ALGORITMA *SIMULATED ANNEALING* UNTUK PEMBENTUKAN SEL MESIN DENGAN DUA TIPE FUNGSI OBJEKTIF DAN DUA CARA PEMBATASAN SEL

Henry Pantas Panggabean
Dosen Fakultas MIPA, Jurusan Ilmu Komputer – Universitas Katolik Parahyangan
E-mail: henrypp@home.unpar.ac.id

ABSTRAK

Artikel ini menguraikan penerapan algoritma *simulated annealing* (SA) untuk menyelesaikan masalah pembentukan sel mesin dalam *group technology* dengan dua tipe fungsi objektif, yaitu minimasi jumlah perpindahan *part* antar sel dan minimasi jumlah duplikasi mesin *bottleneck*. Juga disediakan dua alternatif pembatasan sel bagi pengguna, yaitu penetapan batas atas ukuran sel atau penetapan jumlah sel mesin yang dikehendaki. Algoritma ini diujikan terhadap beberapa kasus uji dengan berbagai ukuran dalam literatur dan terbukti dapat menghasilkan kualitas solusi lebih baik dalam beberapa kasus dibandingkan yang tercatat dalam literatur. Sedangkan jumlah iterasi (waktu komputasi) yang dibutuhkan algoritma SA untuk mencapai solusi terbaiknya tergolong cukup kecil.

Kata kunci: teknologi grup, pembentukan sel mesin, perpindahan antar sel, algoritma *simulated annealing*

ABSTRACT

This article presents the implementation of simulated annealing (SA) algorithm to solve machine cell formation problem in group technology, with two types of objective function, i.e. minimization of intercellular part move and minimization of bottleneck machine duplication. There are also two ways for users to arrange the cell structure, i.e. define the upper bound of cell size or set the number of cells in final solution. This SA algorithm is then applied to solve some test cases with various size found in literature and proves itself to be able to achieve solutions with better quality then ones recorded in literature in some test cases. Meanwhile, number of iterations or the amount of computation time needed by this algorithm to proceed its best solution is quite small.

***Keywords**: group technology, machine cell formation, intercellular move, simulated annealing algorithm*

1. PENDAHULUAN

Group Technology (GT) adalah sebuah filosofi dalam dunia manufaktur yang mengidentifikasi dan mengelompokkan *part-part* yang serupa ke dalam kelompok *part* (*part family*) dengan memanfaatkan kesamaan dalam hal rancangan produk dan proses fabrikasi dalam siklus manufaktur. GT bertujuan untuk mengurangi waktu *setup*, aktivitas penanganan material, waktu *throughput*, inventori *in-process*, kebutuhan ruangan, waktu

idle mesin, dan kompleksitas kontrol, yang pada gilirannya akan meningkatkan efisiensi produksi.

Cellular manufacturing (CM) merupakan salah satu aplikasi dari GT. Ide yang mendasari CM adalah pengelompokan mesin ke dalam sel-sel untuk memproduksi *part family*, yaitu sekelompok *part* yang membutuhkan proses-proses manufaktur yang serupa. CM mendekomposisi suatu sistem produksi ke dalam beberapa sub sistem, yang disebut sel mesin (*machine cell*), dimana dalam tiap sel dapat diproses satu atau beberapa *part family* secara penuh tanpa melakukan perpindahan antar sel.

Masalah pertama yang harus ditangani dalam desain sistem manufaktur seluler adalah tahap pembentukan sel mesin (*machine cell formation*). Tahap ini menjadi kritis karena keberhasilan sistem ini sangat tergantung pada pengelompokan awal dari mesin dan *part*. Independensi mutlak antar sel mesin sulit dicapai dalam dunia nyata, karena adanya beberapa *part* yang harus diproses di dalam lebih dari satu sel. *Part* seperti ini disebut *exceptional part* dan mesin yang memprosesnya disebut *bottleneck machine*, dimana keduanya mengakibatkan terjadinya perpindahan antar sel yang meningkatkan inventori *work-in-process*, biaya penanganan material, dan menurunkan efisiensi sistem. Perpindahan antar sel dapat dihilangkan dengan melakukan duplikasi terhadap sejumlah *bottleneck machine* atau melakukan sub kontrak untuk *exceptional part*.

Proses pembentukan sel mesin memanfaatkan matriks keterhubungan (*incidence matrix*) mesin-komponen, yang dibentuk dari informasi yang terdapat pada *part routing sheet.* Entri ke-(*i*,*j*) dalam matriks akan bernilai 1 jika komponen ke-*j* diproses oleh mesin ke-*i*; jika tidak, nilainya 0 atau kosong. Matriks ini dimanipulasi sampai diperoleh bentuk diagonal blok yang baik.

Hal yang penting untuk dipertimbangkan dalam pembentukan sel mesin adalah maksimasi tingkat independensi antar sel dan maksimasi tingkat utilisasi mesin dalam tiap sel tanpa melanggar batasan ukuran sel dan jumlah sel yang telah ditetapkan. Fungsi objektif (*cost function*) berupa minimasi jumlah perpindahan *part* antar sel atau minimasi jumlah duplikasi mesin *bottleneck* perlu dirancang secara tepat guna dan sederhana agar dapat memenuhi hal-hal di atas tanpa membutuhkan waktu komputasi yang besar.

Masalah pembentukan sel mesin merupakan suatu bentuk masalah data *clustering*. Karena input datanya berupa sebuah matriks keterhubungan mesin-komponen yang bernilai biner (0 dan 1), maka masalah ini tergolong optimasi kombinatorial, yaitu pencarian nilai optimum dari suatu fungsi variabel-variabel diskrit, yang membutuhkan waktu penyelesaian yang meningkat secara non-polinomial seiring dengan bertambahnya ukuran masalah.

Banyak masalah optimasi kombinatorial teoretis dan praktis hanya dapat diselesaikan dengan metode aproksimasi (heuristik) oleh komputer saat ini. Algoritma *simulated annealing* (SA) merupakan salah satu teknik pencarian acak yang cukup fleksibel dan telah berhasil diterapkan untuk menemukan solusi optimal atau mendekati optimal dalam berbagai tipe masalah optimasi kombinatorial (Ben-Arieh dan Maimon, 1992; Laarhoven *et al.*, 1992; Lo dan Bavarian, 1991). Algoritma SA juga telah diterapkan dalam masalah pembentukan sel mesin. Liu dan Wu (1993) menggunakan algoritma SA untuk membentuk sel mesin dengan dua pilihan fungsi objektif, yaitu minimasi jumlah perpindahan *part* antar sel dan minimasi jumlah duplikasi mesin *bottleneck*. Untuk memperkirakan jumlah sel yang akan dibentuk, digunakan algoritma *cluster identification* (CIA). Chen *et al.* (1995) menerapkan algoritma SA untuk membentuk sel mesin dengan fungsi objektif minimasi jumlah perpindahan *part* antar sel.

Chen *et al.* tidak menggunakan metode tertentu untuk menentukan jumlah sel yang akan dibentuk, tetapi memberi keleluasaan kepada pengguna untuk mengeset batas atas ukuran sebuah sel.

Dalam penelitian ini akan dikembangkan model perangkat lunak yang menerapkan algoritma SA untuk menyelesaikan masalah pembentukan sel mesin dengan masukan berupa matriks keterhubungan mesin-komponen bernilai biner dengan dua tipe fungsi objektif, yaitu (1) minimasi jumlah perpindahan *exceptional part* antar sel, dan (2) minimasi jumlah duplikasi mesin *bottleneck*. Pada awal proses, pengguna dapat memilih salah satu dari dua cara pembatasan sel, yaitu (1) menentukan batas atas ukuran tiap sel (jumlah maksimal mesin dalam tiap sel), atau (2) menentukan jumlah sel mesin yang akan dihasilkan. Lalu akan ditunjukkan bahwa model algoritma SA ini dapat menghasilkan solusi yang baik untuk berbagai ukuran masalah pembentukan sel mesin yang ada dalam literatur dengan waktu komputasi yang relatif kecil, bahkan dalam beberapa masalah, solusi akhir yang dihasilkan lebih baik dibandingkan hasil penelitian yang telah dipublikasikan dalam literatur.

2. ALGORITMA *SIMULATED ANNEALING* (SA)

Algoritma SA diperkenalkan oleh Metropolis *et al.* pada tahun 1953. Dalam bidang optimasi algoritma ini beranalogi dengan proses *annealing* (pendinginan) yang diterapkan dalam pembuatan material yang terdiri dari butir kristal. Dari sisi ilmu fisika, tujuan sistem ini adalah untuk meminimasi energi potensial. Fluktuasi kinematika acak menghalangi sistem untuk mencapai energi potensial yang minimum global, sehingga sistem dapat terperangkap dalam sebuah keadaan minimum lokal. Dengan menurunkan temperatur sistem, diharapkan energi dapat dikurangi ke suatu level yang relatif rendah. Semakin lambat laju pendinginan ini, semakin rendah pula energi yang dapat dicapai oleh sistem pada akhirnya.

Dalam konteks optimasi pada algoritma SA, temperatur adalah variabel kontrol yang berkurang nilainya selama proses optimasi. Level energi sistem diwakili oleh nilai fungsi objektif. Skenario pendinginan dianalogikan dengan prosedur *search* yang menggantikan satu *state* dengan *state* lainnya untuk memperbaiki nilai fungsi objektif. Analogi ini cocok untuk masalah optimasi kombinatorial dimana jumlah *state* terbatas namun terlalu besar untuk ditelusuri dengan cara *enumerative search (brute-force)*. Namun, topologi sistem harus dibuat sedemikian rupa sehingga setiap *state* dapat dicapai dari setiap *state* lainnya, sehingga terdapat sebuah *path* dari setiap minimum lokal menuju minimum global.

Algoritma SA bertujuan untuk meminimasi sebuah fungsi objektif (*cost function*). Pada awal proses, sebuah solusi awal dipilih secara acak dari ruang solusi atau dibuat dengan menggunakan metode heuristik tertentu. Lalu dari solusi awal ini di-*generate* sebuah solusi baru, yang kemudian dibandingkan nilai fungsi objektifnya dengan solusi awal. Jika solusi baru ini lebih baik, ia akan diterima. Keunikan metode SA adalah bahwa solusi yang lebih buruk masih dapat diterima dengan nilai peluang tertentu, sehingga sistem dapat terhindar dari perangkap titik minimum lokal, namun solusi terbaik yang pernah dicapai selama proses *annealing* selalu dicatat. Kemampuan untuk sewaktu-waktu menerima solusi yang lebih buruk inilah yang membedakan algoritma SA dari metode *gradient descent* atau *local search* biasa. Dalam *gradient descent*, apabila sebuah titik minimum lokal dari fungsi objektif telah diperoleh, tidak akan mungkin dilakukan

perbaikan lebih jauh. Dalam algoritma SA, langkah *'hill climbing'* dapat dilakukan dari minimum lokal.

Diterimanya solusi yang lebih buruk didasarkan pada sebuah fungsi probabilitas yang nilainya tergantung pada dua faktor, yaitu (1) besarnya selisih nilai fungsi objektif antara solusi sekarang dengan solusi sebelumnya (makin besar selisih nilainya dengan solusi sebelumnya, peluang diterima akan semakin kecil) dan (2) jumlah iterasi yang telah dilakukan atau nilai temperatur pada kondisi tersebut (makin besar jumlah iterasi atau semakin kecil nilai temperatur, peluang diterima akan semakin kecil). Faktor kedua ini merupakan salah satu penyebab dari sifat konvergensi algoritma SA.

Implementasi algoritma SA sebagai sebuah algoritma optimasi sebenarnya membutuhkan jumlah transisi yang tak terbatas secara asimptotik. Namun dalam aplikasinya, implementasi algoritma SA secara *finite-time* lebih ditujukan untuk mengaproksimasi sebuah solusi optimal dari masalah yang dihadapi. Implementasi *finite-time* ini dapat diartikan bahwa dalam proses pencarian, nilai Markov homogen dengan panjang terbatas dibangkitkan untuk suatu urutan terbatas dari nilai parameter kontrol yang semakin menurun.

Untuk menghasilkan perilaku konvergensi dalam algoritma SA, sekumpulan parameter harus didefinisikan terlebih dahulu di awal proses. Cara pendefinisian parameter-parameter ini disebut *cooling schedule* , yang melibatkan:

(i) nilai awal untuk parameter kontrol temperatur (T_0)
(ii) fungsi/faktor penurunan nilai temperatur (F)
(iii) jumlah iterasi dalam tiap nilai temperatur atau panjang rantai Markov homogen (L)
(iv) nilai akhir untuk temperatur (T_1) atau kriteria terminasi untuk menghentikan eksekusi

1. Tetapkan skema pendinginan (nilai parameter *annealing*): temperatur awal (T_0) dan temperatur akhir (T_1), faktor penurunan temperatur (F), dan jumlah iterasi dalam tiap nilai temperatur (L).
2. Pilih solusi awal X secara acak atau dengan metode heuristik tertentu.
3. Hitung nilai fungsi objektif: $E = f(X)$.
4. Set nilai parameter kontrol temperatur : $T = T_0$.
5. Ulangi langkah 6 s.d. 8 sebanyak L kali
6. Bangkitkan solusi baru X_{new}
7. Hitung nilai fungsi objektif: $E_{new} = f(X_{new})$.
8. Tetapkan $X = X_{new}$ dan $E = E_{new}$ dengan probabilitas (kriteria Metropolis): min $\{1, exp((f(X) - f(X_{new}) / T)\}$
9. Kurangi nilai parameter kontrol temperatur dengan mengeset : $T = T \times F$
10. Berhenti jika telah dicapai kriteria terminasi; jika tidak kembali ke langkah 5.

Gambar 1. Algoritma Umum SA

Kondisi terminasi proses pencarian dalam algoritma SA dapat berupa dicapainya suatu jumlah iterasi tertentu dimana selama itu tidak ada solusi baru yang diterima, atau dicapainya nilai temperatur tertentu yang telah ditetapkan sebelumnya ($T < T_1$). Gambar 1 menunjukkan algoritma SA secara umum.

3. PEMODELAN MASALAH

Ada beberapa hal penting yang harus dirancang dalam menerapkan algoritma SA untuk menyelesaikan sebuah masalah optimasi kombinatorial, yaitu fungsi objektif (*cost function*), mekanisme inisialisasi solusi awal dan pembuatan solusi baru, skema pendinginan (*cooling schedule*), dan penetapan batasan terhadap *output* yang dikehendaki. Tiap hal tersebut akan diuraikan pada bagian-bagian berikut ini.

3.1 Fungsi Objektif

Untuk mengakomodasi dua tipe fungsi objektif pada pembentukan sel mesin yang disediakan dalam penelitian ini, ditempuh dua alternatif pengelompokan, yaitu:

(a) Tipe 1 : Pengelompokan mesin-mesin ke dalam sel-sel untuk meminimasi jumlah perpindahan antar sel (*intercellular move*) untuk *part* yang harus dikerjakan pada lebih dari satu sel (*exceptional part*), lalu diikuti dengan pengalokasian *part-part* ke dalam sel-sel yang ada berdasarkan data pada *incidence matrix*. Jika sebuah *part* harus mengunjungi *n* sel untuk diproses secara parsial atau secara penuh untuk kebutuhan pemesinannya, maka *part* tersebut mengakibatkan (*n*-1) perpindahan antar sel. Jadi fungsi objektif (*cost function*) yang digunakan adalah :
$MD = \Sigma\ (D_j), j = 1, \ldots, m$; m = jumlah mesin
MD = jumlah total duplikasi seluruh mesin; D_j = jumlah duplikasi untuk mesin j

(b) Tipe 2 : Pengelompokan *part-part* ke dalam sel-sel untuk meminimasi jumlah duplikasi mesin *bottleneck* yang dibutuhkan oleh lebih dari satu sel, lalu diikuti dengan pengalokasian mesin-mesin ke dalam sel-sel yang ada berdasarkan data pada *incidence matrix*. Jika sebuah mesin dibutuhkan oleh *n* sel untuk mencapai secara parsial atau secara penuh kebutuhan proses dari beberapa *part* dalam sel tersebut, maka mesin tersebut perlu diduplikasi (*n*-1) kali. Jadi fungsi objektif (*cost function*) yang digunakan adalah :
$IM = \Sigma\ (M_k), k = 1, \ldots, p$; p = jumlah *part*
IM = jumlah total perpindahan antar sel; M_k = jumlah perpindahan antar sel untuk *part k*

3.2 Inisialisasi Solusi Awal dan Mekanisme Pembuatan Solusi Baru

Pada penelitian ini, solusi awal untuk formasi sel mesin diinisialisasi secara acak dengan cara sbb.: untuk fungsi objektif tipe 1, solusi awal dibuat dengan membagi mesin secara acak ke dalam sel-sel yang ada dan mengalokasikan *part-part* ke dalam sel sesuai dengan mesin yang dibutuhkannya; untuk fungsi objektif tipe 2, solusi awal dibuat dengan membagi *part* secara acak ke dalam sel-sel yang ada dan mengalokasikan mesin-mesin ke dalam sel sesuai dengan *part* yang membutuhkannya.

Mekanisme transisi yang digunakan untuk membangkitkan solusi baru adalah :

A) Untuk fungsi objektif tipe 1 :
Pilih sebuah tipe mesin secara acak, dan pilih sebuah nomor sel secara acak sebagai tempat alokasi baru untuk mesin tersebut, yang bukan merupakan sel yang ditempatinya sebelumnya. Ukuran *neighborhood* = jumlah total tipe mesin × (jumlah

sel – 1). Lalu realokasi semua *part* yang membutuhkan mesin tersebut berdasarkan transisi ini.

B) Untuk fungsi objektif tipe 2 :
Pilih sebuah tipe *part* secara acak, dan pilih sebuah nomor sel secara acak sebagai tempat alokasi baru untuk *part* tersebut, yang bukan merupakan sel yang ditempatinya sebelumnya. Ukuran *neighborhood* = jumlah total tipe *part* × (jumlah sel – 1). Lalu realokasi semua mesin yang dibutuhkan oleh *part* tersebut berdasarkan transisi ini.

3.3 Skema Pendinginan

Seperti telah dijelaskan sebelumnya, sebelum proses *annealing* dapat dimulai, harus ditetapkan terlebih dahulu skema pendinginan yang akan digunakan. Pada prinsipnya, semakin lambat proses *annealing* berlangsung, semakin besar peluangnya untuk menghasilkan solusi yang lebih baik, karena jumlah solusi yang dapat dievaluasi semakin banyak atau ruang pencarian yang dapat dijelajahi semakin luas.

Ada tiga cara yang dapat ditempuh untuk memperlambat laju proses *annealing*, yaitu (a) memperbesar temperatur awal (T_0) atau memperkecil temperatur akhir (T_1), dimana $0.0 < T_1 < T_0$; (b) memperbesar faktor penurunan temperatur (F), dimana $0.0 \quad F \quad 1.0$; dan (c) memperbesar jumlah iterasi dalam tiap nilai temperatur (L), dimana $L \quad 1$.

Namun penelitian ini tidak ditujukan secara khusus untuk mengukur sejauh mana pengaruh perubahan skema pendinginan (*cooling schedule*) terhadap kinerja algoritma SA. Karena itu, setelah melakukan serangkaian percobaan terhadap berbagai kombinasi nilai parameter *annealing*, dengan mempertimbangkan *trade-off* antara kualitas solusi dan waktu komputasi yang dibutuhkan, dipilihlah *cooling schedule* sbb.: T_0 = 1000,0; T_1 = 1,0; F = 0,995; dan $L = m$ div 5 (m = jumlah mesin dalam masalah yang diselesaikan).

3.4 Penetapan Batasan Sel

Guna memberi keleluasaan bagi pengguna untuk menentukan format struktur sel mesin yang dihasilkan, dalam penelitian ini disediakan dua alternatif pembatasan sel, yaitu:

(a) Pengguna mendefinisikan batas atas ukuran sel (jumlah maksimal mesin yang dapat dialokasikan ke dalam tiap sel), dan berdasarkan angka tersebut algoritma SA menentukan sebuah bilangan sebagai jumlah maksimal sel mesin yang akan dibentuk. Pada akhir eksekusi SA, dapat terjadi adanya sel mesin yang kosong dalam solusi akhir; sel yang demikian akan dihapus dari solusi.

(b) Pengguna menentukan jumlah sel mesin yang harus dibentuk; solusi akhir akan memuat sel mesin seperti jumlah yang ditetapkan pengguna. Tidak akan ada sel mesin yang kosong pada solusi akhir yang dihasilkan. Model kedua ini dapat dimodifikasi sedikit sehingga pengguna dapat memberikan batas atas jumlah sel mesin yang akan dibentuk; solusi akhir dapat memuat sel mesin dalam jumlah yang kurang dari atau sama dengan nilai batas atas tersebut. Dapat terjadi adanya sel mesin yang kosong yang kemudian akan dihapus dari solusi akhir. Berdasarkan jumlah sel yang dikehendaki pengguna tersebut, ditentukan batas atas dari ukuran tiap sel.

3.5 Model Algoritma SA yang Diimplementasikan

Berdasarkan uraian di atas, dikembangkan sebuah model implementasi algoritma SA untuk pembentukan sel mesin seperti yang diberikan pada Gambar 2.

1. Lakukan inisialisasi parameter *annealing*:
   - ✓ temperatur awal (T_0) dan temperatur akhir (T_1) : $0.0 < T_1 < T_0$
   - ✓ faktor penurunan temperatur : $0.0 \quad F \quad 1.0$
   - ✓ jumlah iterasi dalam tiap nilai temperatur (panjang rantai Markov) : $L \quad 1$

   Lakukan inisialisasi parameter masalah pembentukan sel mesin :
   - ✓ jumlah mesin dan jumlah *part* (berdasarkan kasus uji yang dipilih)
   - ✓ jumlah sel mesin yang dikehendaki atau batas atas ukuran tiap sel mesin
2. Pilih solusi awal X :
   (a) Tipe 1 : Tempatkan tiap mesin secara acak dalam salah satu sel yang ada.
   Tipe 2 : Tempatkan tiap *part* secara acak dalam salah satu sel yang ada.
   (b) Berdasarkan pengelompokan mesin (atau *part*) pada langkah 2(a), tempatkan tiap *part* (atau mesin) ke dalam semua sel di mana mesin yang dibutuhkannya (atau *part* yang membutuhkannya) berada.
   (c) Tipe 1 : Identifikasi semua *exceptional part*, yaitu *part* yang harus dikerjakan pada lebih dari satu sel, dan hitung jumlah perpindahan dari tiap *part*.
   Tipe 2 : Identifikasi semua *bottleneck machine*, yaitu mesin yang dibutuhkan oleh lebih dari satu sel, dan hitung jumlah duplikasi untuk tiap mesin.
3. Hitung nilai fungsi objektif: $E = f(X)$, yaitu:
   Tipe 1 : Hitung jumlah total perpindahan untuk semua *exceptional part*.
   Tipe 2 : Hitung jumlah total duplikasi untuk semua mesin *bottleneck*.
4. Set nilai parameter kontrol temperatur : $T = T_0$.
5. Ulangi langkah 6 s.d. 8 sebanyak L kali.
6. Bangkitkan solusi baru X_{new} :
   (a) Tipe 1 : Pilih sebuah mesin secara acak untuk dipindahkan dan pilih sebuah sel secara acak untuk tujuan pemindahan (bukan sel yang ditempati saat ini).
   Tipe 2 : Pilih sebuah *part* secara acak untuk dipindahkan dan pilih sebuah sel secara acak untuk tujuan pemindahan (bukan sel yang ditempati saat ini).
   (b) Ubah penempatan *part-part* (atau mesin-mesin) yang berkaitan dengan mesin (atau *part*) yang dipindahkan pada langkah 6(a).
   (c) Tipe 1 : Identifikasi *exceptional part* dan hitung jumlah perpindahan tiap *part*.
   Tipe 2 : Identifikasi *bottleneck machine* dan hitung jumlah duplikasi tiap mesin.
7. Hitung nilai fungsi objektif: $E_{new} = f(X_{new})$. (sama seperti langkah 3)
8. Tetapkan $X = X_{new}$ dan $E = E_{new}$ dengan probabilitas berikut:

$$P\{terima\ X_{new}\} = \begin{cases} 1 & if\ f(X_{new}) \le f(X) \\ \exp\left(\dfrac{f(X) - f(X_{new})}{T}\right) & if\ f(X_{new}) > f(X) \end{cases}$$

9. Kurangi nilai temperatur dengan mengeset: $T = T \times F$.
10. Jika nilai $T \le T_1$, maka terminasi proses; jika tidak, maka kembali ke langkah 4.

Gambar 2. Model Algoritma SA yang Diimplementasikan

4. HASIL PERCOBAAN DAN ANALISIS KINERJA ALGORITMA SA

4.1 Kasus Uji yang Digunakan

Untuk mengukur kinerja dari model algoritma SA yang dikembangkan, dipilih lima buah kasus uji pembentukan sel mesin dari literatur dengan matriks keterhubungan mesin-komponen bernilai biner dan dimensi masalah berbeda-beda, yaitu :

(i) Masalah 1 (20 mesin, 35 *part*) dari Liu dan Wu (1993); sumber asli Carrie (1973).
(ii) Masalah 2 (20 mesin, 50 *part*) dari Chen *et al*. (1995) sebagai sumber asli.
(iii) Masalah 3 (20 mesin, 60 *part*) dari Chen *et al*. (1995) sebagai sumber asli.
(iv) Masalah 4 (16 mesin, 43 *part*) dari Joines *et al.* (1996); sumber asli Burbidge (1975).
(v) Masalah 5 (40 mesin, 100 *part*) dari Joines *et al.* (1996); sumber asli Chandrasekharan dan Rajagopalan (1986).

Pada Gambar 3, diberikan contoh matriks keterhubungan mesin – *part* untuk kasus uji 16 mesin 43 *part* yang digunakan dalam penelitian ini (Joines *et al.*, 1996).

	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4 4 4
	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
01	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
02	0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0
03	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
04	0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
05	0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1
06	1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1
07	1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
08	1 1 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1
09	0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0
10	1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
11	0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
12	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
13	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
15	0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1
16	0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0

Gambar 3. *Machine-part Incidence Matrix* untuk Masalah 16 Mesin 43 *Part*

Algoritma SA dieksekusi sebanyak 20 kali untuk menyelesaikan tiap kasus uji dengan menggunakan skema pendinginan yang telah disebutkan di atas. Lalu kinerjanya diukur berdasarkan tiga hal, yaitu :

a. Feasibilitas (validitas) solusi yang dihasilkan dan kualitas solusi terbaik yang dapat dicapai, dibandingkan terhadap catatan solusi terbaik untuk masing-masing kasus uji yang berhasil dikumpulkan dari literatur.
b. Waktu komputasi yang dibutuhkan, juga dibandingkan terhadap catatan waktu komputasi dalam literatur (karena perangkat komputer yang digunakan berbeda, maka pembandingannya dikaitkan dengan jumlah iterasi yang dilakukan dalam tiap eksekusi).

c. Fleksibilitas algoritma dalam menghasilkan solusi yang feasibel sesuai pembatasan sel yang ditetapkan oleh pengguna (berdasarkan batas atas ukuran sel atau jumlah sel yang dikehendaki pengguna).

Tiap item kinerja di atas akan diuraikan lebih rinci pada bagian-bagian berikut ini.

4.2 Kualitas Solusi untuk Kedua Tipe Objektif

Melalui serangkaian percobaan eksekusi (*run*) yang telah dilakukan terhadap tiap kasus uji, didapati bahwa model algoritma SA yang diimplementasikan dalam penelitian ini dapat menghasilkan kualitas solusi akhir yang sama baik dengan yang telah dicatat pada literatur, bahkan pada beberapa kasus uji dapat dihasilkan kualitas solusi yang lebih baik. Berikut ini akan ditunjukkan solusi terbaik yang berhasil dicapai untuk tiap kasus uji.

4.2.1 Masalah 20 mesin 35 *part.* Solusi terbaik yang berhasil dicapai untuk kasus uji ini sama dengan solusi terbaik yang dicatat dalam literatur (Liu dan Wu, 1993; Chen *et al.*, 1995, dan Joines *et al.* 1996). Untuk fungsi objektif tipe 1, dengan jumlah sel mesin = 4, jumlah minimal perpindahan *part* antar sel = 2 (*part* 23 dan *part* 31). Sedangkan untuk fungsi objektif tipe 2, dengan jumlah *part family* = 4, jumlah minimal duplikasi mesin *bottleneck* = 2 (mesin 3 dan mesin 20).

4.2.2 Masalah 20 mesin 50 *part.* Solusi terbaik yang dicatat dalam literatur (Chen *et al.*, 1995) untuk masalah ini adalah 9 perpindahan antar sel untuk 4 sel mesin dan 6 perpindahan antar sel untuk 3 sel mesin. Pada penelitian ini, berhasil dicapai solusi akhir yang lebih baik, yaitu 7 perpindahan antar sel untuk 4 sel mesin dan 5 perpindahan antar sel untuk 3 sel mesin. Solusi terbaik untuk fungsi objektif tipe 1 diberikan pada Tabel 1 dan Tabel 2.

Sedangkan untuk fungsi objektif tipe 2, solusi terbaik yang berhasil dicapai adalah 7 duplikasi mesin dengan jumlah *part family* = 4. Ada beberapa konfigurasi solusi akhir berbeda dengan nilai objektif sama yang dapat dihasilkan oleh algoritma SA. Salah satu solusi tersebut diberikan pada Tabel 3. Tidak diperoleh catatan pembanding dalam literatur untuk kasus uji ini. Ada satu catatan khusus pada matriks *incidence* kasus uji ini, yaitu *part* 30 dan *part* 50 tidak terhubung dengan mesin mana pun.

Tabel 1. Solusi Terbaik Masalah 20 Mesin 50 *Part* untuk Objektif Tipe 1 dan Jumlah Sel = 4

Nomor Sel	Jumlah Mesin	Nomor Mesin	Jumlah *Part*	Nomor *Part*
1	5	1, 2, 3, 4, 5	13	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 14, 42
2	5	11, 12, 13, 14, 15	14	5, 6, 16, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35
3	9	6, 8, 9, 10, 16, 17, 18, 19, 20	26	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
4	1	7	2	14, 40

Jumlah perpindahan *part* antar sel = 7, yaitu 6(1), 14(2), 16(1), 24(1), 40(1), 42(1).

Catatan: angka di depan kurung menunjukkan nomor *exceptional part* dan angka di dalam kurung menunjukkan jumlah perpindahan antar sel yang dibutuhkan oleh *part* tersebut.

Tabel 2. Solusi Terbaik Masalah 20 Mesin 50 *Part* untuk Objektif Tipe 1 dan Jumlah Sel = 3

Nomor Sel	Jumlah Mesin	Nomor Mesin	Jumlah *Part*	Nomor *Part*
1	5	1, 2, 3, 4, 5	13	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 14, 42
2	5	11, 12, 13, 14, 15	14	5, 6, 16, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35
3	10	6, 7, 8, 9, 10, 16, 17, 18, 19, 20	26	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49

Jumlah perpindahan *part* antar sel = 5, yaitu 6(1), 14(1), 16(1), 24(1), 42(1).

Tabel 3. Solusi Terbaik Masalah 20 Mesin 50 *Part* untuk Objektif Tipe 2 dan Jumlah *Family* = 4

Nomor *Family*	Jumlah *Part*	Nomor *Part*	Jumlah Mesin	Nomor Mesin
1	14	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 39, 40, 50	8	4, 6, 7, 8, 9, 10, 15, 19
2	12	36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49	7	2, 9, 16, 17, 18, 19, 20
3	13	5, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35	6	9, 11, 12, 13, 14, 15
4	11	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12	6	1, 2, 3, 4, 5, 11

Jumlah duplikasi mesin *bottleneck = 7, yaitu* 2(1), 4(1), 9(2), 11(1), 15(1), 19(1).

Catatan: angka di depan kurung menunjukkan nomor mesin *bottleneck* dan angka di dalam kurung menunjukkan jumlah duplikasi yang dibutuhkan oleh mesin tersebut.

4.2.3 Masalah 20 mesin 60 *part*. Solusi terbaik yang dicatat dalam literatur (Chen *et al.*, 1995) untuk masalah ini adalah 27 perpindahan antar sel tanpa menyebutkan batas atas ukuran sel yang digunakan. Pada penelitian ini, dilakukan percobaan dengan menggunakan lima nilai batas atas ukuran sel, yaitu 4, 5, 6, 7, dan 8. Hasil lengkap untuk tiap nilai batas atas ditunjukkan pada Tabel 9. Solusi akhir yang dicapai untuk batas atas ukuran sel = 4 (lihat Tabel 4) sudah mengungguli solusi terbaik yang dicatat dalam Chen *et al.* (1995). Sedangkan untuk fungsi objektif tipe 2, solusi terbaik dengan jumlah *part family* = 4 diberikan pada Tabel 5.

Tabel 4. Solusi Terbaik Masalah 20 Mesin 60 *Part* untuk Objektif Tipe 1

Nomor Sel	Jumlah Mesin	Nomor Mesin	Jumlah *Part*	Nomor *Part*
1	4	13, 14, 15, 16	20	6, 9, 18, 27, 31, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 53, 54, 55
2	4	5, 6, 7, 8	16	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 34, 38, 40, 53, 54
3	4	9, 10, 11, 12	19	1, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 39, 42, 47, 52, 54
4	4	17, 18, 19, 20	14	12, 20, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60
5	4	1, 2, 3, 4	16	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 31, 36, 52, 55

Jumlah perpindahan *part* antar sel = 25, yaitu 1(1), 6(1), 9(1), 12(1), 18(1), 20(1), 23(1), 27(1), 31(1), 34(1), 36(1), 38(1), 39(1), 40(1), 42(1), 47(1), 52(2), 53(2), 54(3), 55(2).

Tabel 5. Solusi Terbaik Masalah 20 Mesin 60 *Part* untuk Objektif Tipe 2

Nomor *Family*	Jumlah *Part*	Nomor *Part*	Jumlah Mesin	Nomor Mesin
1	24	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 39, 40, 50	13	4, 6, 7, 8, 9, 10, 15, 19
2	25	36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49	13	2, 9, 16, 17, 18, 19, 20
3	11	5, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35	6	9, 11, 12, 13, 14, 15
4	11	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12	6	1, 2, 3, 4, 5, 11

Jumlah duplikasi mesin *bottleneck = 7, yaitu* 2(1), 4(1), 9(2), 11(1), 15(1), 19(1).

4.2.4 Masalah 16 mesin 43 *part.* Solusi terbaik yang dicatat dalam literatur untuk masalah ini adalah 26 perpindahan antar sel untuk batas atas ukuran sel = 5 (Chen *et al.*, 1995) atau untuk jumlah sel = 4 (Joines *et al.*, 1996). Pada penelitian ini, untuk batas atas ukuran sel yang sama, yaitu 5, berhasil dicapai beberapa konfigurasi solusi akhir yang lebih baik dengan 24 perpindahan antar sel. Salah satu solusi akhir tersebut diberikan pada Tabel 6. Sedangkan untuk fungsi objektif tipe 2, berhasil dicapai beberapa konfigurasi solusi terbaik dengan 5 duplikasi mesin (jumlah *part family* = 3). Tidak diperoleh catatan pembanding dalam literatur untuk objektif ini.

Tabel 6. Solusi Terbaik Masalah 16 Mesin 43 *Part* untuk Objektif Tipe 1

Nomor Sel	Jumlah Mesin	Nomor Mesin	Jumlah *Part*	Nomor *Part*
1	5	4, 5, 6, 8, 15	34	1, 2, 3, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 23, 24, 27, 28, 29, 31, 32, 33, 34, 37, 38, 39, 40, 41, 42, 43
2	3	11, 12, 13	8	3, 9, 11, 20, 22, 24, 27, 30
3	3	1, 7, 10	9	1, 12, 13, 25, 26, 31, 37, 39, 42
4	5	2, 3, 9, 14, 16	16	2, 4, 6, 7, 10, 17, 18, 28, 32, 34, 35, 36, 37, 38, 40, 42

Jumlah perpindahan *part* antar sel = 24, yaitu 1(1), 2(1), 3(1), 6(1), 7(1), 9(1), 11(1), 12(1), 13(1), 17(1), 20(1), 24(1), 27(1), 28(1), 31(1), 32(1), 34(1), 37(2), 38(1), 39(1), 40(1), 42(2).

4.2.5 Masalah 40 mesin 100 *part*. Solusi terbaik yang dicatat oleh Joines et al. (1996) untuk masalah ini adalah 36 perpindahan antar sel untuk jumlah sel = 10, sedangkan Chen *et al.* (1995) mencatat solusi terbaik 31 perpindahan antar sel untuk batas atas ukuran sel = 6. Pada penelitian ini, untuk jumlah sel yang sama, yaitu 10, dapat dicapai kualitas solusi yang sama (36 perpindahan antar sel). Namun untuk batas atas ukuran sel = 6, hanya dapat dicapai 32 perpindahan antar sel. Jika batas atas ukuran sel diperbesar menjadi 7, dapat dicapai solusi dengan 31 perpindahan antar sel (lihat Tabel 7). Sedangkan untuk fungsi objektif tipe 2 dengan jumlah sel = 10, solusi terbaik yang dicapai adalah 42 duplikasi mesin *bottleneck*. Tidak diperoleh catatan pembanding dalam literatur untuk objektif ini.

Tabel 7. Solusi terbaik masalah 40 mesin 100 *part* untuk objektif tipe 1
(a) batas atas ukuran sel = 6 (jumlah sel = 9; jumlah perpindahan *part* = 32)
(b) batas atas ukuran sel = 7 (jumlah sel = 8; jumlah perpindahan *part* = 31)

Nomor Sel	Jumlah Mesin	Nomor Mesin	Jumlah *Part*	Nomor *Part*
1	5	15, 18, 33, 34, 36	11	39, 45, 48, 67, 71, 73, 75, 91, 93, 94, 100
2	5	2, 10, 16, 21, 31	12	35, 47, 53, 65, 72, 77, 78, 79, 83, 86, 88, 93
3	5	11, 13, 14, 17, 35	19	1, 2, 3, 4, 5, 6, 7, 15, 16, 21, 24, 25, 27, 28, 39, 43, 48, 60, 81
4	4	1, 3, 7, 32	14	10, 18, 25, 29, 33, 34, 37, 41, 44, 48, 49, 50, 54, 55
5	6	5, 8, 22, 23, 37, 39	21	40, 63, 66, 68, 69, 77, 82, 84, 85, 86, 87, 89, 90, 92, 94, 95, 96, 97, 98, 99, 100
6	5	6, 12, 26, 38, 40	21	36, 38, 42, 50, 51, 52, 64, 65, 70, 72, 74, 75, 76, 78, 80, 81, 86, 87, 88, 95, 97
7	3	24, 27, 29	11	9, 12, 17, 19, 26, 30, 31, 40, 41, 43, 46
8a	3	4, 9, 20	9	8, 11, 13, **14**, 20, 22, 23, 32, 84
9a	4	19, 25, 28, 30	14	**14**, 46, 56, 57, 58, 59, 61, 62, 71, 73, 81, 83, 91, 99
8b	7	4, 9, 19, 20, 25, 28, 30	22	8, 11, 13, **14**, 20, 22, 23, 32, 46, 56, 57, 58, 59, 61, 62, 71, 73, 81, 83, 84, 91, 99

Jumlah perpindahan *part* antar sel = 32 (untuk batas atas ukuran sel = 6), yaitu:
14(1), 25(1), 39(1), 40(1), 41(1), 43(1), 46(1), 48(2), 50(1), 65(1), 71(1), 72(1), 73(1), 75(1), 77(1), 78(1), 81(2), 83(1), 84(1), 86(2), 87(1), 88(1), 91(1), 93(1), 94(1), 95(1), 97(1), 99(1), 100(1). *Part* no. 14 tidak memerlukan perpindahan jika ukuran sel = 7.

4.3 Jumlah Iterasi dan Waktu Komputasi

Model algoritma SA yang diimplementasikan dalam penelitian ini menggunakan skema pendinginan sbb.: T_0 = 1000,0; T_1 = 1,0; F = 0,995; L = *m* div 5 (*m* = jumlah mesin). Algoritma tersebut dikodekan dalam bahasa Pascal dan dieksekusi dengan

prosesor Intel Pentium-2 dengan RAM 64 Mbyte. Sebagai model pembanding, digunakan hasil penelitian Chen *et al.* (1995), yang menerapkan skema pendinginan sbb: $T_0 = 3,0$; $T_1 = 0,00001$; F = 0,975; L = 4 × *m* (*m* = jumlah mesin), dikodekan dalam bahasa Fortran, dan dieksekusi dengan perangkat sistem komputer Sun 4/490. Jumlah iterasi yang dihasilkan oleh masing-masing skema pendinginan dan waktu komputasi yang dibutuhkan oleh kedua model perangkat lunak ini ditunjukkan pada Tabel 8.

Tampak bahwa model algoritma SA yang dikembangkan dalam penelitian ini membutuhkan jumlah iterasi yang lebih sedikit dibandingkan model Chen *et al.* (1995), sedangkan kualitas solusi yang dihasilkan oleh model ini telah terbukti lebih baik dalam beberapa kasus uji. Jumlah iterasi yang jauh lebih kecil ini tentu saja akan berpengaruh langsung terhadap waktu komputasi yang dibutuhkannya.

Tabel 8. Perbandingan Jumlah Iterasi dan Waktu Komputasi

Masalah	Model implementasi SA Chen *et al.* (1995)		Model implementasi SA dalam penelitian ini	
	jumlah iterasi	waktu komputasi	jumlah iterasi	waktu komputasi
20 mesin 35 *part*	39920	1,8 detik	5516	0,50 detik
20 mesin 50 *part*	39920	*data tidak ada*	5516	0,50 detik
20 mesin 60 *part*	39920	7,6 detik	5516	0,75 detik
16 mesin 43 *part*	31936	18,1 detik	4137	0,43 detik
40 mesin 100 *part*	79840	19 detik	11032	3,40 detik

4.4 Fleksibilitas Algoritma SA terhadap Berbagai Ukuran Pembatasan Sel

Untuk membuktikan bahwa model algoritma SA yang dikembangkan mampu mengakomodasi berbagai batasan sel yang dikehendaki pengguna, telah dilakukan percobaan untuk fungsi objektif tipe 1 dengan menggunakan beberapa nilai batas atas ukuran sel (lihat Tabel 9) dan beberapa nilai jumlah sel mesin (lihat Tabel 10). Tampak bahwa model ini cukup fleksibel untuk menyelesaikan masalah dunia nyata guna menghasilkan formasi sel mesin yang paling sesuai dengan kendala *space* di lingkungan manufaktur. Perlu dicatat pula bahwa jumlah perpindahan *part* antar sel akan cenderung menurun apabila batas atas ukuran sel diperbesar atau jumlah sel yang hendak dibentuk diperkecil.

Tabel 9. Solusi Terbaik untuk Objektif Tipe 1 pada Beberapa Nilai Batas atas Ukuran Sel

Ukuran masalah		Batas atas ukuran sel							
		5		6		7		8	
Σ mesin	Σ *part*	Σ *move*	Σ sel	Σ *move*	Σ sel	Σ *move*	Σ sel	Σ *move*	Σ sel
20	35	2	4	2	4	2	4	2	4
20	50	9	4	9	4	9	4	8	3
20	60	25	5	22	4	20	3	17	3
16	43	24	5	20	4	17	3	14	2
40	100	45	9	32	9	31	8	31	7

Tabel 10. Solusi Trbaik yang Dicapai untuk Kedua Fungsi Objektif dengan Batasan Jumlah Sel Mesin yang Dibentuk

Ukuran masalah		Jumlah sel mesin yang harus dibentuk				
		2	3	4	5	6
Σ mesin	Σ *part*	Σ *move*	Σ *move*	Σ *move*	Σ *move*	Σ *move*
20	35	1	1	2	6	10
20	50	3	5	7	10	12
20	60	9	16	19	21	25
16	43	8	14	16	20	24
40	100	26 (7)	29 (8)	34 (9)	36 (10)	40 (11)

Catatan: Khusus untuk masalah 40 mesin 100 *part*, angka di dalam kurung menunjukkan jumlah sel mesin yang harus dibentuk (tidak mengikuti angka pada judul kolomnya).

Dari Tabel 9 dan 10 tampak bahwa penetapan jumlah sel berpeluang menghasilkan nilai fungsi objektif yang lebih baik dibandingkan penetapan batas atas ukuran sel. Misalnya, pada masalah 20 mesin 60 *part*, untuk jumlah sel mesin berturut-turut 3, 4, dan 5, Tabel 9 mencatat jumlah perpindahan 17, 22, dan 25, sedangkan Tabel 10 mencatat jumlah minimal perpindahan adalah 16, 19, dan 21. Hal ini dapat dijelaskan sbb.: apabila jumlah sel mesin telah ditetapkan, maka proses pencarian dapat berkonsentrasi penuh pada sub himpunan solusi dengan jumlah sel tertentu (seluruh iterasi dapat ditujukan untuk menjelajahi satu sub ruang solusi saja), sedangkan jika yang ditetapkan adalah batas atas ukuran sel, maka jumlah sel mesin dapat berubah-ubah selama proses pencarian sehingga pencarian kurang berfokus pada satu sub ruang solusi tertentu saja (sejumlah iterasi yang ada harus dibagi untuk menjelajahi beberapa sub ruang solusi yang berbeda).

5. KESIMPULAN

Dari serangkaian percobaan yang telah dilakukan terhadap beberapa masalah pembentukan sel mesin, dapat diambil kesimpulan berikut mengenai model algoritma SA yang telah dikembangkan dalam penelitian ini:

- Model ini dapat mengakomodasi dengan baik dua tipe fungsi objektif, yaitu minimasi jumlah perpindahan *part* antar sel dan minimasi jumlah duplikasi mesin *bottleneck*, serta dua pilihan cara pembatasan sel bagi pengguna, yaitu penetapan batas atas ukuran sel atau penetapan jumlah sel mesin yang harus dibuat.
- Model ini dapat menghasilkan kualitas solusi yang sama baik dengan hasil yang dicatat dalam literatur untuk berbagai ukuran kasus uji, khususnya untuk fungsi objektif minimasi jumlah perpindahan *part* antar sel, dengan waktu komputasi yang relatif kecil (jumlah iterasi yang relatif sedikit), bahkan untuk beberapa kasus uji dapat dihasilkan solusi dengan jumlah perpindahan *part* antar sel yang lebih kecil.
- Penetapan jumlah sel yang akan dibentuk berpeluang menghasilkan kualitas solusi yang lebih baik dibandingkan penetapan batas atas ukuran sel, karena jumlah sel yang tetap membuat proses pencarian dapat berfokus pada satu sub ruang solusi saja.

Untuk penelitian lebih lanjut, model ini dapat diperluas agar mampu menyelesaikan masalah pembentukan sel mesin dengan matriks keterhubungan yang bernilai tidak biner.

DAFTAR PUSTAKA

Ben-Arieh, D., O.Maimon, 1992, 'Annealing Method for PCB Assembly Scheduling on Two Sequential Machines", *International Journal of Computer Integrated Manufacturing*, Vol.5, No.6, 361 - 367.

Chen, C.L., N.A. Cotruvo, W. Baek, 1995. "A Simulated Annealing Solution to the Cell Formation Problem", *International Journal of Production Research*, Vol.33, No.9, 2601–2614.

Elsayed, E.A., T.O. Boucher, 1994, *Analysis and Control of Production Systems*, 2nd edition, Prentice-Hall International, New Jersey.

Joines, J.A., C.T. Culbreth, R.E. King, 1996, 'Manufacturing Cell Design: An Integer Programming Model Employing Genetic Algorithms", *IIE Transactions*, Vol. 28, No. 1, 69 – 85.

Laarhoven, P.J.M., E.H.L. Aarts, J.K. Lenstra, 1992. "Job Shop Scheduling by Simulated Annealing", *Operations Research*, Vol.40, No.1, 113 - 125.

Liu, C.M., J.K. Wu, 1993. 'Machine Cell Formation: Using the Simulated Annealing Algorithm", *International Journal of Computer Integrated Manufacturing*, Vol.6, No.6, 335 - 349.

Lo, Z.P., B. Bavarian, 1991. "Job Scheduling on Parallel Machines Using Simulated Annealing", *IEEE*, 391 - 396.