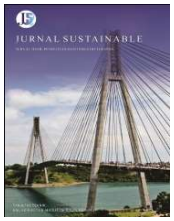


Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan
Vol. 06, No. 01, hal. 9-15, Mei 2017
Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan
ISSN 2087-5347

Perbandingan Teknik Pengkodean Langsung dan Tidak Langsung Pada Kasus Penjadwalan *Jobshop*

Muhamad Radzi Rathomi*
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Maritim Raja Ali Haji
Jl. Politeknik Senggarang, Tanjungpinang 29100
*Corresponding Author: radzi@umrah.ac.id

Abstract— Development of technology help human life in problem solving. Scheduling is a one of the problem which could be solved with it. In scheduling research, jobshop case is frequently used to test the scheduling problem solving algorithm. This study provide the comparison between direct encoding and indirect encoding approach. These approach are choices in jobshop secheduling problem research. The apparent differences of these approach are in used technique. Genetic algorithm is used as the testing algorithm. The Cases which will be used are the common cases from OR-Lib. The testing is done by looking the makespan, processing time, and objective value transformation. Testing result shows the direct encoding approach found more small makespan. Whereas indirect encoding approach can found optimal makespan in running with large number of generation.
Keywords— Comparison, Encoding, Scheduling, Genetic Algorithm.

Intisari—Perkembangan teknologi sangat membantu kehidupan manusia dalam menyelesaikan masalah-masalah rutinitas sehari-hari. Penjadwalan merupakan salah satu masalah yang dapat diselesaikan dengan teknologi. Pada penelitian tentang penjadwalan, kasus jobshop selalu digunakan untuk menguji algoritma penyelesaian masalah penjadwalan. Penelitian ini memberikan perbandingan antara pendekatan pengkodean langsung dan pendekatan pengkodean tidak langsung yang menjadi pilihan pada penelitian tentang masalah penjadwalan jobshop. Perbedaan yang terlihat pada kedua pendekatan tersebut adalah pada teknik-teknik yang digunakan, dimana pendekatan tidak langsung menggunakan teknik-teknik tembahan untuk menyusun jadwal, sedangkan pengkodean langsung menggunakan urutan solusi untuk menyusun jadwal. Algoritma genetika akan digunakan sebagai algoritma yang akan melakukan pencarian solusi dari masalah tersebut. Kasus-kasus yang digunakan untuk pengujian adalah kasus-kasus umum yang diperoleh dari OR-Lib. Pengujian dilakukan dengan memperhatikan nilai *makespan*, waktu pemrosesan dan perubahan nilai objektif pada beberapa generasi. Hasil pengujian menggunakan kasus-kasus umum memperlihatkan bahwa pendekatan pengkodean langsung lebih banyak menemukan *makespan* yang kecil. Sedangkan pendekatan pengkodean tidak langsung dapat menemukan *makespan* yang lebih optimal dengan menggunakan jumlah generasi yang banyak.
Kata kunci— Perbandingan, Pengkodean, Penjadwalan, Algoritma Genetika.

 ISSN 2087-5347 (Print)

I. PENDAHULUAN

Penjadwalan merupakan masalah yang paling sering muncul dalam kehidupan sehari-hari. Penjadwalan termasuk kedalam jenis masalah optimasi kombinatorial. Penjadwalan mudah diselesaikan dengan cara mengkombinasikan agenda-agenda yang akan dilakukan sehingga membentuk sebuah jadwal. Namun, penjadwalan akan menjadi semakin rumit jika terdapat batasan-batasan dalam penyusunan agenda-agenda tersebut.

JobShop Scheduling Problem (JSSP) adalah salah satu contoh masalah penjadwalan. JSSP merupakan masalah penjadwalan buatan yang memiliki banyak batasan-batasan. Pada JSSP, terdapat beberapa pekerjaan dimana setiap pekerjaan memiliki operasi-operasi yang jumlahnya sama untuk setiap pekerjaan. Operasi-operasi pada setiap pekerjaan harus dikerjakan oleh mesin yang berbeda-beda. Jumlah mesin yang digunakan untuk mengerjakan setiap operasi hanya sebanyak jumlah operasi pada satu pekerjaan. Sehingga setiap operasi harus bergantian menggunakan mesin-mesin tersebut[1]. Penggunaan mesin harus dijadwalkan agar memperoleh waktu tersingkat dalam menyelesaikan semua pekerjaan.

Algoritma-algoritma evolusi seperti algoritma genetika bisa digunakan untuk menyelesaikan masalah penjadwalan, khususnya JSSP. Ada dua pendekatan dalam melakukan pengkodean masalah JSSP. Pengkodean langsung (Direct) dan pengkodean tidak langsung (Indirect). Peneliti dapat menggunakan salah satu dari pendekatan pengkodean ini. Berikut adalah penelitian-penelitian yang dilakukan untuk menyelesaikan masalah JSSP.

Zhang *et al*. [2] menyelesaikan masalah JSSP menggunakan algoritma koloni semut. Berdasarkan teknik penyelesaiannya, penelitian ini menggunakan pendekatan langsung, Dimana node yang digunakan pada jalur semut disimbolkan sebagai operasi-operasi pada pekerjaan.

Yusof *et al*. [3] menyelesaikan masalah JSSP menggunakan pendekatan tidak langsung yang diproses menggunakan algoritma genetika mikro paralel hybrid. Penelitian ini dapat memperoleh waktu penyelesaian semua pekerjaan JSSP lebih singkat dibangdingkan dengan waktu tercepat yang pernah diperoleh sebelumnya.

Zhang [4] menggunakan algoritma genetika untuk mengoptimasikan hasil pencarian dari algoritma yang diusulkannya. Pendekatan pengkodean yang digunakan adalah pendekatan langsung. Zhou et al. [5] menyelesaikan masalah JSSP menggunakan algoritma koloni lebah dua tahap. Pengkodean yang digunakan adalah pengkodean langsung. Jamili [6] mencoba membuat model matematika untuk menyelesaikan masalah JSSP, model tersebut diimplementasikan menggunakan algoritma optimasi gerombolan burung. pendekatan penyelesain yang digunakan adalah langsung. Kurdi [7] menyelesaikan masalah JSSP menggunakan algoritma genetika paralel dengan model island. Pengkodean yang digunakan adalah pengkodean tidak langsung dimana kromosom solusi dijadikan rujukan teknik penyusunan jadwal. Salido *et al*. [8] melakukan penelitian tentang efisiensi energi pada JSSP. Pendekatan yang digunakan pada penelitian tersebut adalah pendekatan langsung.

Kudakci dan Kulak [9] melakukan penelitian untuk menyelesaikan masalah JSSP menggunakan algoritma genetika hybrid. Pendekatan pengkodean yang digunakan adalah pendekatan tidak langsung. Hasil perbandingan dengan solusi terbaik yang pernah diperoleh menunjukkan penelitian ini memberikan *makespan* yang lebih baik. Muthiah dan Rajkumar [10] mengusulkan algoritma pengembangan dari algoritma genetika untuk menyelesaikan masalah JSSP. Hasil yang diperoleh dari penelitian tersebut dapat dibandingkan dengan masalah umum JSSP yang ada, dan menunjukkan hasil yang lebih baik dari hasil yang pernah ada. Wu *et al*. [11] menyelesaikan masalah JSSP menggunakan algoritma optimasi koloni semut hybrid. Penelitian tersebut menggunakan pendekatan pengkodean langsung dimana masalah diselesaikan dengan graf disjungtif tiga dimensi.

Berdasarkan penelitian-penelitian tersebut, peneliti dapat memilih pendekatan yang bisa

 ISSN 2087-5347 (Print)

digunakan untuk menyelesaikan masalah JSSP. Pada penelitian ini, akan dicoba dua buah algoritma genetika, dimana satu algoritma genetika menggunakan pengkodean langsung, dan satunya lagi menggunakan pengkodean tidak langsung. Hasil dari kedua pengkodean tersebut akan dibandingkan berdasarkan perubahan nilai perubahan nilai objektif dan waktu pemrosesan yang dibutuhkan berdasarkan parameter yang akan dibuat sama.

II. METODE PENELITIAN

Penelitian ini akan menggunakan contoh kasus seperti yang ditunjukkan oleh Tabel 1. Pada tabel tersebut terdapat empat pekerjaan, setiap pekerjaan memiliki tiga operasi yang harus dikerjakan. Masing-masing operasi akan dikerjakan oleh mesin-mesin yang berbeda. Setiap operasi memiliki waktu pemrosesan P_{ijk}, dimana i adalah nomor pekerjaan, j adalah nomor operasi dan k adalah nomor mesin yang akan memproses operasi. Setiap operasi pada satu pekerjaan, harus dikerjakan berurutan berdasarkan nomor operasi, sehingga satu operasi harus menunggu operasi pendahulunya selesai diproses, walaupun mesin yang akan memprosesnya sudah tidak mengerjakan operasi yang lain.

Tabel 1. Contoh kasus JSSP

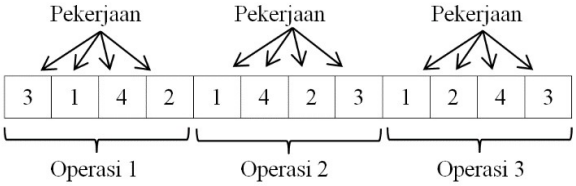
Operasi	P_{ijk}			Mesin		
	1	**2**	**3**	**1**	**2**	**3**
Pekerjaan 1	2	2	12	M_1	M_3	M_2
Pekerjaan 2	2	3	3	M_2	M_1	M_3
Pekerjaan 3	3	1	5	M_1	M_2	M_3
Pekerjaan 4	2	2	3	M_3	M_2	M_1

Berikut ini akan dijelaskan secara detail teknik pengkodean langsung dan pengkodean tidak langsung. Kemudian akan dijelaskan operator-operator yang perlu disesuaikan pada algoritma genetika. Cara paling mudah untuk memenuhi persyaratan format penulisan adalah dengan menggunakan dokumen ini sebagai template. Kemudian ketikkan teks Anda ke dalamnya.

A. Pendekatan Pengkodean Langsung (Direct)

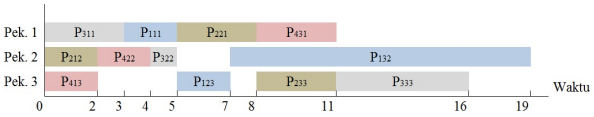
Pada pengkodean dengan pendekatan langsung, kromosom langsung menentukan urutan dari operasi-operasi pekerjaan yang akan diproses oleh mesin. Untuk menyelesaikan masalah JSSP dengan pendekatan ini, maka jumlah *gen* pada kromoson yang akan dibangun adalah sebanyak jumlah pekerjaan dikalikan dengan jumlah operasi pada setiap pekerjaan [1]. Jika masalah yang akan diselesaikan adalah masalah yang ada pada Tabel 1, maka jumlah *gen* pada kromosom adalah sebanyak empat pekerjaan dikalikan dengan tiga operasi, yaitu dua belas.

Setiap *gen* pada kromosom berisi nomor pekerjaan, sedangkan *phenotype* pada sebuah kromosom menyatakan operasi yang berurut. Setiap *gen* dalam satu *phenotype* berisi nomor urut pekerjaan, sehingga nilai-nilai *gen* tersebut harus unik membentuk permutasi. Urutan *gen* menyatakan operasi pertama pada pekerjaan yang mana harus dikerjakan oleh mesin yang bertugas. Gambar 1 menunjukkan susunan *gen* tersebut.



Gambar 1. Komponen kromosom pada pengkodean langsung

Jika kromosom tersebut dimasukkan kedalam Gantt Chart, maka hasilnya akan menjadi seperti yang ditunjukkan oleh Gambar 2. Gambar tersebut menunjukkan bahwa waktu terlama untuk menyelesaikan semua pekerjaan adalah 19.



Gambar 2. Hasil penyusunan berdasarkan pengkodean langsung

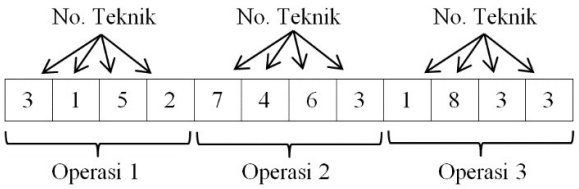
 ISSN 2087-5347 (Print)

B. Pendekatan Pengkodean Tidak Langsung (Indirect)

Pada pengkodean tidak langsung, penyusunan operasi-operasi tidak dilakukan berdasarkan susunan *gen* pada kromosom, tetapi penyusunan dilakukan oleh teknik yang dipilih dari nilai *gen* yang ada pada kromosom. Teknik-teknik tersebut adalah sebagai berikut:

1) SPT (Shortest Processing Time): Memilih operasi dengan waktu pemrosesan tersingkat.
2) LPT (Longest Processing Time): Memilih operasi dengan waktu pemrosesan terlama.
3) MWR (Most Work Remaining): Memilih operasi pada pekerjaan yang memiliki total waktu pemrosesan paling lama.
4) LWR (Least Work Remaining): Memilih operasi pada pekerjaan yang memiliki total waktu pemrosesan paling sedikit.
5) MOR (Most Operations Remaining): Memilih operasi pada pekerjaan yang paling banyak memiliki sisa operasi.
6) LOR (Least Operations Remaining): Memilih operasi pada pekerjaan yang paling sedikit memiliki sisa operasi.
7) FCFS (First Come First Served): Memilih operasi pertama yang menunggu pada mesin.
8) R (Random). Memilih operasi secara random (memerlukan penanganan khusus).

Pemilihan teknik-teknik yang akan digunakan dilakukan berdasarkan nilai *gen* yang terpilih. Nilai *gen* dibuat berdasarkan jumlah teknik yang akan digunakan. Jika delapan teknik tersebut akan digunakan, maka nilai gen minimal adalah satu dan nilai *gen* maksimal adalah delapan. Jumlah *gen* pada satu kromosom adalah sama dengan pengkodean langsung, yaitu jumlah operasi dikalikan dengan jumlah pekerjaan. Gambar 3. menunjukkan susunan *gen* untuk pengkodenan tidak langsung.



Gambar 3. Komponen kromosom pada pengkodean tidak langsung

Algoritma pendekatan pengkodean tidak langsung ini sudah pasti lebih panjang dibandingkan algoritma pendekatan pengkodean langsung. Karena, pendekatan tidak langsung harus memilih terlebih dahulu teknik pemilihan mana yang akan digunakan. Sedangkan pendekatan pengkodean langsung, memilih operasi pekerjaan berdasarkan susunan yang ada pada kromosom secara langsung.

C. Algoritma Genetika

Algoritma Genetika yang digunakan pada penelitian ini adalah algoritma genetika sederhana. Beberapa operator akan disesuaikan dengan pendekatan pengkodean yang digunakan, berikut adalah penyesuaian operator-operator algoritma genetika:

1) Inisialisasi

Pada tahapan inisialisasi, pembangkitan kromosom-kromosom populasi harus menyesuaikan dengan pendekatan langsung yang membutuhkan susunan bilangan permutasi, maupun pendekatan tidak langsung yang mengharuskan nilai *gen* berisi nilai maksimal jumlah teknik yang digunakan.

2) Evaluasi

Tahapan evaluasi dilakukan dengan mengubah nilai objektif menjadi *fitness*. Pada penelitian ini, yang menjadi nilai objektif adalah *makespan*. Semakin kecil *makespan*, maka semakin baik susunan jadwal yang diperoleh. Oleh karena itu, *makespan* akan diubuah menjadi nilai *fitness* melalui proses Transferral menggunakan Persamaan (1).

$$f = \frac{1}{1 + (makespan)} \qquad (1)$$

3) Seleksi

 ISSN 2087-5347 (Print)

Algoritma genetika pada penelitian ini akan menggunakan seleksi turnamen[1], kelompok individu akan dibangkitkan sebanyak ukuran mating pool. Anggota kelompok individu akan dibangkitkan secara random sebanyak ukuran yang ditentukan, pada penelitian ini anggota kelompok individu akan dibangkitkan sebanyak lima anggota.

4) *Kawin* Silang

Operator kawin silang yang akan digunakan pada penelitian ini adalah *order crossover* [1]. Teknik ini digunakan agar permutasi pada *phenotype* tidak rusak khususnya pada pendekatan pengkodean langsung. Jika susunan permutasi rusak, maka akan menyebabkan rusaknya susunan jadwal sehingga akan membuat hasil pencarian algoritma genetika tidak valid. Pada pendekatan pengkodean tidak langsung, tidak memperhatikan susunan permutasi. Namun, operator kawin silang ini dapat juga diimplementasikan untuk pendekatan pengkodean tidak langsung.

5) Mutasi

Operator mutasi yang akan digunakan pada penelitian ini adalah *exchange mutation* [1]. Operator ini juga diimplementasikan agar susunan permutasi pada *phenotype* tidak rusak, khususnya pada pendekatan pengkodean langsung. Operator ini juga akan diimplementasikan pada pendekatan pengkodean tidak langsung.

III. HASIL DAN PEMBAHASAN

Proses pengujian pada penelitian ini akan dilakukan dengan memperhatikan perbedaan *makespan* yang diperoleh, waktu pemrosesan dalam menyelesaikan jumlah generasi yang ditentukan dan kecepatan dalam menemukan solusi yang optimal. Parameter-parameter yang akan digunakan oleh kedua pendekatan ini akan dibuat sama, yaitu Jumlah individu sebanyak 100, jumlah iterasi (Generasi) sebanyak 10.000, Probabilitas kawing silang (P_c) sebesar 0.7, Probabilitas mutasi (P_m) sebesar 0.01, dan jumlah anggota kelompok individu turnamen sebanyak 5. Pada pengujian ini, pendekatan pengkodean tidak langsung hanya menggunakan empat teknik penyusunan, yaitu SPT, LPT, LWR dan MWR.

Penelitian ini akan menggunakan kasus-kasus umum untuk penelitian JSSP [12], yaitu ft10 (10x10), ft20 (20x5), la02 (10x5), 1a06 (15x5), la21 (15x10), la29 (20x10), la31 (30x10), la36 (15x15), orb01 (10x10), and orb04 (10x10). Nilai di dalam kurung adalah ukuran masalah (*N*x*M*), dimana *N* adalah jumlah pekerjaan yang harus dilakukan dan *M* adalah jumlah operasi pada setiap pekerjaan.

A. Perbandingan Nilai Objektif

Pada pengujian ini, untuk mendapatkan nilai minimal, maksimal dan rata-rata, running program dilakukan sebanyak 10 kali. Hasil dari running tersebut ditunjukkan oleh Tabel 2. Berdasarkan tabel tersebut, terlihat bahwa sebagian besar nilai *makespan* yang diperoleh dari pengkodean langsung lebih baik dibandingkan dengan pengkodean tidak langsung. Pada kasus seperti ft10, ft20 dan la02 memiliki nilai *makespan* yang lebih baik pada pengkodean tidak langsung. Ini disebabkan oleh ukuran masalah pada kasus tersebut adalah kecil, sehingga jumlah generasi pada pengujian ini cukup untuk menemukan hasil seperti yang ditunjukkan pada Tabel 2.

Tabel 2. *Makespan* hasil pengujian

Kasus	Langsung			Tidak Langsung		
	Min	Max	Aver.	Min	Max	Aver.
ft10	1133	1134	1133.8	1066	1138	1112.9
ft20	1495	1522	1499.3	1370	1512	1464.7
la02	758	758	758	711	728	716
la06	927	927	927	926	946	933.3
la21	1229	1235	1232.6	1350	1448	1392.5
la29	1420	1420	1420	1638	1823	1720.9
la31	1784	1800	1787.1	2313	2587	2423.9
la36	1494	1494	1494	1624	1916	1805.6
orb01	1279	1279	1279	1236	1355	1287.1
orb04	1037	1037	1037	1105	1255	1181.6

Pengujian lebih lanjut dilakukan pada kasus dengan nilai *makespan* paling besar yaitu la31. Pengujian lanjutan ini menggunakan jumlah generasi sebanyak 100.000. Hasil pengujian menunjukkan bahwa pengkodean tidak langsung masih menemukan *makespan* yang lebih kecil rata-rata pada generasi antara 50.000 dan 80.000. Pengkodean tidak langsung untuk masalah JSSP membutuhkan generasi yang banyak untuk menemukan *makespan* yang

 ISSN 2087-5347 (Print)

optimal. Tentu saja jumlah generasi yang banyak akan menambah waktu pemrosesan.

B. Waktu Pemrosesan

Perbandingan waktu pemrosesan ini didapatkan juga dengan running program sebanyak 10 kali. Waktu minimal, maksimal dan rata-rata ditunjukkan oleh Tabel 3 berikut:

Tabel 3. Perbandingan waktu pemrosesan

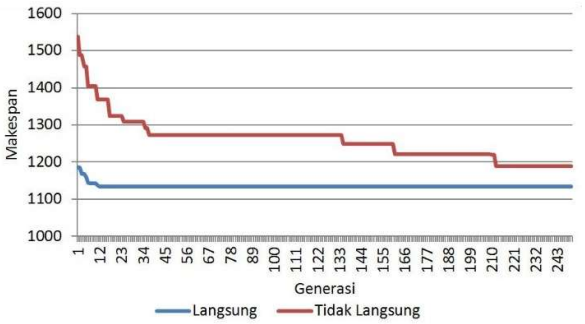
Kasus	Langsung			Tidak Langsung		
	Min	Max	Aver.	Min	Max	Aver.
ft10	24.44	24.76	24.62	36.87	37.25	37.03
ft20	22.46	22.97	22.63	46.06	48.12	46.513
la02	11.71	11.91	11.84	18.59	18.76	18.65
la06	16.76	16.97	16.89	30.28	31.26	30.549
la21	33.29	34.03	33.55	59.77	61.76	60.389
la29	45.12	45.46	45.27	86.6	90.93	89.957
la31	69.93	70.16	70.02	152.01	153.59	152.993
la36	50.37	50.64	50.50	84.41	86.62	85.694
orb01	22.99	23.49	23.14	35.53	36.4	35.88
orb04	23.1	23.46	23.21	34.68	36.08	35.218

Terlihat pada Tabel 3 bahwa rata-rata waktu pemrosesan yang dibutuhkan oleh pengkodean tidak langsung adalah lebih lama dibandingkan dengan pengkodean langsung. Hal ini disebabkan oleh karena pengkodean tidak langsung harus menentukan terlebih dahulu teknik yang akan digunakan, kemudian memilih operasi berdasarkan teknik yang terpilih. Sedangkan pada pengkodean langsung, operasi langsung dipilih berdasarkan susunan *gen* pada kromosom.

Waktu pemrosesan akan lebih lama lagi jika pengkodean tidak langsung digunakan untuk menemukan *makespan* pada kasus-kasus yang membutuhkan jumlah generasi yang lebih banyak seperti kasus la31. Oleh karena itu, pengembangan algoritma khusus seperti paralelisasi dibutuhkan untuk menjalankan pendekatan pengkodean ini agar dapat mengurangi waktu pemrosesan.

C. Kecepatan Penemuan Solusi

Pada bagian pengujian ini hanya akan menggunakan satu kasus JSSP untuk melihat perubahan nilai *makespan* pada setiap generasi. Kasus yang digunakan adalah ft10. Seperti yang ditunjukkan Gambar 4.



Gambar 4. Perubahan nilai *makespan*

Gambar 4 menunjukkan bahwa pendekatan pengkodean langsung cepat dalam menemukan *makespan* terkecil. Dapat dilihat bahwa pada generasi pertama saja *makespan* yang ditemukan sudah dibawah 1200. Kemudian antara generasi ke lima dan ke sepuluh, pengkodean langsung sudah menemukan *makespan* 1133.

Berbeda dengan pengkodean tidak langsung, pada generasi pertama saja *makespan* yang ditemukan lebih dari 1500. Namun hasil pencarian dengan pengkodean tidak langsung ini akan semakin meningkat. Seperti yang ditunjukkan oleh Gambar 4, pengkodean tidak langsung berhenti menemukan *makespan* antara generasi kelima dan kesepuluh, sedangkan pengkodean tidak langsung masih menemukan *makespan* lebih kecil pada generasi 133, 155 dan 210. Pencarian dengan pengkodean ini akan terus meningkat seperti *makespan* yang ditunjukkan oleh Tabel 2 untuk kasus ft10.

IV. KESIMPULAN

Berdasarkan hasil analisis dan pengujian yang telah dilakukan, terlihat bahwa pengkodean langsung cepat dan lebih banyak dalam menemukan *makespan* yang lebih kecil. Pengkodean tidak langsung dapat menemukan *makespan* lebih kecil pada kasus-kasus dengan ukuran yang kecil. Untuk masalah dengan ukuran yang besar pengkodean tidak langsung membutuhkan jumlah generasi yang banyak sehingga membutuhkan waktu yang lebih lama dibandingkan dengan pengkodean langsung. Hal ini disebabkan pengkodean tidak langsung harus memilih teknik yang akan digunakan pada saat memilih operasi yang akan diletakkan pada jadwal. Hal ini dapat diatasi dengan

 ISSN 2087-5347 (Print)

mengembangkan algoritma yang menggunakan pengkodean tidak langsung seperti paralelisasi.

REFERENSI

[1] Y. Xinjie and G. Mitsuo, *Introduction to Evolutionary Algorithms*. 2010.

[2] J. Zhang, X. Hu, X. Tan, J. H. Zhong, and Q. Huang, "Implementation of an Ant Colony Optimization technique for job shop scheduling problem," vol. 1, 2006.

[3] R. Yusof, M. Khalid, G. T. Hui, S. Yusof, and M. F. Othman, "Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm," *Appl. Soft Comput. J.*, vol. 11, no. 8, pp. 5782–5792, 2011.

[4] R. Zhang, "A Successive Genetic Algorithm for Solving the Job," vol. 1, pp. 613–619, 2012.

[5] K. Zhou, P. Nagaratnam, T. Jin, and C. Soon, "Expert Systems with Applications A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, 2015.

[6] A. Jamili, "Robust job shop scheduling problem : Mathematical models , exact and heuristic algorithms," *Expert Syst. Appl.*, vol. 55, pp. 341–350, 2016.

[7] M. Kurdi, "Computers & Operations Research An effective new island model genetic algorithm for job shop scheduling problem," *Comput. Oper. Res.*, vol. 67, pp. 132–142, 2016.

[8] M. A. Salido, J. Escamilla, A. Giret, and F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," pp. 1303–1314, 2016.

[9] N. Kundakcı and O. Kulak, "Computers & Industrial Engineering Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem q," vol. 96, pp. 31–51, 2016.

[10] A. Muthiah and R. Rajkumar, "A novel algorithm for solving job-shop scheduling problem," vol. 23, no. 4, pp. 610–617, 2017.

[11] U. Management, "FLEXIBLE JOB-SHOP SCHEDULING PROBLEM BASED ON," vol. 16, pp. 497–505, 2017.

[12] R. J. Mattfeld, D.C, Vaessens, *OR-Library: a set of 82 JSP test Instances*. 2017.