keputusan (*decision variables*). Perbaikan nilai fungsi objektif pada HSA menerapkan improvisasi yang terus ditingkatkan dari iterasi ke iterasi sama seperti perbaikan kualitas suara estetika yang diperbaiki dengan latihan demi latihan.



Gambar 1 Struktur Harmony Search Algorithm

Pada gambar 2.1, setiap pemain musik (saxophonist, double bassist, dan guitarist) merepresentasikan suatu

decision variable (x1, x2, dan x3). Kumpulan bunyi yang dihasilkan oleh setiap instrumen musik (saxophone = { Do, Re, Mi }; double bass = { Mi, Fa, Sol }; guitar = { Sol, La, Si }) menyatakan rentang nilai variabel (x1 = { 100, 200, 300 }; x2 = { 300, 400, 500 }; x3 = { 500, 600, 700}). Sebagai contoh, misalnya saxophonist mengeluarkan bunyi Re, double bassist membunyikan Fa dan guitarist mengeluarkan bunyi La, maka ketiganya membangun suatu harmoni baru (Re, Fa, La). Jika harmoni ini lebih indah dibandingkan harmoni saat ini, maka harmoni baru ini dipertahankan. Harmoni yang diperoleh tersebut dalam dunia optimasi disebut dengan solusi yang direpresentasikan dalam bentuk dimensi vektor solusi.

Berdasarkan konsep di atas, harmony search algorithm terdiri dari lima tahapan [11] :

a. Identifikasi masalah dan parameter HS
b. Inisialisasi harmony memory
c. Improvisasi harmoni baru
d. Harmony memory update
e. Lakukan pengulangan tahap c dan d hingga kriteria berhenti tercapai

Pada permasalahan penjadwalan flow shop, HSA diimplementasikan dengan tahapan sebagai berikut :

A. Identifikasi Masalah dan Parameter HS

Minimize (atau *maximize*) $f(x)$

dengan $x_i \in X_i$, i = 1,2,…, N

dimana $f(x)$ adalah suatu fungsi objektif

x_i adalah variabel keputusan ke *i*

X_i adalah himpunan variabel keputusan

N merupakan jumlah variabel keputusan

Parameter-parameter yang dibutuhkan adalah sebagai berikut :

a. *Harmony Memory Size* (HMS) adalah banyaknya vektor solusi yang terbentuk
b. *Harmony Memory Consideration Rate* (HMCR) bernilai 0 ≤HMCR ≤1. Umumnya berkisar antara 0.7 sampai 0.99
c. *Pitch Adjustment Rate* (PAR) bernilai 0 ≤HMCR ≤1. Umumnya berkisar antara 0.1 sampai 0.5
d. Kriteria berhenti adalah banyaknya iterasi untuk melakukan improvisasi

B. Inisialisasi *Harmony Memory*

Persamaan (6) merepresentasikan inisialisasi HM yang didapatkan dengan membangkitkan variabel keputusan x_i secara acak sehingga membentuk vektor solusi X_i. Kemudian hitung nilai fungsi objektif $f(x)$ masing-masing vektor solusi.

$$HM = \left[\begin{array}{ccccc|c} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 & f(x^2) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} & f(x^{HMS-1}) \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} & f(x^{HMS}) \end{array}\right] \quad (6)$$

Dalam penelitian ini, anggota dari himpunan variabel keputusan merupakan representasi dari banyaknya *job* yang akan diproses. Sebagai contoh, jika terdapat 6 *job* yang akan diproses pada 5 mesin maka himpunan variabel keputusan yang terbentuk adalah $X = \{1,2,3,4,5,6\}$, Sehingga inisialisasi HM yang mungkin terbentuk sebagai berikut :

$$HM = \begin{bmatrix} 2 & 3 & 1 & 4 & 5 \\ 1 & 4 & 2 & 3 & 5 \\ 4 & 1 & 3 & 2 & 5 \\ 3 & 1 & 4 & 5 & 2 \end{bmatrix}$$

Setelah didapat vektor solusi sebanyak HMS yakni banyaknya baris dari HM, maka selanjutnya hitunglah nilai fungsi objektif masing-masing vektor tersebut. Sehingga didapat sebagai berikut :

$$HM = \left[\begin{array}{ccccc|l} 2 & 3 & 1 & 4 & 5 & f(C_{\max}) = 438 \\ 1 & 4 & 2 & 3 & 5 & f(C_{\max}) = 457 \\ 4 & 1 & 3 & 2 & 5 & f(C_{\max}) = 428 \\ 3 & 1 & 4 & 5 & 2 & f(C_{\max}) = 458 \end{array}\right]$$

C. Improvisasi Harmoni Baru

Improvisasi harmoni baru dilakukan pembangkitan harmoni baru x_i sehingga membentuk vektor solusi baru $x' = (x_1', x_2', \ldots, x_N')$. Pembangkitan harmoni baru tersebut dilakukan dengan dua aturan yakni :

a. Harmony memory consideration

Pada tahap ini, nilai variabel keputusan x' dipilih secara acak dari variabel-variabel mana saja yang tersimpan dalam HM $\left(\{x_i^1, x_i^2, \ldots, x_i^{HMS}\}\right)$ dengan probabilitas $0 \le HMCR \le 1$.

Pembangkitan variabel keputusan yang tidak berada pada HM, maka akan dipilih secara acak dari himpunan variabel x' dengan probabilitas $1 - HMCR$. Pembangkitan variabel keputusan baru pada tahp ini seperti (7).

$$x_i' \leftarrow \begin{cases} x_i' \in \left(\{x_i^1, x_i^2, \ldots, x_i^{HMS}\}\right) & HMCR \\ x_i' \in X_i & HMCR - 1 \end{cases} \quad (7)$$

b. *Pitch adjustment*

Tahap ini merupakan tahap penyesuaian variabel keputusan baru x_i yang dihasilkan pada tahap harmony memory consideration. Variabel keputusan tersebut akan disesuaikan dengan variabel-variabel tetangganya dengan probabilitas $HMCR \times PAR$.

Variabel keputusan x_i yang dihasilkan akan harmony memory consideration dipertahankan dengan probabilitas $HMCR \times (1 - PAR)$. Penyesuaian variabel pada tahap ini ada dua

formulasi yakni variabel diskrit menggunakan aturan (8).

$$x_i^{'} \leftarrow \begin{cases} x_i^{'}(k+m) & HMCR \times PAR \\ x_i^{'} & HMCR \times (1-PAR) \end{cases} \quad (8)$$

dimana k adalah indeks pada elemen X_i

$x_i^{'}$ adalah variabel keputusan ke i

X_i adalah variabel ke-k pada elemen X_i

m adalah indeks tetangga (+1 atau -1)

Pada penelitian ini menggunakan parameter HMCR dan PAR yakni HMCR = 0.8 dan PAR = 0.4. Pada Tahap ini dapat dirancang pseudocode Fesanghary (2005) berikut ini

```
while (stopCondition())
    {
        for (int i = 0; i < NVAR; i++)
        {
            if (rand() < HMCR)
            {
                memoryConsideration(i);
                if (rand() < PAR)
                    pitchAdjustment(i);
            }
            else
                randomSelection(i);
        }
        double[] currentFit;
        currentFit = fitness(NCHV);
        updateHarmonyMemory(currentFit);
        generation++;
    }
```

Gambar 2 *Pseudocode* Improvisasi *Harmony Search Algorithm*

Dalam penelitian ini, proses improvisasi tersebut direpresentasikan sebagai berikut :

- Iterasi 1
  Bilangan random yang dibangkitkan misalnya $r_1 = 0.5986$. Nilai tersebut kemudian disesuaikan dengan parameter HMCR. Karena bilangan acak yang dibangkitkan lebih kecil dari nilai HMCR, maka varibel keputusan dipilih secara acak dengan aturan (7).
  $$x_1^{'} = 4$$
  Selanjutnya dilakukan pembangkitan bilangan acak untuk melakukan penyesuaian dengan nilai PAR. Bilangan yang dibangkitkan secara acak pada tahap PAR adalah $r_2 = 0.4174$. Karena bilangan acak tersebut lebih besar dari nilai PAR, maka tetap mempertahankan $x_1^{'} = 4$
- Itearsi 2
  Bilangan random yang dibangkitkan misalnya $r_1 = 0.2701$. Nilai tersebut kemudian disesuaikan dengan parameter HMCR. Karena bilangan acak yang dibangkitkan lebih kecil dari nilai HMCR, maka varibel keputusan dipilih secara acak dengan aturan (7).
  $$x_2^{'} = 3$$
  Selanjutnya dilakukan pembangkitan bilangan acak untuk melakukan penyesuaian dengan nilai PAR. Bilangan yang dibangkitkan secara acak pada tahap PAR adalah $r_2 = 0.1121$. Karena bilangan acak tersebut lebih kecil dari nilai PAR, maka pilihlah variabel keputusan $x_2^{'}$ yang bertetanggaan dengan $x_2^{'} = 3$. Penyesuaian dengan tetanggaannya dilakukan dengan aturan (8) dimana k bernilai -1, sehingga didapat variabel keputusan baru $x_2^{'} = 2$.
- Iterasi 3
  Bilangan random yang dibangkitkan misalnya $r_1 = 0.9285$. Nilai tersebut kemudian disesuaikan dengan parameter HMCR. Karena bilangan acak yang dibangkitkan lebih besar dari nilai HMCR, maka varibel keputusan dipilih secara acak dengan aturan (7). Sehingga variabel keputusan yang terbentuk adalah $x_3^{'} = 1$.
- Iterasi 4
  Bilangan random yang dibangkitkan misalnya $r_1 = 0.9875$. Nilai tersebut kemudian disesuaikan dengan parameter HMCR. Karena bilangan acak yang dibangkitkan lebih besar dari nilai HMCR, maka varibel keputusan dipilih secara acak dengan aturan (7). Sehingga variabel keputusan yang terbentuk adalah $x_4^{'} = 1$.
- Iterasi 5
  Bilangan random yang dibangkitkan misalnya $r_1 = 0.8215$. Nilai tersebut kemudian disesuaikan dengan parameter HMCR. Karena bilangan acak yang dibangkitkan lebih besar dari nilai HMCR, maka varibel keputusan dipilih secara acak dengan aturan (7). Sehingga variabel keputusan yang terbentuk adalah $x_5^{'} = 3$.

Setelah proses iterasi selesai dilakukan maka didapat suatu vektor solusi baru $x_{new}^{'} = [4 \;\; 2 \;\; 1 \;\; 5 \;\; 3]$. Vektor harmoni baru selanjutnya dihitung nilai fungsi objektifnya sehingga di dapat $f(C_{\max}) = 465$

D. Upgrade HM

Nilai fungsi objektif dari vektor solusi baru nilainya buruk daripada vektor-vektor yang ada pada HM, maka vektor-vektor solusi dalam HM tetap dipertahankan.

E. Kriteria Berhenti

Kriteria berhenti menentukan banyaknya proses pembangkitan solusi baru dan memperbarui HM. Apabila kriteria berhenti terpenuhi maka proses iterasi akan berhenti.

V. HASIL DAN PEMBAHASAN

Pengujian untuk menyelesaikan masalah penjadwalan *flow shop* menggunakan dataset dari benchmark Taillard (1993). Parameter yang akan digunakan pada pengujian adalah HMS = 5; HMCR = 0.9; PAR = 0.3 yang disarankan oleh Omran & Mahdavi[12].

Masing-masing data akan diuji dengan banyak improvisasi 500, 1000, 2000, 3500, dan 5000. Dimana akan

dilakukan pengujian sebanyak 10 kali replikasi pada setiap jenis data dan banyaknya improvisasi.

A. Pengujian 5 mesin 20 job

Pengujian ini melibatkan 5 mesin dengan 20 job. Benchmark yang digunakan adalah ta001. Hasil pengujian dataset tersebut disajikan pada tabel 2.

Tabel 2.
Hasil Pengujian 5 Mesin 20 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	1232	1232	1232	1248	1259
2	1232	1254	1264	1241	1232
3	1264	1267	1232	1232	1232
4	1254	1232	1264	1232	1264
5	1264	1248	1232	1232	1237
6	1293	1264	1232	1254	1232
7	1267	1232	1232	1237	1232
8	1292	1249	1232	1244	1232
9	1255	1249	1232	1232	1232
10	1273	1264	1232	1232	1237
Makespan	**1232**	**1232**	**1232**	**1232**	**1232**

Pengujian tersebut dilakukan sebanyak 10 kali replikasi pada setiap iterasi. Hasil pengujian menunjukkan makespan yang minimal adalah 1232. Hasil tersebut didapat pada setiap iterasi yang telah dilakukan.

B. Pengujian 10 mesin 20 job

Pengujian ini melibatkan 10 mesin dengan 20 job. Benchmark yang digunakan adalah ta007. Hasil pengujian dataset tersebut disajikan pada tabel 3.

Tabel 3.
Hasil Pengujian 10 Mesin 20 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	1288	1270	1274	1266	1274
2	1274	1280	1274	1268	1274
3	1278	1287	1274	1286	1265
4	1282	1257	1270	1265	1274
5	1274	1274	1270	1266	1270
6	1289	1280	1270	1282	1265
7	1288	1282	1278	1266	1265
8	1282	1277	1274	1265	1265
9	1289	1288	1277	1266	1270
10	1260	1257	1282	1274	1274
Makespan	**1260**	**1257**	**1270**	**1265**	**1265**

Hasil pengujian untuk dataset tersebut menunjukkan makespan yang minimal adalah 1257. Hasil dengan makespan tersebut didapat pada iterasi 1000. Namun, ketika iterasi pengalami kenaikan menjadi 2000, 3500 dan 5000 hasil yang didapat menjadi kurang baik daripada iterasi 1000.

C. Pengujian 20 mesin 20 job

Pengujian ini melibatkan 20 mesin dengan 20 job. Benchmark yang digunakan adalah ta004. Hasil pengujian dataset tersebut disajikan pada tabel 4.

Tabel 4.
Hasil Pengujian 20 Mesin 20 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	1936	1840	1837	1841	1862
2	1886	1873	1886	1869	1874
3	1847	1900	1878	1863	1818
4	1911	1850	1872	1830	1855
5	1864	1867	1879	1858	1792
6	1912	1850	1808	1857	1778
7	1864	1814	1867	1836	1837
8	1907	1854	1867	1812	1830
9	1887	1892	1818	1845	1834
10	1898	1874	1846	1840	1864
Makespan	**1847**	**1814**	**1808**	**1812**	**1778**

Hasil pengujian untuk dataset tersebut menunjukkan makespan yang minimal adalah 1778. Hasil dengan makespan tersebut didapat pada iterasi 5000. Peningkatan iterasi pada setiap uji coba, menjelaskan perbaikan nilai makespan yang cukup baik, namun ketika iterasi 3500, makespan yang didapat kembali turun menjadi kurang baik dari iterasi 500, 1000 dan 2000.

D. Pengujian 5 mesin 50 job

Pengujian ini melibatkan 5 mesin dengan 50 job. Benchmark yang digunakan adalah ta009. Hasil pengujian dataset tersebut disajikan pada tabel 5.

Tabel 5.
Hasil Pengujian 5 Mesin 50 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	2452	2440	2440	2440	2440
2	2440	2448	2440	2440	2440
3	2452	2440	2440	2440	2440
4	2440	2440	2441	2440	2440
5	2453	2440	2440	2440	2440
6	2440	2441	2440	2440	2440
7	2447	2452	2440	2440	2440
8	2445	2440	2440	2440	2440
9	2453	2440	2440	2440	2440
10	2465	2445	2440	2440	2440
Makespan	**2440**	**2440**	**2440**	**2440**	**2440**

Hasil pengujian untuk dataset tersebut menunjukkan makespan yang minimal adalah 2440. Hasil tersebut dapat diperoleh pada setiap iterasi yang dilakukan dalam pengujian data.

E. Pengujian 10 mesin 50 job

Pengujian ini melibatkan 10 mesin dengan 50 job. Benchmark yang digunakan adalah ta005. Hasil pengujian dataset tersebut disajikan pada tabel 6.

Tabel 6.
Hasil Pengujian 10 Mesin 50 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	2846	2804	2800	2768	2760
2	2774	2769	2799	2761	2804
3	2844	2786	2767	2760	2816
4	2871	2809	2816	2781	2812
5	2808	2813	2815	2804	2767
6	2804	2819	2775	2760	2760
7	2831	2795	2810	2815	2792
8	2798	2790	2821	2806	2786
9	2778	2795	2804	2760	2787
10	2850	2786	2820	2766	2794
Makespan	**2774**	**2769**	**2767**	**2760**	**2760**

Hasil pengujian untuk dataset tersebut menunjukkan makespan yang minimal adalah 2760. Hasil tersebut diperoleh pada iterasi 3500 dan 5000. Tahap pengujian data setiap iterasi menjelaskan bahwa terjadi perbaikan hasil makespan dari iterasi 500, 1000 dan 2000.

F. Pengujian 20 mesin 50 job

Pengujian ini melibatkan 20 mesin dengan 50 job. Benchmark yang digunakan adalah ta003. Hasil pengujian dataset tersebut disajikan pada tabel 7.

Tabel 7.
Hasil Pengujian 50 Mesin 50 Job

Replikasi	Kriteria Iterasi				
	500	1000	2000	3500	5000
1	3048	3094	3125	3081	3076
2	3146	3130	3093	3093	3102
3	3120	3109	3093	3102	3124
4	3143	3089	3129	3071	3065
5	3138	3138	3094	3084	3100
6	3110	3141	3093	3097	3061
7	3083	3127	3138	3084	3103
8	3090	3093	3061	3115	3104
9	3077	3093	3112	3096	3048
10	3111	3117	3063	3045	3032
Makespan	**3048**	**3089**	**3061**	**3045**	**3032**

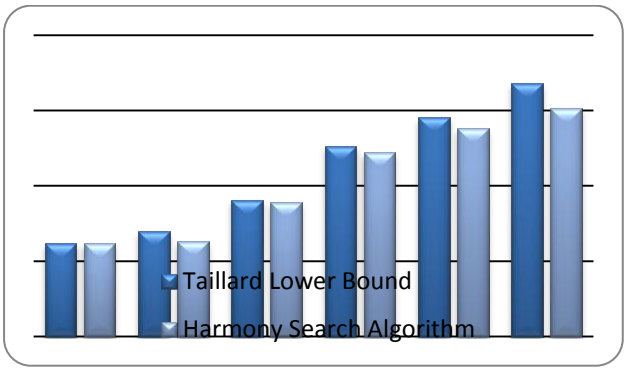
Hasil pengujian untuk dataset tersebut menunjukkan makespan yang minimal adalah 3032. Hasil tersebut diperoleh pada iterasi 5000. Namun setiap peningkatan iterasi makespan yang diperoleh tidak stabil, kadang terjadi peningkatan dan penurunan.

Hasil pengujian tersebut menggambarkan hasil pengujian yang bervariasi pada setiap iterasi dan replikasi. Hasil yang bervariasi tersebut didapat dari tahap improvisasi yang melibatkan penggunaan bilangan random dalam memperbaiki nilai makespan. Namun, hasil yang bervariasi, HSA mampu mendapatkan hasil yang lebih baik dari *lower bound* dataset Taillard untuk enam ukuran permasalahan *flow shop* tersebut. Sehingga hasil pengujian yang telah dilakukan dapat dirangkum pada tabel 8.

Tabel 8.
Hasil Akhir Pengujian

No	Dataset	Taillard (Lower Bound)	Harmony Search Algorithm
1	5 x 20	1232	1232
2	10 x 20	1388	1257
3	20 x 20	1810	1778
4	5 x 50	2527	2440
5	10 x 50	2908	2760
6	50 X 50	3351	3032

Tabel 8 menjelaskan hasil makespan dari pengujian enam dataset menggunakan *harmony search algorithm* dan hasil *makespan lower bound* yang terdapat dalam dataset Taillard. Dari tabel tersebut dapat dijelaskan bahwa penyelesaian masalah *flow shop* dengan menggunakan *harmony search algorithm* mampu menghasilkan hasil makespan yang lebih baik dari nilai *lower bound* setiap jenis dataset. Hasil pengujian tersebut dapat dijelaskan dengan gambar 3.



Gambar 3 Grafik Perbandingan Hasil Akhir Harmony Search Algorithm dengan Lower Bound Bechmarck Taillard

VI. KESIMPULAN

Harmony search algorithm dapat digunakan dalam menyelesaikan permasalahan penjadwalan *flow shop*. Dari pengujian yang telah dilakukan dapat disimpulkan bahwa *harmony search algorithm* mampu menghasilkan nilai makespan minimum pada dataset yang memiliki ukuran permasalahan yang besar. Dari pengujian yang telah dilakukan dengan iterasi 500, 1000, 2000, 3500 dan 5000 algoritma tersebut efektif menghasilkan solusi yang baik pada iterasi 3500 dan 5000.

Selanjutnya penelitian ini memiliki potensi untuk dikembangkan lebih lanjut. Pada penelitian selanjutnya disarankan untuk menambahkan teknik-teknik yang dapat memberikan kestabilan hasil dalam setiap iterasi dan memperkuat nilai *initial solution* HSA dengan mempertimbangkan tingkat kemampuan komputer.

DAFTAR PUSTAKA

[1] Pinedo, M.L. 2011. *Scheduling Theory, Algorithm, and System*. 4th edition. New York: Springer.

[2] Pinedo M.L. *2002. Scheduling: theory, algorithms, and systems, 2nd edition. Prentice-Hall, New Jersey.*

[3] Quan-Ke Pan, Ling Wang, dan Bao-Hua Zhao. 2008. An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *International Journal Advanced Manufacturing Technology* 38: hal. 778-786.

[4] Etiler,O., Toklu, Atak dan Wilson. 2004. A genetic algorithm for flow shop scheduling problems. *Journal of the Operational Research Society* 55: hal. 830-835.

[5] Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang, dan Xion Wang. 2008. A hybrid differential evolution method for permutation flow-shop scheduling. *International Journal Advanced Manufacturing Techonology* 38: hal. 757-777.

[6] Rajendran, Chandrasekharan dan Ziegler, Hans. 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research* 155: hal. 426–438.

[7] Nearchou, A. C. 2004. Flow shop sequencing using hybrid simulated annealing. *Journal of Intelligent Manufacturing* 15: hal. 317-328.

[8] Ching-Jong Liao, Chao-Tang Tseng dan Pin Luarn. 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research* 34: hal. 3099 – 3111.

[9] Ling Wang, Quan-Ke Pan dan Tasgetiren, M.F. 2010. Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. *Expert System with Applications* 37: hal. 7929-7936.

[10] Kai-zhou Gao, Quan-ke Pan dan Jun-qing Li. Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. *International Journal Advanced Manufacturing Techonology* 56: hal. 683-692.

[11] Zong Woo Geem(ed.). 2009. *Music-Inspired Harmony Search Algorithm Theory and Applications*. New York: Springer.

[12] Omran, M.G.H. dan Mahdavi, M. 2008. Global-best harmony search. *Applied Mathematics and Computation* 198: hal. 643-656.

[13] Sviridensko, M.I. 2004. A note on permutation flow shop problem. *Annals of Operations Research* 129: hal. 247–252.