TENSOR Pure and Applied Mathematics Journal

April 2021 | Vol. 2 | No. 1 Halaman 5–12

p-ISSN: 2723-0325 e-ISSN: 2723-0333

Cuckoo Search Algorithm untuk Menyelesaikan Bi-Objective Permutation Flowshop Scheduling Problem

Siti Sarah[1], Herry Suprajitno[1*], Asri Bekti Pratiwi[1]

[1] Department of Mathematics, Universitas Airlangga, Indonesia.
*Email: Email: herry-s@fst.unair.ac.id

Manuscript submitted : March 2021;
Accepted for publication : April 2021.

Abstract: Tujuan dari penelitian ini adalah menyelesaikan permasalahan *Bi-objective Permutation Flowshop Scheduling Problem* (BPFSP) menggunakan *Cuckoo Search Algorithm* (CSA). BPFSP memiliki lebih dari satu fungsi tujuan yaitu meminimalkan *makespan* dan total *tardiness*. Program penerapan CSA untuk menyelesaikan BPFSP diimplementasikan dalam kasus dengan tiga jenis data yaitu data kecil dengan 5-pekerjaan 4-mesin, data sedang dengan 20-pekerjaan 10-mesin, dan data besar 50-pekerjaan 20-mesin dengan penggunaan beberapa nilai parameter yang bervariasi diantaranya maksimum iterasi, banyaknya sarang serta probabilitas pergantian sarang. Berdasarkan hasil running pada ketiga jenis data diperoleh bahwa semakin banyak jumlah sarang serta iterasi maka akan memberikan nilai fungsi tujuan BPFSP yang cenderung lebih baik. Sebaliknya, nilai fungsi tujuan BPFSP akan cenderung lebih baik jika nilai probabilitas pergantian sarang semakin kecil.

2010 Mathematical Subject Classification : 68W05, 90B35.
Keywords: Bi-Objective Permutation Flowshop Scheduling Problem, Cuckoo Search Algorithm, Permutation Flowshop Scheduling Problem.

1. Pendahuluan

Masalah *Flowshop Scheduling Problem* (FSP) berperan penting dalam sistem manufaktur. FSP merupakan teknik penjadwalan yang bagus secara signifikan dapat meningkatkan efisiensi produksi. Agar mencapai urutan yang baik dalam persaingan pasar, metode penjadwalan yang efektif selalu dibutuhkan. *Permutation Flowshop Scheduling Problem* (PFSP) adalah salah satu masalah penjadwalan produksi yang paling popular. Dalam PFSP, setiap mesin hanya dapat memproses satu pekerjaan pada satu waktu, di mana urutan pekerjaan yang sama diikuti di semua mesin. Meskipun pada umumnya masalah ini dipelajari sebagai permasalahan dengan satu fungsi tujuan, namun kenyataannya terdapat juga permasalahan PFSP yang memerlukan pengoptimalan pada dua fungsi tujuan (*bi-objective*), seperti terdapatnya waktu tenggat untuk suatu pekerjaan, sehingga jika pekerjaan tersebut terlambat untuk diselesaikan maka perusahaan akan dikenakan pinalti atau beban keterlambatan (*weighted tardiness*) yang merugikan. Oleh karena itu selain meminimumkan *makespan* diperlukan juga untuk meminimalkan total beban keterlambatan [1].

Permasalahan ini disebut juga sebagai *Bi-Objective Permutation Flowshop Scheduling Problem* (BPFSP).

Beberapa metode telah digunakan untuk menyelesaikan BPFSP diantaranya adalah algoritma Branch and Bound dengan fungsi tujuan meminimalkan *total flow time* dan *total tardiness* (Lee dan Wu, 2001), serta algoritma Ant Colony Optimization (ACO) yang bertujuan untuk mengoptimalkan dua fungsi tujuan yaitu *makespan* dan *total flowtime* (Yenisey dan Yagmahan). *Cuckoo Search Algorithm* (CSA) adalah salah satu algoritma metaheuristik yang terinspirasi oleh alam, dikembangkan pada tahun 2009 oleh Xin-She Yang dan Suash Deb [2]. Cuckoo adalah burung yang menarik, bukan hanya karena suara indah yang mereka dapat buat, tetapi juga karena strategi reproduksi agresif burung tersebut. Meskipun burung *cuckoo* mungkin membuang telur spesies lain untuk meningkatkan kemungkinan penetasan telur mereka sendiri, cukup jumlah spesies melibatkan parasitisme induk yang bertanggung jawab dengan bertelur di dalam sarang burung tuan rumah lainnya [3].

Pada penelitian sebelumnya, terdapat beberapa ahli seperti Civicioglu dan Besdok (2011) yang telah membuktikan bahwa CSA mempunyai hasil lebih baik dari *Particle Swarm Optimization* (PSO) dan *Artificial Bee Colony* (ABC) [4]. Serta Gandomi dkk, 2011, telah memecahkan berbagai masalah optimasi dan menyimpulkan bahwa CSA mempunyai hasil solusi yang lebih baik dari PSO dan Genetic Algorithm [5]. Keuntungan lebih lanjut dari CSA adalah pencarian globalnya yang menggunakan *Lévy Flights Random Walk* (LFRW). Keuntungan ini, dikombinasikan dengan kemampuan pencarian lokal dan kovergensi global yang terjamin, membuat CSA sangat efisien [3]. Oleh karena itu, sangatlah menarik untuk mengembangkan penerapan CSA untuk menyelesaikan BPFSP.

Artikel ini disusun dengan sebagai berikut. Bagian selanjutnya membahas mengenai model BPFSP. Selanjutnya pada Bagian 3 disajikan penjelasan mengenai CSA, sedangkan pada Bagian 4 membahas mengenai tahapan penerapan CSA untuk menyelesaikan BPFSP. Hasil dan pembahasan yang berisi running program disajikan di Bagian 5. Terakhir, Kesimpulan disajikan di Bagian 6.

2. Bi-Objective Permutation Flowshop Scheduling Problem (BPFSP)

Penjadwalan Flowshop adalah salah satu pengaturan penjadwalan yang paling banyak dipelajari. Banyak pekerjaan yang menggunakan *Flowshop* dengan tujuan untuk meminimalkan *makespan,* yaitu total waktu penyelesaian. Dalam masalah ini, terdapat satu himpunan *n* pekerjaan perlu diproses dalam satu himpunan *m* mesin dengan urutan mesin yang sama untuk setiap pekerjaan. Semua pekerjaan mengikuti proses yang sama melalui mesin, yaitu, mereka pertama kali diproses pada mesin 1, lalu pada mesin 2 dan seterusnya sampai mesin *m* [6].

Pada permutation flowshop, *processing times* untuk semua pekerjaan telah diketahui sebelumnya, dengan urutan *n job* telah ditentukan. *Job* menjalankan operasi pada semua mesin yang ada tanpa mengubah urutannya dengan ketentuan [7]:

1. Semua pekerjaan yang tersedia dan siap untuk dilaksanakan pada awal penjadwalan.
2. Setiap pekerjaan hanya diproses di satu mesin pada saat yang sama.
3. Setiap pekerjaan diproses pada semua mesin dengan urutan mesin 1,2,…, *m*.
4. Sebuah pekerjaan harus diselesaikan dahulu prosesnya secara keseluruhan di sebuah mesin sebelum diproses di mesin selanjutnya.
5. Tidak ada penghentian operasi yang sedang dikerjakan.
6. Mesin selalu tersedia tidak pernah mengalami kerusakan.

Pada *Permutation Flowshop Scheduling Problem* (PFSP) terdapat lebih dari satu fungsi tujuan (*bi-objective*), yaitu meminimalkan *makespan* dan total *tardiness.* Dimana *makespan* yaitu meminimalkan waktu penyelesaian maksimum yang diperlukan untuk memproses semua pekerjaan, sedangkan total *tardiness* merupakan total keseluruhan keterlambatan dari semua pekerjaan.

Makespan adalah waktu penyelesaian seluruh pekerjaan pada semua mesin. Perhitungan *makespan* adalah sebagai berikut [8]:

$$c(\pi_1,1) = p_{\pi(1),1} \quad (1)$$

$$c(\pi_1,j) = c(\pi_1,j-1) + p_{\pi(1),j} \quad (2)$$

$$c(\pi_i,1) = c(\pi_{i-1},1) + p_{\pi(i),1} \quad (3)$$

$$c(\pi_i,j) = \max\{c(\pi_{i-1},j), c(\pi_i,j-1)\} + p_{\pi(i),j} \quad (4)$$

$$C_{max} = c(\pi_n, m) \quad (5)$$

dengan $i = 1,2,\ldots,n$ dan $j = 1,2,\ldots,m$, sehingga nilai *makespan*

$$M = c(\pi_{(n)}, m) \quad (6)$$

Total beban keterlambatan (*total weighted tardiness*) merupakaan total keseluruhan beban keterlambatan dari semua pekerjaan. Cara untuk menghitung total beban keterlambatan (*TT*) adalah sebagai berikut:

$$T_{\pi(i)} = maks(c_{\pi(i),m} - d_{\pi(i)}, 0) \quad i = 1,2,3,\ldots,n \quad (7)$$

$$TT = \sum_{i=1}^{n} T_{\pi(i)} \quad (8)$$

Weighted Method merupakan metode umum untuk menggabungkan banyak fungsi tujuan, dimana tiap fungsi tujuan akandialokasikan dengan bobot (w_i) antara 0 dan 1 kemudian dijumlahkan sesuai dengan bobotnya yang penjumlahannya menghasilkan nilai 1. Pada permasalahan permutasi *flowshop* ini terdapat dua fungsi tujuan yang akan diminimalkan yakni *makespan* dan total beban keterlambatan, sehingga dengan menggunakan *Weighting Method,* fungsi tujuan yang akan diminimalkan pada permasalahan ini adalah:

$$Min\ Z = w_1 * C_{max} + w_2 * TT \quad (9)$$

$$w_1 + w_2 = 1, w_1, w_2 > 0 \quad (10)$$

Pendefisian parameter dapat dilihat pada Tabel 1.

Tabel 1. Parameter yang digunakan pada modem matematika BPFSP

Notasi	Keterangan
$p_{\pi(i),j}$	Waktu pemrosesan pada pekerjaan *i* di mesin *j*
$p_{\pi(i),1}$	Waktu pemrosesan pada pekerjaan *i* di mesin 1
$c(\pi_1, j)$	Waktu penyelesaian pekerjaan ke 1 pada mesin *j*
$c(\pi_1, 1)$	Waktu penyelesaian pekerjaan ke *i* pada mesin 1
$c(\pi_i, j)$	Waktu penyelesaian pekerjaan ke *i* pada mesin *j*
N	Banyaknya pekerjaan
M	Banyaknya mesin
$d_{\pi(i)}$	Batas waktu penyelesaian (*due date*) pekerjaan yang dikerjakan pada urutan ke *i*
$T_{\pi(i)}$	Durasi keterlambatan pekerjaan *i*
Min Z	Fungsi tujuan (*bi-objective function)*
w_1, w_2	Bobot dari masing-masing fungsi tujuan
C_{max}	Fungsi tujuan *makespan*
TT	Fungsi tujuan total *tardiness*

3. Cuckoo Search Algorithm

Cuckoo Search Algorithm (CSA) adalah salah satu algoritma yang terinspirasi dari alam, yaitu terinspirasi dari sifat parasite beberapa spesies *cuckoo* yang meletakkan telurnya di sarang burung inang lainnya.

Untuk membuat algoritma sesuai untuk masalah optimasi, beberapa penyederhanaan yang nyataperilaku di alam diperlukan. Secara khusus, CSA didasarkan pada tiga aturan yang diidealkan:

1. Setiap *cuckoo* meletakkan satu telur pada suatu waktu di sarang yang dipilih secara acak.
2. Sarang dengan telur terbaik (yaitu, kualitas tinggi solusi) akan dibawa ke generasi yang berikutnya, sehingga memastikan bahwa solusi yang baik dipertahankan dari waktu ke waktu.
3. Jumlah sarang burung lain yang tersedia adalah tetap, dan burung pemilik sarang dapat menemukan telur *cuckoo* dengan probabilitas $p_a \in [0,1]$. Dalam kasus ini, burung pemilik sarang dapat

membuang telur cuckoo atau meninggalkan sarang sehingga pemilik sarang dapat membangun sarang baru di lokasi yang baru. Dengan asumsi bahwa pa merupakan sebagian kecil dari n sarang dapat digantikan oleh sarang baru yang telah didapatkan (dengan solusi acak baru di lokasi yang baru) [2].

Berikut ini adalah beberapa pengertian yang digunakan dalam *Cuckoo Search Algorithm* (CSA) :

a. Telur

Diasumsikan bahwa *cuckoo* meletakkan satu telur pada satu sarang, dan satu telur tersebu merupakan sebuah solusi yang direpresentasikan oleh satu individu (sarang) dalam populasi. Sebuah telur dapat menjadi salah satu calon solusi baru untuk sebuah tempat atau lokasi yang telah dimiliki oleh satu individu lain dalam populasi tersebut [9].

b. Sarang

Sebuah sarang merupakan individu dari populasi yang memiliki jumlah tetap dan merepresentasikan besar populasi sehingga pada pergantian sarang akan melibatkan penggantinya dalam populasi dengan individu baru [9].

c. Fungsi Tujuan

Setiap solusi dalam ruang pencarian dikaitkan dengan nilai nilai objektif numeric sehinggakualitas nilainya sebanding dengan nilai fungsi tujuan. Dalam *Cuckoo Search Algorithm* (CSA), sarang telur dengan kualitas yang lebih baik akan mengarah pada generasi baru yang berarti kualitas telur kukuk berhubungan langsung untuk kemampuannya memberikan *cuckoo* baru [9].

d. Ruang Pencarian

Ruang pencarian merepresentasikan posisi dari sarang yang mungkin pada daerah *feasible*.Posisi sarang dapat diubah dengan memodifikasi nilai dari koordinatnya.Dapat dipastikan bahwa perpndahan sarang atau lokasi sarang tidak melebihi batasan yang sebenarnya [9].

e. *Random Walks*

Pada dasaranya, burung mencari makanan secara acak di alam bebas. Cara paling efektif untuk seekor burung mencari makanannya adalah dengan menggunakan srategi *Random Walks,* karena langkah selanjutnya ditentukan berdasarkan lokasi saat ini dan peluang peralihan ke tempat selanjutnya. *Cuckoo Search Algorithm* (CSA) menggunakan dua *random walks,* yaitu *Lévy Flights Random Walk* (LFRW) dan *Biased Random Walk* (BRW) [10].

Pertama, LFRW merupakan *random walk* dengan *step length* berdasarkan distribusi *Lévy*. Berdasarkan implementasinya [4], *Lévy Flights* merupakan pencarian solusi baru yang berada disekitar solusi terbaik sementara dan dilakukan berdasarkan persamaan (11).

$$x_i{}^{t+1} = x_i{}^{t} + \alpha . S . (x_{best}{}^{t} - x_i{}^{t}) . \kappa \tag{11}$$

dengan $x_i{}^{t}$ merupakan sarang ke i pada iterasi ke t, $\alpha > 0$ merupakan parameter *step length*, S merupakan *Lévy flights* dari algoritma Mantegna, κ merupakan bilangan real acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1 , $x_{best}{}^{t}$merupakan sarang terbaik pada iterasi ke t.

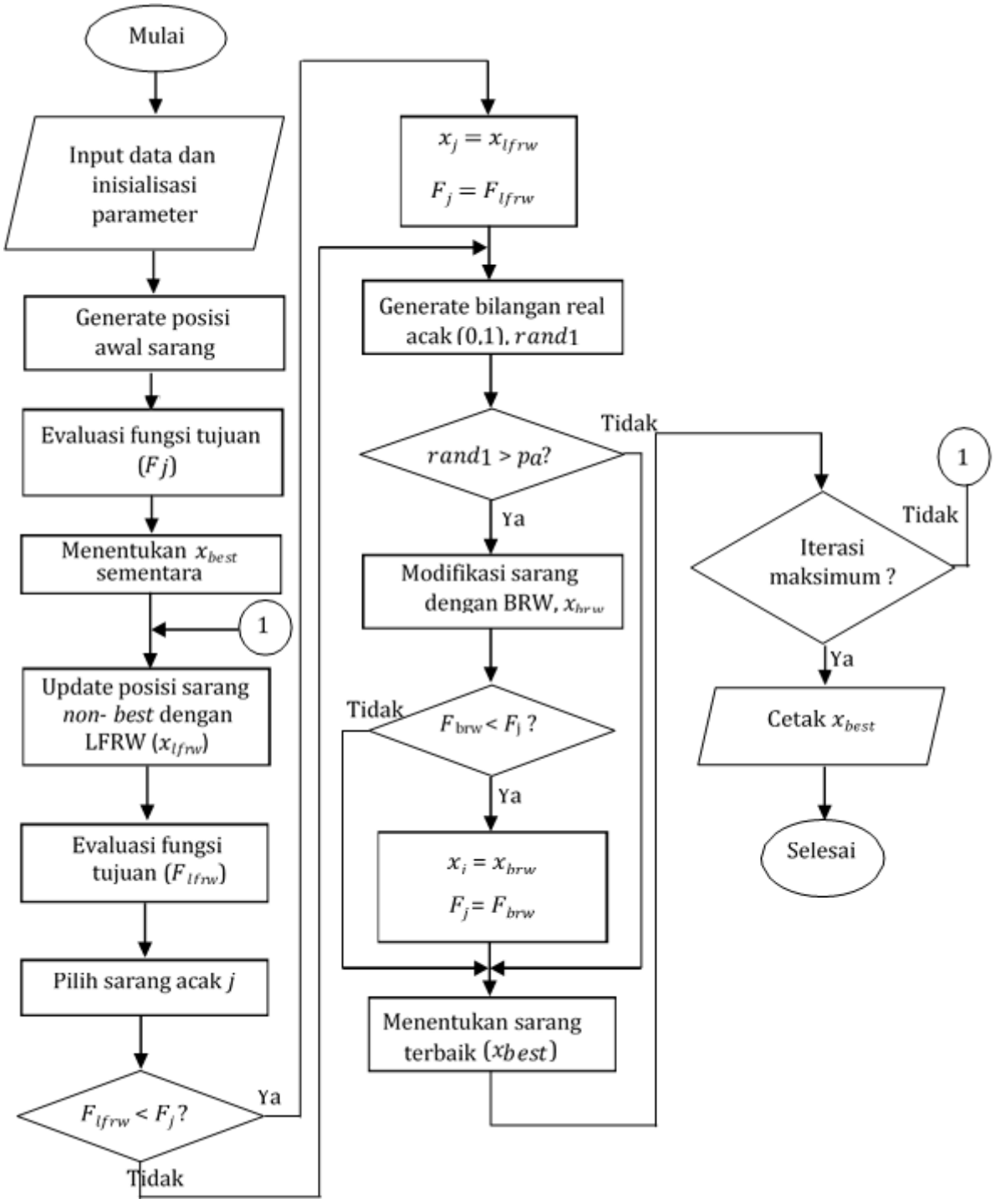
Menurut [11], *step length* (S) dapat dihitung dengan menggunakan

$$S = \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \tag{12}$$

dengan β merupakan parameter pada interval $[1,2]$ dan disarankan β bernilai 1,5 , variabel u dan v bilangan real acak yang berdistribusi normal dengan rata-rata 0 dan simpangan baku 1, dan nilai σ ditentukan berdasarkan :

$$\sigma = \left(\frac{\Gamma(1+\beta).\sin(\pi.\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right).\beta.2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}} \tag{13}$$

dengan Γ merupakan fungsi gamma. Menurut [12], fungsi $\Gamma: (0,\infty) \to \mathbb{R}$ dirumuskan sebagai $\Gamma(x) = \int_0^\infty u^{x-1} e^{-u}\, du$



Gambar 1. Flowchart Cuckoo Search Algorithm.

Kedua, *BRW* digunakan untuk menemukan solusi baru secara acak yang berada diposisi yang lebih jauh dari solusi terbaik.Cara yang digunakan yaitu membangun solusi baru dengan menggunakan solusi saat ini sebagai dasar dan dengan dua solusi lainnya yang dipilih secara acak, dan solusi baru didapatkan dari solusi baru dengan solusi saat ini menggunakan operasi penjumlahan.BRW dapat diformulasikan sebagai berikut :

$$x_i{}^{t+1} = \begin{cases} x_i{}^t + r_i\left(x_{permut1}{}^t - x_{permut2}{}^t\right), & jika\ rand_1 > p_a \\ x_i{}^t & ,jika\ rand_i \le p_a \end{cases} \quad (14)$$

dengan indeks acak *permut*$_1$ dan *permut*$_2$ merupakan bilangan bulat yang diambil secara acak dari populasi sarang yang ada, $rand_1$dan r merupakan bilangan $real$ yang diambil secara acak pada interval [0,1] dan

$p_a = 0.25 \in [0,1]$ merupakan probabilitas telur *cuckoo* yang ditemukan oleh burung pemilik sarang [10]. Langkah-langkah algoritma CSA disajikan pada Gambar 1.

4. Hasil dan Pembahasan

Data yang digunakan pada contoh kasus diambil dari [13] dan http://soa.iti.es/files/Thailard_duedates.7z, . Data yang digunakan adalah data kecil dengan 5-pekerjaan dan 4-mesin [13], data sedang dengan 20-pekerjaan dan 10-mesin (http://soa.iti.es/files/Thailard_duedates.7z), dan data besar dengan 50-pekerjaan dan 20-mesin (http://soa.iti.es/files/Thailard_duedates.7z). Berikut ini perbandingan solusi terbaik data kecil, data sedang dan data besar yang dihasilkan dengan menggunakan variasi parameter jumlah sarang, maksimum iterasi dan probabilitas pergantian sarang, serta nilai nilai α= 0.01, β = 1.5 $\in$ [1,2]. Solusi ini didapatkan menggunakan bahasa pemrograman Java NetBeans IDE 8.0.2. Hasil running program data kecil disajikan pada Tabel 2, data sedang disajikan pada Tabel 3, dan data besar disajikan pada Tabel 4.

Tabel 2. Hasil *Running* Program pada Data 5-pekerjaan dan 4-mesin.

Maks Iterasi	**Jumlah Sarang**	p_a		
		0.1	**0.5**	**0.9**
10	**10**	**48.5**	**48.5**	**48.5**
	50	**48.5**	**48.5**	**48.5**
	100	**48.5**	**48.5**	**48.5**
100	**10**	**48.5**	**48.5**	**48.5**
	50	**48.5**	**48.5**	**48.5**
	100	**48.5**	**48.5**	**48.5**
500	**10**	**48.5**	**48.5**	**48.5**
	50	**48.5**	**48.5**	**48.5**
	100	**48.5**	**48.5**	**48.5**

Pada Tabel 2 dapat dilihat bahwa dengan menggunakan parameter bervariasi, semuanya menghasilkan satu solusi terbaik dengan nilai fungsi tujuan yaitu 48.5 satuan waktu. Hasil tersebut diperoleh dengan urutan pekerjaan $\mathbf{4 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 5}$. Dapat disimpulkan bahwa semakin banyak jumlah maksimum iterasi dan jumlah sarang, serta semakin kecil nilai probabilitas pergantian sarang ($\boldsymbol{p_a}$) maka tidak berpengaruh pada hasil *running* dikarenakan jumlah pekerjaan yang sedikit sehingga perubahan pada urutan pekerjaan terbatas. Sedangkan hasil penyelesaian data sedang disajikan pada Tabel 3. Berdasarkan hasil dari Tabel 3 dapat disimpulkan bahwa semakin banyak jumlah sarang dan jumlah iterasi maka dari nilai fungsi tujuan yang diperoleh cenderung lebih baik, serta semakin kecil nilai probabilitas pergantian sarang ($\boldsymbol{p_\alpha}$) maka nilai fungsi tujuan yang diperoleh cenderung lebih baik.

Tabel 3. Hasil *Running* Program pada Data 20-pekerjaan dan 10-mesin.

Maks Iterasi	**Jumlah Sarang**	p_a		
		0.1	**0.5**	**0.9**
10	**10**	3174	3275.5	3114
	50	2942	3112	3200
	100	2978.5	2982.5	2990.5
100	**10**	2879.5	2754	3111.5
	50	2720.5	2775	2902
	100	2617	2685	2675.5
1000	**10**	2877.5	2844	3218
	50	2837	2984.5	3033
	100	**2582**	2756.5	2997.5

Parameter yang memberikan pengaruh yang baik terhadap solusi terbaik dari hasil *running* program adalah saat menggunakan jumlah sarang 100 dengan 1000 iterasi dan nilai probabilitas ($\boldsymbol{p_\alpha}$) sebesar 0.9 diperoleh nilai fungsi tujuan sebesar 2582 satuan waktu. Urutan pekerjaan dari solusi terbaik adalah 5 - 12 - 17 - 13 - 14 - 19 - 6 - 15 - 16 - 4 - 7 - 18 - 1 - 2 - 3 - 20 - 10 - 11 - 8 – 9. Hasil penyelesaian data besar disajikan pada Tabel 4. Solusi terbaik dari hasil *running* program adalah saat menggunakan 100 sarang dengan 100 iterasi dan nilai probabilitas ($\boldsymbol{p_\alpha}$) sebesar 0.1 diperoleh nilai fungsi tujuan sebesar 17875.5 satuan waktu. Urutan pekerjaan dari solusi terbaik adalah 43 - 33 - 45 - 1 - 18 - 20 - 26 - 44 - 23 - 39 - 41 - 42 - 19 - 11 - 46 - 12 - 49 - 34 - 7 - 40 - 3 - 24 - 14 - 32 - 31 - 6 - 8 - 47 - 38 - 29 - 5 - 37 - 50 - 2 - 4 - 16 - 13 - 35 - 27 - 17 - 48 - 25 - 10 - 28 - 36 - 15 - 21 - 30 - 9 – 22.

Tabel 4. Hasil *Running* Program pada Data 50-pekerjaan 20-mesin.

Maks Iterasi	**Jumlah sarang**	$\boldsymbol{p_a}$		
		0.1	**0.5**	**0.9**
10	**10**	21967.5	21348.5	22835
	50	21017.5	21124.5	21741.5
	100	20369.5	21014.5	20892.5
100	**10**	20643.5	20588.5	21788.5
	50	18655	20225.5	20589.5
	100	**17875.5**	1878.5	19701
1000	**10**	20752.5	20295.5	21737.5
	50	19473.5	20236	19588.5
	100	18153	19056	19115.5

Berdasarkan hasil Tabel 4 dapat disimpulkan bahwa semakin bertambahnya jumlah sarang maka solusi yang diperoleh semakin baik serta semakin bertambahnya iterasi dan semakin kecil nilai probabilitas (p_a) maka solusi yang didapatkan cenderung lebih baik.

5. Kesimpulan

Cuckoo Search Algorithm (CSA) dapat diterapkan untuk menyelesaikan *Bi-objective Permutation Flowshop Scheduling Problem* (BPFSP). BPFSP pada penelitian ini bertujuan untuk meminimalkan *makespan* dan total *tardiness*. Untuk menguji keefektifan algoritma, CSA diterapkan pada tiga tipe ukuran data, yaitu data kecil, data sedang dan data besar. Berdasarkan tiga tipe ukuran data yang digunakan tersebut dapat dilihat bahwa semakin besar jumlah sarang dan banyaknya iterasi memberikan nilai fungsi tujuan yang cenderung lebih baik, sedangkan perubahan nilai probabilitas (p_α) semakin kecil maka memberikan solusi fungsi tujuan yang cenderung lebih baik.

Daftar Pustaka

[1] Wang, X., & Tang, W. (2016). *A Memetic Algorithm for Multi-objective Distributed Permutation Flow Shop Scheduling Problem*. Northeastern University, China.

[2] Yang, X.S., & Deb, S. (2010). Cuckoo Search via Levy Flight in *Proc. of World Congress on Nature & Biologically Inspired Computing* (NaBIC 2009). Inida. IEEE Publication, USA.

[3] Kaveh, A., & Bakhspoori, T. (2011). Optimum design of steel frames using Cuckoo Search Algorithm with Levy Flight. *The Structural Design of Tall and Special Buildings,* 22(13), 1023-1036.

[4] Civicioglu, P., & Besdok, E. (2011). A Conceptual Comparison of the Cuckoo-search, Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony Algorithms. *Artificial Intelligence Review.*

[5] Gamdomi, A.H., Yang, X.S., & Alavi, A.H. (2011). Cuckoo Search Algorithm: A metaheuristic Approach to Solve Structural Optimization Problems. *Engineering with Computers*. 29(1), 17-35.

[6] Liefooghe, A., Basseur, M., Humeau, J., Jourdan, L., & Talbi, E. (2012). On optimizing a bi-objective flowshop scheduling problem in an uncertain enviroment. *Computers & Mathematics with Applications.* 64(12), 3747-3762.

[7] Talaei, A., Najafi, E., & Shahsavavariour. (2013). Minimizing Makespan in Flowshop Scheduling Problem using Combination of Particle Swarm Optimization and Simulated Annealing Algorithms. *Journal of Applied Science and Engineering Management*. 1(2), 18-27.

[8] Baker, K.R., & Trietsch, D. (2009). *Principles of Sequencing and Scheduling*. John Wiley & Sons. New York.

[9] Quaraab, A., Ahiod, B., & Yang, X.S. (2013). Improved and Discrete Cuckoo Search Algorithm for Solving the Traveling Salesman Problem, in : yang X.S. (eds) Cukoo Search and Firefly Algorithms. *Studies in Computational Intelligence.*

[10] Lin, Y., Zhang, Cuiping, & Liang, Z. (2016). Cuckoo Search Algorithm with Hybrid Factor using Dimenetional Distance. *Mathematical Problems in Engineering.* Article ID 4839763.

[11] Mantegna, R.N. (1994). Fast, Accurate Algorithm for Numerical Simulation of Levy Stable Processes. *Phys. Rev. E 49*, 4677-4683.

[12] Dubuc, S. (1990). An Approximation of the Gamma Function. *Journal of Mathematical Analysis and Applications.*

[13] Ozdagoglu, G. (2008). A Simulated Annealing Application on Flow Shop Sequencing Problem : A Comperative Case Study. University Islemete, 357-377.