Jurnal Teknik Industri, Vol. 12, No. 2, Desember 2010, 119-124
ISSN 1411-2485

Perbandingan Kombinasi *Genetic Algorithm – Simulated Annealing* dengan *Particle Swarm Optimization* pada Permasalahan Tata Letak Fasilitas

Isabella Leo Setiawan[1], Herry Christian Palit[2]

Abstract: This article aims to compare the performance of combination of Genetic Algorithm-Simulated Annealing (GA-SA) with Particle Swarm Optimization (PSO) to solve facility layout problem. GA-SA in this article consist of two algorithms, GA-SA I and GA-SA II, with a different mutation rule. PSO uses fuzzy particle swarm concept to represent solution from each particle. Two criteria to analyze all algorithms performance are moment of movement and computational time. Experiments show that GA-SA II has the best performance in minimization both criteria.

Keywords: Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization, fuzzy particle swarm, facility layout problem.

Pendahuluan

Perancangan tata letak fasilitas memiliki arti penting dalam proses operasional perusahaan. Pada sistem manufaktur, kegiatan *material handling* dapat menghabiskan biaya sekitar 15-70% dari total biaya operasi (Purnomo [9]). Hal ini menunjukkan bahwa upaya penurunan biaya *material handling* merupakan salah satu cara efektif untuk menekan biaya produksi. Jadi tujuan dari perancangan tata letak adalah meminimasi biaya perpindahan material, yang besarnya diwakili dengan total momen perpindahan. Sehubungan dengan permasalahan tata letak fasilitas ini, telah dikembangkan *exact* model (Castillo [1]; Nordin *et al.* [8]) untuk mendapatkan solusi yang optimal. Namun pencarian solusi menjadi lama apabila jumlah departemennya banyak dan bentuk tiap departemen berbeda (*unequal areas*). Oleh karena itu, banyak peneliti mengembangkan model-model heuristik yang menghasilkan solusi mendekati optimal dalam waktu yang lebih singkat.

Beberapa algoritma heuristik yang diaplikasikan pada permasalahan tata letak fasilitas, antara lain *Genetic Algorithm* (Chutima [2]; El-Baz [3]), *Simulated Annealing* (McKendall *et al.* [7]). Nordin *et al.* [8] mengusulkan penggunaan algoritma kombinasi dari *Genetic Algorithm* (GA) dan *Simulated Annealing* (SA) dalam permasalahan tata letak fasilitas yang berbeda bentuk.

[1,2] Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Kristen Petra. Jl. Siwalankerto 121-131, Surabaya 60236, Indonesia, Email: herry@petra.ac.id,

Naskah masuk 2 Agustus 2010; revisi1 30 September 2010; Diterima untuk dipublikasikan 10 November 2010.

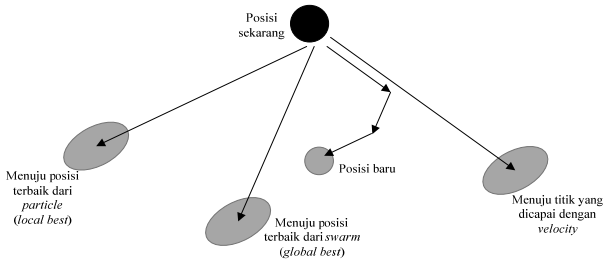
Adanya kombinasi ini dapat meningkatkan performansi dari algoritma dengan menonjolkan kelebihan dan menutupi kelemahan dari masing-masing algoritma. Selain kedua algoritma tersebut, dikenal pula *Particle Swarm Optimization Algorithm* (PSO) yang mulai pertama kali diperkenalkan pada tahun 1995 (Kennedy dan Eberhart [5]). Pada algoritma PSO, anggota populasi disebut *particle*. Semua anggota populasi tetap dipertahankan selama proses pencarian, sehingga setiap anggota dapat berbagi informasi untuk mencari posisi terbaik (Uysal dan Bulkan [11]). *Particle* yang kurang *fit* pada suatu iterasi memiliki kemungkinan untuk menjadi paling *fit* pada iterasi berikutnya. PSO tidak mengeliminasi anggota yang kurang *fit* selama proses pencarian berlangsung, sehingga solusi tidak mudah terjebak dalam *local optimum*. Liu *et al.* [6] menyatakan bahwa posisi dan *velocity* suatu *particle* pada PSO dapat direpresentasikan ke dalam *fuzzy matrix*, sehingga didapatkan solusi posisi yang layak. Pada artikel ini, GA-SA dibandingkan performansinya dengan PSO yang menggunakan konsep *fuzzy matrix* dalam menyelesaikan masalah tata letak fasilitas.

Metode Penelitian

Sebelum penjelasan perancangan dari algoritma GA-SA dan PSO, maka akan dijelaskan terlebih dahulu beberapa konsep dasar yang digunakan dalam perancangan algoritma tersebut.

Particle Swarm Optimization (PSO)

PSO merupakan algoritma yang didasarkan pada interaksi sosial dan komunikasi makhluk hidup. Dalam PSO, setiap anggota disebut dengan *particle* yang bergerak dalam sebuah ruang pencarian



Gambar 1. Prinsip perpindahan *particle*

secara multidimensional. Solusi awal dalam algoritma ini dibangkitkan secara acak, seperti halnya dengan GA. Akan tetapi, PSO tidak memiliki operator rekombinasi seperti *crossover* dan mutasi. Setiap *particle* memiliki posisi dan *velocity* yang dinyatakan dalam vektor. Pencarian solusi pada PSO direpresentasikan dengan perpindahan posisi *particle*. Pada setiap iterasi, setiap *particle* memperbaharui *velocity* dan posisinya menuju posisi terbaiknya. Pada saat yang sama, terjadi pertukaran informasi posisi terbaik di antara seluruh kumpulan *particle*. Pada dasarnya, *particle* tidak dapat berpindah secara tiba-tiba, dan bergerak menuju posisi terbaik berdasarkan pengalaman pribadinya maupun pengalaman dari seluruh *particle*. Prinsip perpindahan posisi *particle* dapat diilustrasikan pada Gambar 1.

Posisi dan *velocity particle* dinyatakan dalam vektor. Secara matematis, posisi *particle* untuk setiap i, diperbaharui dengan persamaan berikut: (Hakim *et al.* [4])

$$x_{k+1}^i = x_k^i + v_{k+1}^i \quad (1)$$

Shi dan Eberhart [10] memperkenalkan *inertia weight* (w) dalam perhitungan *velocity*, yang dapat meningkatkan performansi PSO dalam beberapa aplikasi. *Velocity* untuk *particle* i pada waktu $k + 1$ diperbaharui dengan perhitungan sebagai berikut: (Hakim *et al.* [4])

$$v_{k+1}^i = w_k v_k^i + c_1 r_1 \left(p_k^i - x_k^i\right) + c_2 r_2 (p_k^g - x_k^i) \quad (2)$$

dimana c_1 dan c_2 merupakan bilangan konstan positif, r_1 dan r_2 merupakan bilangan *random* berdistribusi *uniform* antara 0 sampai 1, p_k^i merupakan posisi terbaik dari *particle* i pada waktu k, dan p_k^g merupakan posisi terbaik global dari keseluruhan kumpulan *particle* pada waktu k. Pada penentuan parameter w, didapatkan bahwa w yang bernilai tinggi di awal dan bernilai rendah di akhir memberikan hasil yang lebih baik. Uysal dan Bulkan [11] menentukan nilai w mulai dari 0,9 dan menurun secara linear hingga 0,4; dengan *decrement factor* α. Secara matematis, w pada waktu k dihitung sebagai berikut:

$$w^k = w^{k-1} \times \alpha \quad (3)$$

Velocity dibatasi dengan nilai maksimum yaitu v_{max}. Kemampuan eksplorasi *particle* dikontrol oleh nilai v_{max}. Penetapan v_{max} sangat mempengaruhi solusi yang diperoleh. Jika v_{max} terlalu kecil atau terlalu besar, *particle* tidak dapat mengeksplorasi ruang pencarian dengan baik dan akan mudah terjebak dalam *local optimum*.

Fuzzy Particle Swarm

Pada permasalahan tata letak fasilitas, operasi vektor posisi dan velocity pada persamaan (1) dan (2) tidak dapat langsung digunakan untuk mencari solusi. Solusi berupa urutan departemen harus dinyatakan dalam suatu bentuk yang mudah dioperasikan. *Fuzzy particle swarm* merupakan metode yang menggunakan konsep *fuzzy* untuk menyatakan posisi dan *velocity* suatu *particle* pada PSO ke dalam bentuk matriks yang disebut *fuzzy matrix* (Liu *et al.* [6]). Matriks posisi (*X)* dan *velocity* (*V)* berupa matriks bujur sangkar berukuran $n \times n$, dimana n adalah jumlah departemen. Elemen x_{ij} dan v_{ij} merepresentasikan keanggotaan dari posisi dan *velocity* departemen j di lokasi i. Elemen-elemen dalam matriks X harus memenuhi kendala sebagai berikut:

$$x_{ij} \in \{0,1); \qquad i = 1,2,\dots,n; j = 1,2,\dots,n$$
$$\sum_{i=1}^{n} x_{ij} = 1; \qquad i = 1,2,\dots,n; j = 1,2,\dots,n$$
$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1,2,\dots,n; j = 1,2,\dots,n$$

Velocity dan posisi yang berupa matriks tersebut diperbaharui dengan mengganti persamaan (1) dan (2) menjadi persamaan (4) dan (5).

$$V_{k+1}^i = w_k \otimes V_k^i \oplus (c_1 r_1) \otimes \left(P_k^i \ominus X_k^i\right) \oplus (c_2 r_2) \otimes (P_k^g \ominus X_k^i) \quad (4)$$

$$X_{k+1}^i = X_k \oplus V_{k+1}^i \quad (5)$$

dimana P_k^i merupakan matriks posisi terbaik dari particle i pada waktu k dan P_k^g merupakan matriks posisi terbaik global dari keseluruhan kumpulan particle pada waktu k.

Setelah diperbaharui, matriks posisi mungkin tidak memenuhi ketiga kendala yang diharuskan dalam matriks posisi X. Hal yang harus dilakukan adalah membuat setiap elemen yang bernilai negatif dalam matriks menjadi nol. Jika semua elemen pada satu kolom bernilai nol, elemen-elemen tersebut harus diganti dengan bilangan *random* dalam interval [0,1]. Selanjutnya, dilakukan transformasi pada matriks X dengan membagi setiap elemen dengan jumlah seluruh elemen di kolom yang bersangkutan yang disebut sebagai matriks X_{normal}. Langkah

terakhir, matriks X_{normal} perlu disesuaikan agar menjadi solusi yang layak. Caranya adalah memilih elemen yang bernilai maksimum dalam elemen matriks, kemudian mengganti nilai elemen tersebut dengan "1", sedangkan setiap angka dalam baris dan kolom yang sama diganti dengan "0". Demikian seterusnya hingga semua baris dan kolom diproses.

Fungsi Obyektif

Fungsi obyektif dalam perancangan tata letak digunakan untuk mengevaluasi *layout* sebagai ukuran yang bersifat kuantitatif. Tujuan dari perancangan tata letak adalah meminimasi biaya perpindahan material, yang besarnya diwakili dengan total momen perpindahan. Jarak antar departemen dihitung dengan metode *rectilinier*. Momen perpindahan dihitung dengan menjumlahkan hasil kali antara jarak dan frekuensi perpindahan, yang dirumuskan sebagai:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{ij} \; ; \; i \neq j \qquad (6)$$

dimana:

Z : total momen perpindahan
f_{ij} : frekuensi perpindahan dari departemen i ke departemen j
d_{ij} : jarak dari departemen i ke departemen j
n : jumlah departemen

Perancangan Algoritma GA-SA dan PSO

Menurut Nordin *et al.* [8] masalah tata letak fasilitas yang berbeda ukuran dapat dimodelkan dengan membagi area menjadi sel-sel persegi berukuran sama. Setiap departemen akan menempati sel persegi dalam jumlah yang sesuai dengan ukurannya. Gambar 2 menunjukkan contoh area yang telah dimodelkan dengan sel-sel persegi berukuran sama. Angka dalam kurung yang berada pada setiap sel melambangkan departemen. Pada Gambar 2, tampak bahwa departemen 1 menempati sel A, D, dan E, sedangkan departemen 2 menem-pati sel B dan C, demikian juga dengan departemen yang lain.

A(1)	B(2)	C(2)
D(1)	E(1)	F(3)
G(4)	H(5)	I(3)

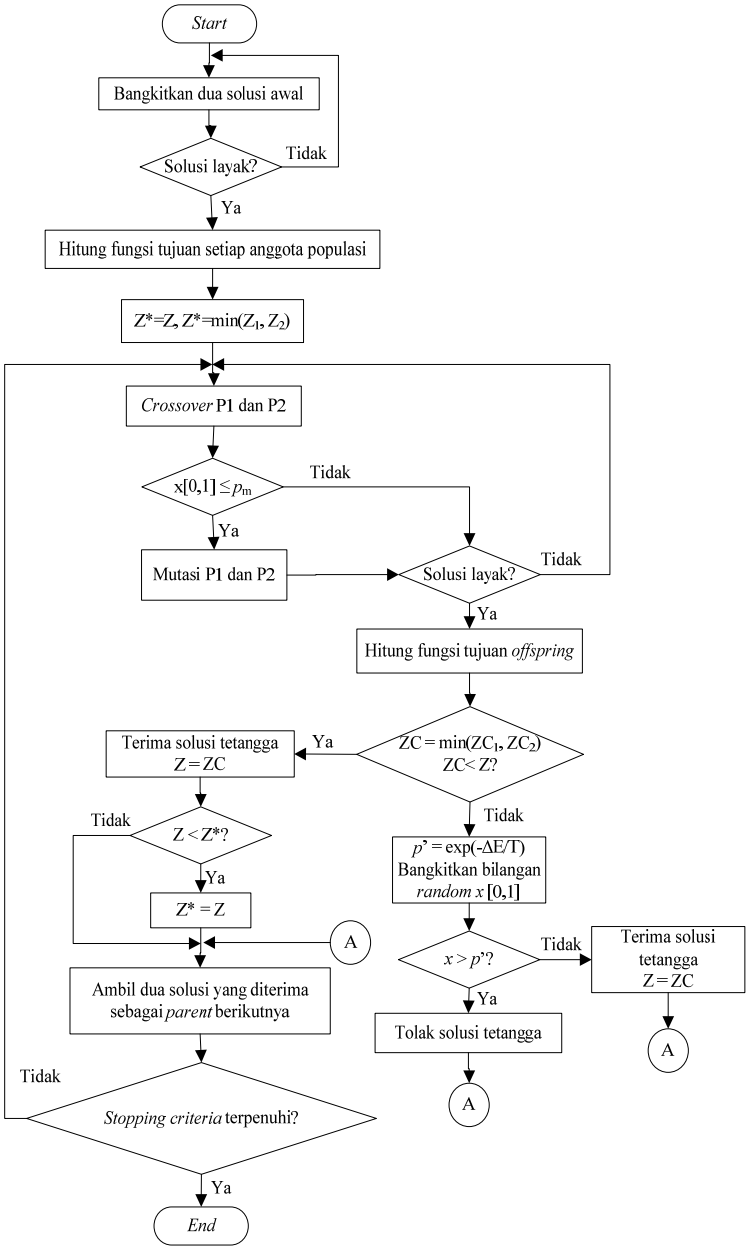
Gambar 2. Contoh pemodelan area tata letak fasilitas

Atas dasar pemikiran tersebut, maka dirancanglah dua Algoritma GA-SA, yaitu GA-SA I dan GA-SA II. Perbedaan antara GA-SA I dan GA-SA II terletak pada aturan mutasi. GA-SA I melakukan mutasi jika kedua *parent* sama (Nordin *et al.* [8]), sedangkan GA-SA II melakukan mutasi jika bilangan *random* dalam interval [0,1] yang dibangkitkan lebih kecil dari probabilitas mutasi. Metode *Crossover* yang digunakan pada GA-SA I dan GA-SA II, yaitu metode *Partially Mapped Crossover* (PMX). Metode ini diawali dengan membangkitkan dua bilangan random sebagai *cutpoint* kromosom. Bagian kromosom yang berada diantara kedua *cutpoint* tersebut ditukar dan selanjutnya dibuat pemetaan berdasarkan bagian kromosom yang ditukar tadi.

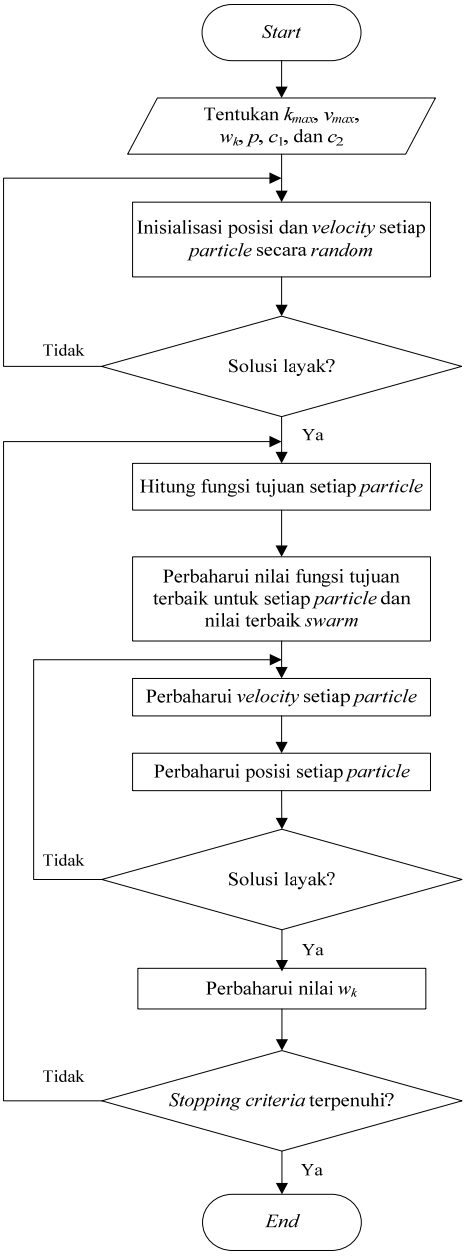
Gambar 3 menunjukkan prosedur algoritma GA-SA II, sedangkan prosedur GA-SA I tidak ditampilkan, oleh karena perbedaannya hanya terletak pada proses mutasinya, dimana terjadi ketika $P_1 = P_2$. Pada algoritma GA-SA II, solusi awal dibangkitkan secara random. Jumlah anggota populasi adalah 2, sama halnya dengan GA-SA I yang menggunakan metode usulan Nordin *et al.* [8]. Solusi dikatakan layak apabila sel-sel persegi dari satu departemen tidak terpisah satu sama lain. Jika solusi layak, maka dapat dilakukan perhitungan fungsi tujuan *Z*. *Stopping criteria* dalam algoritma terpenuhi jika jumlah iterasi mencapai 400.

Algoritma PSO dalam penelitian ini menggunakan konsep *fuzzy particle swarm*. Langkah awal dalam algoritma PSO adalah menentukan parameter-parameter yang akan mempengaruhi perhitungan algoritma, antara lain jumlah iterasi maksimum k_{max}, nilai maksimum *velocity* v_{max}, *inertia weight* w_k, jumlah *particle p*, serta nilai konstan positif c_1 dan c_2. Langkah selanjutnya adalah membangun posisi x_k^i dan *velocity* v_k^i setiap *particle* secara *random*. Posisi dan *velocity particle* dinyatakan dalam matriks X_k^i dan V_k^i. Setiap elemen dalamV_k^i tidak boleh melebih interval [$-v_{max}$, v_{max}].

Setelah posisi dan *velocity* setiap *particle* dibangun, dilakukan perhitungan fungsi tujuan dari masing-masing *particle*, dilambangkan f_k^i. Pada saat $k = 0$, maka nilai fungsi tujuan terbaik dari masing-masing *particle*, dilambangkan f_{best}^i, dimana nilainya sama denganf_k^i. Untuk nilai fungsi tujuan terbaik dari *swarm*, dilambangkan f_{best}^g, dimana nilainya sama dengan nilai f_{best}^i terkecil. Langkah selanjutnya, *velocity* dan posisi setiap *particle* diperbaharui dan kemudian matriks posisi X_k^i dilakukan normalisasi. Nilai w_k juga diperbaharui, caranya dengan mengalikan w_k dengan *decrement factor* α. Setelah melalui tahap ini, nilai k ditambah 1 dan langkah algoritma kembali pada perhitungan fungsi tujuan setiap *particle*. Siklus ini terus diulang hingga *stopping criteria* terpenuhi. *Stopping criteria* dalam algoritma ini dikatakan tercapai jika iterasi



Gambar 3. Prosedur algoritma GA-SA II



Gambar 4. Prosedur algoritma PSO

mencapai nilai yang ditentukan, yaitu 400. Jumlah *particle* (*p*) ditetapkan sebesar 5. Prosedur algoritma PSO dijelaskan lebih detail dalam *flowchart* yang ditunjukkan pada Gambar 4.

18	18	13	12	12	12	12	12	12	5	5	5	5	5	16	16
18	18	13	12	12	12	12	12	12	5	5	5	5	5	1	1
18	18	13	13	12	12	12	12	12	5	5	5	5	5	1	1
18	18	13	13	12	12	12	12	12	5	5	2	2	5	1	1
14	17	17	17	10	4	4	4	4	4	4	4	4	4	1	1
14	17	17	17	17	7	7	7	3	3	3	3	3	3	1	1
14	17	17	17	17	7	7	7	3	3	3	3	3	3	1	1
14	17	17	17	17	7	7	7	3	3	3	3	3	3	1	1
14	17	17	17	17	7	7	7	6	6	11	8	15	9	1	1

Gambar 5. *Initial layout* perusahaan

Hasil dan Pembahasan

Untuk membandingkan performansi dari ketiga algoritma tersebut, maka diambil studi kasus di sebuah perusahaan keramik sebagai obyek penelitian yang menghasilkan tiga jenis produk, yaitu guci, vas, dan roster. Pengolahan data awal menghasilkan momen perpindahan *initial layout* perusahaan sebesar 73.553,79. Gambar 5 menunjukkan penggambaran initial layout perusahaan.

Salah satu faktor yang mempengaruhi solusi dari perancangan algoritma adalah parameter yang digunakan. Pemilihan nilai parameter yang kurang tepat akan membuat performansi algoritma tidak sebaik yang diharapkan. Oleh karena itu, pada

pengolahan data terdapat beberapa parameter yang ditentukan melalui serangkaian percobaan.

Parameter awal yang ditetapkan dalam algoritma GA-SA adalah temperatur awal dan temperatur minimum. Temperatur minimum ditentukan sebesar 0,01 dan temperatur awal adalah -0,1 f_0 / ln (0,25), di mana f_0 adalah fungsi tujuan *initial solution*. Parameter awal untuk PSO antara lain p sebesar 5, c_1 dan c_2 sebesar 1,49, dan w ditentukan bernilai 0,9 dan menurun hingga 0,1 dengan α sebesar 0,975. Parameter yang akan ditentukan dengan percobaan adalah *cooling ratio* untuk GA-SA I (0,9 ; 0,95), *cooling ratio* (0,9 ; 0,95) dan probablitas mutasi untuk GA-SA II (0,1 ; 0,3 ; 0,5 ; 0,7 ; 0,9), dan v_{max} (4 dan 5) untuk PSO. Percobaan dilakukan dalam jumlah iterasi 400 dan replikasi sebanyak sepuluh kali. Berdasarkan hasil percobaan, parameter yang dipilih untuk GA-SA I adalah *cooling ratio* sebesar 0,9, sedangkan parameter untuk GA-SA II adalah probabilitas mutasi sebesar 0,7 dan *cooling ratio* sebesar 0,95. Untuk PSO, nilai parameter v_{max} yang dipilih adalah 5.

Berdasarkan hasil perhitungan menggunakan *platform* program *visual basic*, didapatkan nilai momen terkecil, rata-rata momen, dan rata-rata *computational time* setiap algoritma seperti terlihat pada Tabel 1. Hasil pengujian *two-sample t test* dengan *confidence interval* 95%, diperoleh bahwa nilai momen GA-SA I, GA-SA II, dan PSO berbeda secara signifikan. Nilai momen terkecil dicapai oleh algoritma GA-SA II.

Perbedaan hasil antara GA-SA I dan GA-SA II menunjukkan bahwa mutasi berpengaruh terhadap nilai momen yang diperoleh. Pada GA-SA I, kemungkinan terjadinya mutasi sangat kecil, karena mutasi hanya dilakukan jika kedua *parent* sama. Mutasi dapat mengurangi kemungkinan solusi terjebak dalam *local optimum*, sehingga GA-SA II dapat menghasilkan nilai momen lebih kecil dari GA-SA I. Algoritma PSO menghasilkan nilai momen yang lebih besar dari GA-SA II, tetapi masih lebih kecil dari GA-SA I. Pada percobaan yang telah dilakukan, algoritma GA-SA I memiliki performansi terburuk dalam meminimumkan momen.

Berdasarkan segi waktu, maka selisih *computational time* antar algoritma ini tidak menghasilkan sesuatu yang berbeda bagi perusahaan. *Layout* yang telah dihasilkan dari GA-SA II sebagai *layout* terbaik masih memerlukan *adjusment* berdasarkan masukan dari perusahaan, agar bentuk *layout* departemen lebih beraturan dan dapat direalisasikan. Hasil *layout* yang telah mengalami *adjustment* ditunjukkan pada Gambar 6 dengan nilai momen sebesar 56.364,9. Warna abu-abu pada *layout*

Tabel 1. Perbandingan Momen dan *Computational Time* Algoritma

Algoritma	Momen		Rata-rata CT (detik)
	Rata-rata	Terkecil	
GA-SA I	65.705,92	57.623,40	8,47
GA-SA II	**50.061,07**	**46.653,08**	8,54
PSO	60.782,92	53.150,28	**8,04**

18	18	15	12	12	12	12	12	12	12	12	12	12	12	16	16
18	18	13	12	12	12	12	12	12	12	12	12	12	12	1	1
18	18	13	13	13	9	8	11	7	7	7	7	7	7	1	1
18	18	13	13	6	6	7	7	7	7	7	2	2	7	1	1
14	17	17	17	10	5	5	5	5	5	5	5	5	5	1	1
14	17	17	17	17	5	5	5	3	3	3	3	3	3	1	1
14	17	17	17	17	5	5	5	3	3	3	3	3	3	1	1
14	17	17	17	17	5	5	5	3	3	3	3	3	3	1	1
14	17	17	17	17	4	4	4	4	4	4	4	4	4	1	1

Gambar 6. *Layout* setelah *adjustment*

menunjukkan area *fixed department* sesuai permintaan perusahaan. Hal ini berarti terjadi penurunan momen perpindahan sebesar 23,37% dibandingkan momen *initial layout* perusahaan. Perbedaan hasil antara GA-SA I dan GA-SA II menunjukkan bahwa mutasi berpengaruh terhadap nilai momen yang diperoleh.

Pada GA-SA I, kemungkinan terjadinya mutasi sangat kecil, karena mutasi hanya dilakukan jika kedua *parent* sama. Mutasi dapat mengurangi kemungkinan solusi terjebak dalam *local optimum*, sehingga GA-SA II dapat menghasilkan nilai momen lebih kecil dari GA-SA I. Algoritma PSO menghasilkan nilai momen yang lebih besar dari GA-SA II, tetapi masih lebih kecil dari GA-SA I. Pada percobaan yang telah dilakukan, algoritma GA-SA I memiliki performansi terburuk dalam meminimumkan momen. Berdasarkan segi waktu, maka selisih *computational time* antar algoritma ini tidak menghasilkan sesuatu yang berbeda bagi perusahaan. *Layout* yang telah dihasilkan dari GA-SA II sebagai *layout* terbaik masih memerlukan *adjusment* berdasarkan masukan dari perusahaan, agar bentuk *layout* departemen lebih beraturan dan dapat direalisasikan. Hasil *layout* yang telah mengalami *adjustment* ditunjukkan pada Gambar 6 dengan nilai momen sebesar 56.364,9. Warna abu-abu pada *layout* menunjukkan area *fixed department* sesuai permintaan perusahaan. Hal ini berarti terjadi penurunan momen perpindahan sebesar 23,37% dibandingkan momen *initial layout* perusahaan.

Perbandingan performansi suatu algoritma tidak dapat dilihat melalui penerapan pada satu kasus saja, karena ada kemungkinan performansi suatu

Tabel 2. Rata-rata momen dan *computational time* algoritma setiap kasus

Kasus	GA-SA I		GA-SA II		PSO	
	Momen	CT (detik)	Momen	CT (detik)	Momen	CT (detik)
5 dept.	4.970,5739	**3,28**	**4.253,4683**	3,31	4.547,2530	4,25
10 dept.	12.480,5336	**4,75**	**8.959,8580**	4,81	10.711,7593	8,33
15 dept.	17.488,1822	6,59	**13.569,9494**	**6,48**	15.479,7830	14,24

algoritma bersifat *problem dependent*, oleh karena itu penulis melakukan percobaan pada kasus tata letak fasilitas dengan jumlah departemen yang berbeda-beda. Masing-masing algoritma akan diterapkan untuk mencari solusi terbaik untuk kasus 5 departemen, 10 departemen, dan 15 departemen tanpa adanya *fixed department*. Data dimensi setiap departemen dan frekuensi perpindahan dibangkitkan secara *random*. Rata-rata momen dan *computational time* setiap algoritma pada setiap kasus ditunjukkan pada Tabel 2.

Berdasarkan hasil percobaan untuk setiap kasus, momen terkecil selalu dicapai oleh GA-SA II, diikuti oleh PSO dan GA-SA I. Setelah dilakukan pengujian yang sama dengan studi kasus sebelumnya, didapatkan bahwa nilai momen yang dihasilkan ketiga algoritma berbeda secara signifikan. Dengan demikian, pada semua kasus yang dicoba, algoritma GA-SA II selalu menghasilkan momen terkecil. Hal ini disebabkan oleh faktor banyaknya mutasi yang terjadi, sehingga memperbesar daerah pencarian solusi. Jika dilihat dari segi *computational time*, diperoleh bahwa PSO membutuhkan waktu paling lama dibandingkan GA-SA I dan GA-SA II. Hasil pengujian juga menunjukkan bahwa *computational time* PSO berbeda secara signifikan dengan GA-SA I dan GA-SA II, sedangkan *computational time* GA-SA I dan GA-SA II tidak berbeda secara signifikan. *Computational time* PSO yang lebih singkat pada studi kasus sebelumnya disebabkan karena adanya beberapa *fixed department*.

Simpulan

Dari beberapa kasus yang dicoba, Algoritma GA-SA II memiliki performansi terbaik dalam meminimumkan momen perpindahan dengan *computational time* yang pendek bila dibandingkan dengan GA-SA I dan PSO. Parameter terbaik yang digunakan GA-SA II dalam penelitian ini adalah *cooling ratio* sebesar 0,95 dan probabilitas mutasi 0,7.

Daftar Pustaka

1. Castillo I., Westerlund J., Emet S., and Westerlund T., Optimization of Blocklayout Design Problems with Unequal Areas: A Comparison of MILP and MINLP Optimization Methods, *Computers and Chemical Engineering*, 30(1), 2005, pp. 54-69.
2. Chutima, P., Genetic Algorithm for Facility Layout Design with Unequal Departmental Areas and Different Geometric Shape Constraints, *Thammasat Int. J. Sc. Tech.*, 6(2), 2001, pp. 33-43.
3. El-Baz, M. A., A Genetic Algorithm for Facility Layout Problems of Different Manufacturing Environments, *Computers and Industrial Engineering*, 47, 2004, pp. 233-246.
4. Hakim, E. A., Soeprijanto, A., and Mauridhi, H. P., PSS Design Based on PD and PI Fuzzy Controller by Particle Swarm Optimization, *Proceedings of the International Conference on Electrical Engineering and Informatics*, 2007, pp. 723-726.
5. Kennedy, J., and Eberhart, R., Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
6. Liu, H., Abraham, A., and Zhang, J., A Particle Swarm Approach to Quadratic Assignment Problems, *Soft Computing in Industrial Application*, 39, 2007, pp. 213-222.
7. McKendall, A. R., Jr., Shang, J., and Kuppusamy, S., Simulated Annealing heuristics for the Dynamic Facility Layout Problem, *Computers and Operations Research*, 33, 2006, pp. 2431-2444.
8. Nordin, N. N., Zainuddin, Z. M., Salim, S., and Ponnusamy, R. R., Mathematical Modeling and Hybrid Heuristic for Unequal Size Facility Layout Problem, *Journal of Fundamental Science*, 5, 2009, pp. 79-87.
9. Purnomo, H., *Perencanaan dan Perancangan Fasilitas*, Graha Ilmu Yogyakarta, 2004.
10. Shi, Y., and Eberhart, R. C., A Modified Particle Swarm Optimizer, *Proceedings of IEEE International Conference on Evolutionary Computation*, 1998.
11. Uysal, O., and Bulkan, S., Comparison Genetic Algorithm and Particle Swarm Optimization for Bicriteria Permutation Flowshop Scheduling Problem, *International Journal of Computational Intelligence Research*, 8(2), 2008, pp. 159-175.