Penerapan Algoritma Consultant-Guided Search dalam Masalah Penjadwalan Job Shop untuk Meminimasi Makespan

Hotna Marina Sitorus [1*], Cynthia P. Juwono [2], Yogi Purnawan [3]
[1*,2,3)] Fakultas Teknologi Industri, Jurusan Teknik Industri,
Universitas Katolik Parahyangan
Jl. Ciumbuleuit 94, Bandung 40141
email : nina@unpar.ac.id, juwonocp@unpar.ac.id

Abstract

This research uses the Consultant-Guided Search (CGS) algorithm to solve job shop scheduling problems minimizing makespan. CGS is a metaheuristics inspired by people making decisions based on consultant's recommendations. A number of cases from literatures is developed to evaluate the optimality of this algorithm. CGS is also tested against other metaheuristics, namely Genetic Algorithms (GA) and Artificial Immune Systems (AIS) for the same cases. Performance evaluations are conducted using the best makespan obtained by these algorithms. From computational results, it is shown that CGS is able to find 3 optimal solutions out of 10 cases. Overall, CGS performs better compared to the other algorithms where its solution lies within 0 - 6,77% from the optimal solution, averaging only 2,15%. Futhermore, CGS outperforms GA in 7 cases and performs equally well in the other 3 cases. CGS is also better than AIS in 8 cases and is equally well in only 2 cases.

Abstrak

Penelitian ini bertujuan untuk menerapkan algoritma *Consultant-Guided Search* (CGS) untuk menyelesaikan masalah penjadwalan *job shop* untuk meminimasi *makespan*. CGS merupakan metode metaheuristik yang terinspirasi dari cara orang membuat keputusan berdasarkan saran yang diterima dari konsultan. Untuk mengevaluasi algoritma CGS yang dikembangkan, dilakukan pengujian pada sepuluh kasus penjadwalan *job shop* yang diperoleh dari literatur. Selain dievaluasi kemampuannya menghasilkan solusi optimal, algoritma yang dikembangkan juga dibandingkan dengan metode metaheuristik lainnya yaitu *Genetic Algorithms* (GA) dan *Artificial Immune Systems* (AIS) dalam menyelesaikan kasus-kasus yang sama. Perbandingan performansi dilakukan dengan menggunakan nilai *makespan* terbaik yang didapatkan oleh masing-masing metode. Berdasarkan studi komputasional yang dilakukan, algoritma CGS dapat mencapai solusi optimal pada 3 dari 10 kasus. Untuk keseluruhan kasus, selisih antara solusi CGS dengan solusi optimal berada pada kisaran 0 6,77% dengan rata-rata 2,15%. Perbandingan solusi antara algoritma CGS dan GA untuk kesepuluh kasus yang diujikan menunjukkan bahwa CGS lebih baik di 7 kasus dan sebanding di 3 kasus, sementara jika dibandingkan dengan AIS, CGS unggul di 8 kasus dan sebanding di 2 kasus.

Kata Kunci: *Consultant-Guided Search*, penjadwalan *job shop*, *makespan*

1 Pendahuluan

Dalam menjalankan usahanya, terdapat beberapa tujuan yang ingin dicapai oleh perusahaan manufaktur, yaitu untuk memaksimasi *output* produksi selama periode waktu tertentu, memenuhi kepuasan konsumen dari segi kualitas dan ketepatan waktu, dan meminimasi biaya yang dibutuhkan (Morton & Pentico, 1993). Dalam mencapai tujuan-tujuan tersebut, diperlukan suatu sistem penjadwalan yang baik.

Penjadwalan merupakan proses pengalokasian sekumpulan pekerjaan yang akan

*Korespondensi Penulis

dikerjakan dengan menggunakan sumber daya yang terbatas yang berhubungan dengan kegiatan produksi (Baker, 2001). Beberapa tujuan yang ingin dicapai dalam penjadwalan antara lain meningkatkan utilisasi sumber daya, mengurangi inventori dalam proses, mengurangi keterlambatan, dan meminimasi waktu total pengerjaan seluruh pekerjaan yang ada (Bedworth & Bailey, 1987).

Dalam perusahaan yang menjalankan strategi *make to order*, salah satu masalah dalam penjadwalan adalah total waktu penyelesaian sekumpulan pesanan (*makespan*). Waktu penyelesaian kumpulan pesanan pertama menentukan kapan kumpulan pesanan berikutnya dapat dimulai untuk dikerjakan. Mengingat faktor kepuasan konsumen adalah kunci sukses dalam memenangkan persaingan, maka total waktu penyelesaian kumpulan pesanan ini perlu diperhatikan.

Banyak perusahaan manufaktur yang menggunakan sistem *job shop*, dimana aliran pekerjaan mengalir melewati sejumlah mesin yang berbeda-beda dengan urutan yang berbeda-beda pula. Urutan mesin harus sesuai dengan presedensi operasi yang telah ditetapkan dan jumlah mesin yang digunakan pun terbatas. Kendala presedensi antar operasi dan keterbatasan mesin ini mengakibatkan proses penjadwalan *job shop* menjadi semakin kompleks.

Masalah penjadwalan *job shop* termasuk ke dalam kategori masalah optimasi kombinatorial, yang banyak diselesaikan menggunakan metode metaheuristik. Metode metaheuristik yang digunakan dalam masalah penjadwalan *job shop* di antaranya adalah *Ant Colony System* (Zhang et al., 2006), *Genetic Algorithms* (Shah, 2004), dan *Artificial Immune Systems* (Weckman et al., 2012).

Consultant-Guided Search merupakan salah satu metode metaheuristik yang relatif baru. Dikemukakan pertama kali oleh Iordache (2010), CGS merupakan algoritma yang terinspirasi dari cara orang membuat keputusan berdasarkan saran yang diterima dari konsultan. Iordache (2010) menerapkan CGS dalam memecahkan masalah *Travelling Salesman Problem* dan menemukan bahwa solusi yang dihasilkan CGS dapat sebanding atau lebih baik dibandingkan dengan *Ant Colony System* dan *Max-Min Ant System*.

Penelitian ini bertujuan untuk menerapkan algoritma CGS (Iordache, 2010) untuk menyelesaikan masalah penjadwalan *job shop* (*Job Shop Scheduling Problem/JSSP*), dengan kriteria meminimasi *makespan*. Pada waktu yang bersamaan, Deepanandhini & Amudha (2013) menerapkan algoritma CGS untuk menyelesaikan masalah yang sama.

Algoritma yang dikembangkan dalam penelitian ini (selanjutnya disebut algoritma CGS-JSSP) akan dievaluasi performansinya dalam menghasilkan solusi optimal, dan juga akan dibandingkan dengan algoritma metaheuristik lainnya. Selain itu, solusi yang dihasilkan oleh algoritma yang dikembangkan juga akan dibandingkan dengan algoritma CGS dari Deepanandhini & Amudha (2013).

2 Metode Penelitian

Penerapan CGS dalam menyelesaikan masalah penjadwalan *job shop* dilakukan dalam langkah-langkah berikut:

1. Perancangan algoritma CGS-JSSP
   Pada langkah ini dirancang algoritma CGS-JSSP, termasuk di dalamnya adalah algoritma pembentukan strategi, algoritma pembentukan solusi, algoritma pembaharuan reputasi dan algoritma pembaharuan status.
2. Verifikasi dan validasi algoritma
   Langkah ini bertujuan untuk memastikan algoritma CGS-JSSP yang dirancang telah sesuai dengan konsep CGS dan mampu menggambarkan dan menyelesaikan permasalahan penjadwalan *job shop*.
3. Studi komputasional
   Algoritma yang dirancang diujikan pada 10 buah kasus yang diambil dari penelitian Weckman et al. (2012). Studi komputasional ini dilakukan dengan bantuan perangkat lunak yang dirancang khusus sesuai algoritma CGS-JSSP yang dihasilkan. Pada langkah ini dicatat nilai *makespan* terbaik yang dihasilkan pada masing-masing kasus. Sebelum hasil yang diperoleh digunakan pada tahap selanjutnya, dilakukan pengujian verifikasi dan validasi perangkat lunak yang digunakan.
4. Evaluasi performansi algoritma
   Algoritma CGS-JSSP yang dirancang dievaluasi kemampuannya menghasilkan solusi optimal. Selain itu algoritma yang dikembangkan juga dibandingkan dengan algoritma *Artificial Immune Systems* (Weckman et al., 2012) dan *Genetic Algorithms* (Shah, 2004) dalam menyelesaikan kasus-kasus yang sama. Perbedaan solusi yang dihasilkan dengan solusi algoritma CGS dari Deepanandhini & Amudha (2013) juga akan dibahas pada langkah ini.

3 *Consultant-Guided Search*

Consultant-Guided Search (CGS) merupakan salah satu metode metaheuristik dengan konsep *swarm intelligence* yang ditujukan untuk memecahkan masalah *combinatorial optimization*. Algoritma ini terinspirasi dari cara orang dalam membuat keputusan berdasarkan rekomendasi yang diterimanya dari konsultan. Algoritma yang berdasarkan *population-based method* ini dikemukakan oleh Iordache (2010).

Dalam algoritma CGS, setiap orang (*virtual person*) akan berperan sebagai klien dan konsultan secara bersamaan. Sebagai klien, mereka membentuk sebuah solusi dari setiap iterasi dalam permasalahan yang ada. Sebagai konsultan, mereka memberikan saran kepada kliennya sesuai dengan strategi konsultan tersebut.

Di setiap langkah dalam sebuah iterasi, seorang klien akan menerima saran dari konsultan yang terpilih pada iterasi tersebut dalam membentuk solusinya. Pembentukan solusi ini merupakan proses stokastik, yaitu terdapat probabilitas tertentu dimana klien tidak selalu harus mengikuti saran dari konsultan.

Dalam CGS, solusi dari suatu permasalahan direpresentasikan dalam urutan komponen - komponennya (*sequence of components*). Pembentukan solusi dalam CGS merupakan *multi-step process*. Pada setiap langkah, komponen baru ditambahkan ke dalam solusi. Sebagai contoh dalam permasalahan *Travelling Salesman Problem* (TSP), solusi direpresentasikan dengan perjalanan (*tour*) dengan komponennya berupa kota-kota yang harus dikunjungi. Pada setiap langkah pembuatan solusi, sebuah keputusan perlu diambil, yaitu untuk menentukan komponen mana yang akan dipilih selanjutnya. Dalam kasus TSP, keputusan yang perlu diambil adalah memilih kota yang akan dikunjungi selanjutnya.

Di awal setiap iterasi, seorang klien memilih seorang konsultan berdasarkan preferensi personal klien dan reputasi konsultan tersebut. Dengan kata lain, seorang klien harus memiliki seorang konsultan. Dalam memilih konsultan, pengaplikasian preferensi personal klien dan reputasi konsultan tersebut dalam CGS disesuaikan dengan spesifikasi masalah-masalah yang ingin diselesaikan. Preferensi personal ini merupakan faktor lain yang diperhitungkan selain reputasi konsultan dalam memilih konsultan. Seorang klien dapat saja memilih konsultan dengan reputasi yang buruk, dikarenakan faktor kedekatan personal dengan konsultan tersebut.

Reputasi konsultan bertambah seiring dengan jumlah kesuksesan yang dicapai dari para kliennya. Seorang klien dikatakan sukses jika klien tersebut membentuk solusi yang lebih baik dari semua solusi yang ada. Setiap kali klien mencapai kesuksesan, konsultan tersebut perlu mengatur kembali strateginya, yaitu mengganti strateginya dengan strategi dari klien yang mencapai sukses tersebut. Jika solusi yang didapatkan dari klien lebih baik dari solusi dari iterasi sebelumnya, maka konsultan yang memberikan saran kepada klien tersebut akan mendapatkan tambahan bonus reputasi. Reputasi dari konsultan akan hilang seiring berjalannya waktu.

Untuk tetap mempertahankan reputasinya, konsultan perlu memastikan kliennya agar tetap sukses. Jika reputasi dari seorang konsultan berada di bawah nilai minimum reputasi, maka konsultan tersebut akan cuti panjang (*sabbatical leave*). Beberapa konsultan mungkin saja memiliki reputasi yang lebih baik dibandingkan konsultan lainnya. Selama iterasi dalam algoritma CGS ini, konsultan-konsultan terbaik (*best consultants*), yaitu konsultan memiliki reputasi yang relatif tinggi, memiliki probabilitas yang cukup besar untuk dipilih oleh klien-klien yang ada. Sebaliknya, konsultan-konsultan yang memiliki reputasi buruk (*poor consultants*) mungkin saja tidak mendapatkan klien sama sekali.

Algoritma CGS ini juga mempertahankan - peringkat (*ranking*) dari para konsultan yang telah memberikan solusi terbaik kepada kliennya. Untuk para konsultan yang berada pada peringkat teratas, algoritma ini mencegah reputasi mereka berada di bawah nilai minimum reputasi.

Dalam masa cuti panjang, konsultan akan berhenti memberikan saran kepada para klien dan mulai mencari strategi baru yang akan digunakan di kemudian waktu. Setelah periode waktu yang telah ditentukan sebelumnya, masa cuti berakhir dan orang akan memilih strategi terbaik yang ditemukan dalam periode waktu tersebut. Strategi terbaik ini kemudian akan digunakan untuk memandu para klien. Di akhir masa cuti panjang, reputasi konsultan akan diatur kembali pada nilai reputasi yang telah ditentukan sebelumnya.

4 Algoritma CGS-JSSP

Algoritma CGS-JSSP yang dirancang terdiri atas:

1. Algoritma CGS-JSSP secara keseluruhan
2. Algoritma pembentukan strategi (Algoritma A)
3. Algoritma pembentukan solusi (Algoritma B)

4. Algoritma pembaharuan reputasi (Algoritma C)
5. Algoritma pembaharuan status (Algoritma D)

5 Algoritma CGS-JSSP Keseluruhan

Algoritma CGS-JSSP secara keseluruhan terdiri atas langkah-langkah berikut:

1. *Input* jumlah *job*, jumlah mesin, matriks *routing*, matriks waktu proses, dan parameter CGS, yaitu jumlah *virtual person* (P), lamanya status *sabbatical* (SP), nilai awal reputasi (*initrep*), nilai maksimum reputasi (*maxrep*), nilai minimum reputasi (*minrep*), parameter reputasi (α), parameter preferensi personal (β), parameter strategi (A), parameter solusi (B), probabilitas mengikuti strategi konsultan (Q), nilai tambahan reputasi (*bon*), nilai *reputation fading* (r), jumlah *protected top consultants* (*proranks*), dan jumlah iterasi (*itermax*).
2. Atur:
   (a) Nilai awal *makespan* strategi terbaik dengan nilai yang sangat besar.
   (b) Kosongkan himpunan strategi terbaik untuk seluruh *virtual person*.
   (c) Nilai awal makespan solusi terbaik dengan nilai yang sangat besar.
   (d) Iterasi awal adalah 1.
   (e) Status seluruh *virtual person* adalah *sabbatical*.
   (f) Durasi status *sabbatical* untuk seluruh *virtual person* adalah sebesar lamanya status *sabbatical*.
   (g) Kosongkan himpunan konsultan yang tersedia.
   (h) Kosongkan himpunan solusi terbaik.
3. Tugaskan *virtual person* pertama.
4. Periksa apakah status *virtual person* adalah *sabbatical*. Jika ya, lanjut ke langkah 5. Jika tidak, lanjut ke langkah 14.
5. *Virtual person* akan membentuk strategi (Algoritma A).
6. Periksa apakah seluruh *virtual person* sudah ditugaskan. Jika sudah, lanjut ke langkah 8. Jika belum, lanjut ke langkah 7.
7. Tugaskan *virtual person* selanjutnya, lalu kembali ke langkah 4.
8. *Virtual person* memperbaharui reputasinya (Algoritma C).
9. *Virtual person* memperbaharui statusnya (Algoritma D).
10. Periksa apakah himpunan konsultan yang tersedia merupakan himpunan kosong. Jika ya, lanjut ke langkah 12. Jika tidak, lanjut ke langkah 11.
11. Hitung probabilitas memilih konsultan untuk setiap konsultan yang ada dalam himpunan konsultan yang tersedia (p_z):

$$p_z = \frac{rep_z^{\alpha} \cdot \left[\frac{1}{bestMStr_z}^{\beta}\right]}{\sum_{z \varepsilon Econs} rep_z^{\alpha} \cdot \left[\frac{1}{bestMStr_z}^{\beta}\right]} \quad (1)$$

   dimana:
   p_z : probabilitas memilih konsultan ke-z
   rep_z : reputasi konsultan ke-z
   $bestMStr_z$: *makespan* terbaik dari konsultan z
   α : parameter reputasi
   β : parameter preferensi personal
   Econs : himpunan konsultan yang tersedia untuk dipilih
   z : indeks orang (*virtual person*) sebagai konsultan yang tersedia
12. Periksa apakah iterasi sekarang sudah sama dengan iterasi maksimum. Jika sudah, lanjut ke langkah 16. Jika belum, lanjut ke langkah 13.
13. Tambahkan jumlah iterasi, lalu kembali ke langkah 3.
14. *Virtual person* memilih konsultan.
15. *Virtual person* membentuk solusi (Algoritma B), lalu kembali ke langkah 6.
16. Tugaskan *virtual person* pertama.
17. Periksa apakah status *virtual person* adalah *sabbatical*. Jika ya, lanjut ke langkah 18. Jika tidak, lanjut ke langkah 20.
18. Periksa apakah makespan strategi terbaik *virtual person* lebih kecil daripada *makespan* terbaik. Jika ya, lanjut ke langkah 19. Jika tidak, lanjut ke langkah 20.
19. Atur *makespan* terbaik digantikan oleh *makespan* strategi terbaik *virtual person* dan solusi terbaik digantikan oleh strategi terbaik *virtual person*.
20. Periksa apakah seluruh *virtual person* sudah ditugaskan. Jika sudah, lanjut ke langkah 22. Jika belum, lanjut ke langkah 21.
21. Tugaskan *virtual person* selanjutnya, lalu kembali ke langkah 17.
22. Tampilkan solusi akhir yaitu *makespan* terbaik dan solusi terbaik.
23. Selesai.

6 Algoritma Pembentukan Strategi

Berikut adalah algoritma pembentukan strategi (Algoritma A):

1. Atur:
   (a) Kosongkan himpunan strategi *virtual person*.
   (b) Nilai *completion time* untuk semua pekerjaan ke-*i* dan operasi ke-*j* adalah 0.
   (c) Nilai *ready time* untuk semua mesin *k* adalah 0.
2. Atur indeks operasi dengan nilai 1.
3. Atur:
   (a) Masukkan operasi *ijk* ke dalam *EO*.
   (b) Nilai O sebanyak jumlah operasi *ijk* dalam *EO*.
4. Bangkitkan bilangan random a[0,1].
5. Periksa apakah nilai a lebih kecil daripada parameter pembentukan strategi (*A*). Jika ya, lanjut ke langkah 6. Jika tidak, lanjut ke langkah 17.
6. Atur indeks operasi yang tersedia dalam *EO* (o) dengan nilai 1.
7. Ambil nilai i, j, k dari operasi *ijk* pada nilai o pada *EO*.
8. Hitung *ready time* operasi terkecil.
9. Hitung *completion time* untuk pekerjaan ke-*i* operasi ke-*j*.
10. Periksa apakah indeks operasi sudah sama dengan jumlah operasi yang ada pada *EO*. Jika sudah, lanjut ke langkah 12. Jika belum, lanjut ke langkah 11.
11. Tambahkan nilai indeks operasi, lalu kembali ke langkah 7.
12. Hitung *completion time* terkecil,
13. Ambil operasi *ijk* terpilih dengan nilai *completion time* yang paling kecil, lalu masukkan ke dalam strategi *virtual person* (str_p).
14. Perbaharui nilai *ready time* mesin k (R_k) - dengan nilai *completion time* dari operasi *ijk* yang terpilih (C_{ij}).
15. Hapus operasi *ijk* yang sudah terpilih dari *EO*, kemudian perbaharui *EO* dengan menambahkan operasi setelah operasi *ijk* yang sudah terpilih dan jumlah operasi yang terdapat dalam *EO*.
16. Periksa apakah *EO* merupakan himpunan kosong. Jika ya, lanjut ke langkah 21. Jika tidak, kembali ke langkah 4.
17. Ambil operasi *ijk* secara acak dari *EO*, lalu masukkan ke dalam strategi *virtual person* (str_p).
18. Ambil nilai *i*, *j*, *k* dari operasi *ijk* yang terpilih.
19. Hitung *ready time* operasi terkecil.
20. Hitung *completion time* untuk pekerjaan ke-*i* operasi ke-*j*, lalu kembali ke langkah 14.
21. Hitung *makespan* dari operasi *ijk* yang ada dalam strategi *virtual person* (str_p).
22. Periksa apakah *makespan* strategi *virtual person* ($MStr_p$) lebih kecil daripada *makespan* strategi terbaik *virtual person* ($bestMStr_p$). Jika ya, lanjut ke langkah 23. Jika tidak, selesai.
23. Atur:
    (a) *Makespan* strategi terbaik *virtual person* digantikan oleh *makespan* strategi *virtual person* ($bestMStr_p = MStr_p$).
    (b) Strategi terbaik *virtual person* digantikan oleh strategi *virtual person* yang telah dibentuk ($bestr_p = str_p$).
24. Selesai.

7 Algoritma Pembentukan Solusi

Berikut adalah algoritma pembentukan solusi (Algoritma B):

1. Atur:
   (a) Kosongkan himpunan solusi *virtual person* (sol_p).
   (b) Kosongkan nilai konsultan yang akan mendapatkan bonus reputasi (*bon*).
   (c) Nilai *completion time* (C_{ij}) untuk semua pekerjaan ke-*i* dan operasi ke-*j* adalah 0.
   (d) Nilai *ready time* untuk semua mesin *k* (R_k) adalah 0.
2. Atur indeks operasi dengan nilai 1 ($j = 1$).
3. Atur:
   (a) Masukkan operasi *ijk* ke dalam *EO*.
   (b) Nilai O sebanyak jumlah operasi *ijk* dalam *EO*.
4. Bangkitkan bilangan random q[0,1].
5. Periksa apakah nilai q lebih kecil daripada parameter probabilitas mengikuti rekomendasi konsultan (Q). Jika ya, lanjut ke langkah 6. Jika tidak, lanjut ke langkah 16.
6. Ambil dan lihat strategi terbaik konsultan (*z*) yang telah dipilih oleh *virtual person* ($bestr_z$).

7. Periksa apakah sol_p merupakan himpunan kosong. Jika ya, lanjut ke langkah 8. Jika tidak, lanjut ke langkah 9.
8. Ambil operasi *ijk* pertama dalam strategi terbaik konsultan ($bestr_z$), lalu masukkan ke dalam solusi *virtual person* (sol_p), lalu lanjut ke langkah 30.
9. Atur x sebagai operasi terakhir yang terdapat dalam solusi *virtual person* (sol_p).
10. Periksa apakah terdapat satu operasi sebelum dan/atau sesudah x dalam strategi terbaik konsultan ($bestr_z$). Jika tidak ada, lanjut ke langkah 15. Jika minimal ada satu operasi, periksa apakah satu/lebih operasi tersebut ada dalam *EO*. Jika ada, lanjut ke langkah 11. Jika tidak ada, lanjut ke langkah 15.
11. Atur Y sebagai jumlah rekomendasi operasi yang tersedia.
12. Periksa apakah nilai Y adalah 1. Jika ya, lanjut ke langkah 13. Jika tidak, lanjut ke langkah 14.
13. Ambil rekomendasi operasi, lalu masukkan ke dalam himpunan sol_p, lalu lanjut ke langkah 30.
14. Ambil satu dari dua rekomendasi operasi *ijk* secara acak, lalu masukkan ke dalam himpunan sol_p, lalu lanjut ke langkah 30.
15. Periksa apakah operasi kedua sebelum dan/atau sesudah x terdapat dalam $bestr_z$. Jika tidak ada, lanjut ke langkah 16. Jika minimal ada satu operasi, periksa apakah satu/lebih operasi tersebut ada dalam *EO*. Jika ada, kembali ke langkah 11. Jika tidak ada, lanjut ke langkah 16.
16. Bangkitkan bilangan random b[0,1].
17. Periksa apakah nilai b lebih kecil daripada parameter pembentukan solusi (B). Jika ya, lanjut ke langkah 18. Jika tidak, lanjut ke langkah 29.
18. Atur indeks operasi (o) yang tersedia dalam *EO* dengan nilai 1.
19. Ambil nilai i, j, k dari operasi *ijk* pada nilai o pada *EO*.
20. Hitung *ready time* operasi terkecil.
21. Hitung *completion time* untuk pekerjaan ke-i operasi ke-j.
22. Periksa apakah indeks operasi sudah sama dengan jumlah operasi yang ada pada *EO*. Jika sudah, lanjut ke langkah 24. Jika belum, lanjut ke langkah 23.
23. Tambahkan nilai indeks operasi, lalu kembali ke langkah 18.
24. Hitung *completion time* terkecil.
25. Ambil operasi *ijk* terpilih dengan nilai *completion time* yang paling kecil, lalu masukkan ke dalam solusi *virtual person* (sol_p).
26. Perbaharui nilai *ready time* mesin k dengan nilai *completion time* dari operasi *ijk* yang terpilih ($R_k = C_{ij}$).
27. Hapus operasi *ijk* yang sudah terpilih dari *EO*, kemudian perbaharui *EO* dengan menambahkan operasi setelah operasi *ijk* yang sudah terpilih dan jumlah operasi yang terdapat dalam *EO*.
28. Periksa apakah *EO* merupakan himpunan kosong. Jika ya, lanjut ke langkah 33. Jika tidak, kembali ke langkah 4.
29. Ambil operasi *ijk* secara acak dari *EO*, lalu masukkan ke dalam solusi *virtual person* (sol_p).
30. Ambil nilai i, j, k dari operasi *ijk* yang terpilih.
31. Hitung *ready time* operasi terkecil.
32. Hitung *completion time* untuk pekerjaan ke-i operasi ke-j, lalu kembali ke langkah 26.
33. Hitung *makespan* dari operasi *ijk* yang ada dalam solusi *virtual person* (sol_p).
34. Periksa apakah *makespan* solusi *virtual person* lebih kecil daripada *makespan* strategi terbaik konsultannya ($MSol_p < bestMStr_z$). Jika ya, lanjut ke langkah 35. Jika tidak, lanjut ke langkah 37.
35. Atur:
    (a) *Makespan* strategi terbaik konsultan digantikan oleh *makespan* solusi *virtual person* ($bestMStr_z = MSol_p$).
    (b) Strategi terbaik konsultan digantikan oleh solusi *virtual person* yang telah dibentuk ($bestr_z = sol_p$).
36. Tambahkan jumlah sukses konsultan ($succ_z$) dengan nilai 1.
37. Periksa apakah *makespan* strategi terbaik konsultan lebih kecil daripada nilai *makespan* terbaik ($bestMStr_z < bestMS$). Jika ya, lanjut ke langkah 38. Jika tidak, selesai.
38. Atur:
    (a) *Makespan* solusi terbaik digantikan oleh *makespan* strategi terbaik konsultan ($bestMS = bestMStr_z$).
    (b) Solusi terbaik digantikan oleh strategi terbaik konsultan ($best = bestr_z$).
39. Simpan dan perbaharui konsultan z dalam nilai konsultan yang akan mendapatkan bonus reputasi (*bon*).
40. Selesai.

8 Algoritma Pembaharuan Reputasi

Berikut adalah algoritma pembaharuan reputasi (Algoritma C):

1. Periksa apakah himpunan konsultan yang tersedia merupakan himpunan kosong. Jika ya, selesai. Jika tidak, lanjut ke langkah 2.
2. Tugaskan konsultan pertama.
3. Hitung reputasi konsultan :

$$rep = rep(1 - r) \quad (2)$$

   dimana *rep* merupakan reputasi konsultan dan (*r*) merupakan reputation *fading*.
4. Hitung reputasi yang didapatkan oleh konsultan berdasarkan jumlah sukses kliennya.
5. Periksa apakah konsultan merupakan konsultan yang berhasil menggantikan solusi akhir dengan strategi terbaiknya. Jika ya, lanjut ke langkah 6. Jika tidak, lanjut ke langkah 7.
6. Tambahkan bonus reputasi.
7. Periksa apakah reputasi konsultan lebih besar daripada nilai maksimum reputasi. Jika ya, lanjut ke langkah 8. Jika tidak, lanjut ke langkah 9.
8. Atur reputasi konsultan sebesar nilai maksimum reputasi.
9. Periksa apakah seluruh konsultan sudah ditugaskan. Jika sudah, lanjut ke langkah 11. Jika belum, lanjut ke langkah 10.
10. Tugaskan konsultan berikutnya, lalu kembali ke langkah 3.
11. Urutkan konsultan berdasarkan strategi terbaiknya.
12. Tambahkan konsultan ke dalam himpunan konsultan terbaik sejumlah konsultan yang dilindungi.
13. Selesai.

9 Algoritma Pembaharuan Status

Berikut adalah algoritma pembaharuan status (Algoritma D):

1. Tugaskan *virtual person* pertama.
2. Periksa apakah status *virtual person* adalah *normal*. Jika ya, lanjut ke langkah 7. Jika tidak, lanjut ke langkah 3.
3. Status *sabbatical virtual person* dikurangi sebesar 1.
4. Periksa apakah status *sabbatical* sudah berakhir. Jika ya, lanjut ke langkah 5. Jika tidak, lanjut ke langkah 13.
5. Tambahkan *virtual person* ke dalam himpunan konsultan yang tersedia.
6. Ubah status *virtual person* menjadi *normal* dan beri nilai reputasi sebesar nilai awal reputasi, lalu lanjut ke langkah 13.
7. Periksa apakah reputasi lebih kecil daripada nilai minimum reputasi. Jika ya, lanjut ke langkah 8. Jika tidak, lanjut ke langkah 13.
8. Periksa apakah *virtual person* termasuk dalam himpunan konsultan terbaik. Jika ya, lanjut ke langkah 9. Jika tidak, lanjut ke langkah 10.
9. Atur nilai reputasi sebesar nilai awal reputasi, lalu lanjut ke langkah 13.
10. Hilangkan *virtual person* dalam himpunan konsultan yang tersedia.
11. Ubah status *virtual person* menjadi *sabbatical*, hilangkan strategi terbaik konsultan, dan atur durasi status *sabbatical* sebesar lamanya status *sabbatical*, lalu lanjut ke langkah 13.
12. Tugaskan *virtual person* selanjutnya.
13. Periksa apakah seluruh *virtual person* sudah ditugaskan. Jika sudah, selesai. Jika belum, kembali ke langkah 12.

10 Studi Komputasional

Studi komputasional dilakukan menggunakan 10 buah kasus permasalahan penjadwalan *job shop* yang diambil dari penelitian Weckman et al. (2012). Data lengkap untuk setiap kasus diambil dari penelitian Bondal (2008). Kumpulan kasus tersebut terdiri atas beragam level kompleksitas, dimulai dari kasus dengan 6 *job* dan 6 mesin hingga 10 *job* dan 10 mesin, seperti tampak pada Tabel 1.

Algoritma CGS-JSSP yang dikembangkan diujikan pada 10 kasus tersebut. Nilai parameter yang digunakan dalam pengujian ini dapat dilihat pada Tabel 2. Untuk setiap kasus, dilakukan dalam 500 iterasi dengan 5 kali replikasi.

Nilai *makespan* terbaik (*best known solution*) yang dihasilkan algoritma CGS-JSSP dicatat dan dibandingkan dengan *makespan* terbaik dari GA (Shah, 2004) dan algoritma AIS (Weckman et al., 2012), serta dibandingkan pula dengan solusi optimal. Hasil selengkapnya dapat dilihat pada Tabel 3.

Berdasarkan hasil studi komputasional, ditemukan bahwa algoritma CGS-JSSP dapat mendapatkan solusi optimal pada 3 kasus, yaitu

Tabel 1: Kasus yang diujikan

No	Kasus	Job x mesin
1	FT06	6 x 6
2	LA01	10 x 5
3	LA02	10 x 5
4	LA03	10 x 5
5	LA04	10 x 5
6	LA05	10 x 5
7	LA16	10 x 10
8	FT10	10 x 10
9	ABZ5	10 x 10
10	ABZ6	10 x 10

Tabel 2: Nilai parameter CGS yang digunakan

No	Parameter CGS	Nilai
1	Jumlah *virtual person* (*P*)	1000
2	Lamanya status *sabbatical* (*SP*)	50
3	Nilai awal reputasi (*initrep*)	15
4	Nilai maksimum reputasi (*maxrep*)	40
5	Nilai minimum reputasi (*minrep*)	1
6	Parameter reputasi (α)	1 & 5
7	Parameter preferensi personal (β)	1 & 5
8	Parameter strategi (*A*)	0,3 & 0,7
9	Parameter solusi (*B*)	0,3 & 0,7
10	Probabilitas mengikuti strategi konsultan (Q)	0,3 & 0,7
11	Nilai tambahan reputasi (*bon*)	10
12	Nilai *reputation fading* (*r*)	0,1 & 0,5
13	Jumlah *protected top consultants* (*proranks*)	10

FT06, LA01 dan LA05. Pada kasus lainnya, yaitu LA02, LA03, LA04, LA16, FT10, ABZ5 dan ABZ6 selisih antara *makespan* CGS-JSSP dan optimal masing-masing adalah 1,22%; 4,02%; 1,36%; 4,76%; 6,77%; 0,32%, dan 3,08%, dengan rata-rata sebesar 2,15%.

Dibandingkan dengan algoritma GA (Shah, 2004), algoritma CGS-JSSP sebanding di 3 kasus yaitu FT06, LA01 dan LA05. Dari Tabel 3 terlihat bahwa GA dan CGS-JSSP mampu menghasilkan solusi optimal pada ketiga kasus tersebut, dan CGS-JSSP unggul dari GA pada 7 kasus lainnya. Dibandingkan dengan algoritma AIS (Weckman et al., 2012), CGS-JSSP sebanding dengan AIS di kasus FT06 dan LA05 (mampu mencapai solusi optimal) dan unggul di 8 kasus lainnya.

Dari 10 kasus yang diujikan pada penelitian ini, terdapat 9 kasus yang juga diujikan pada Algoritma CGS Deepanandhini & Amudha (2013). Pengecualian terdapat pada kasus FT06, sehingga perbandingan performansi akan dilakukan untuk kasus LA01-LA05, LA16, FT10, ABZ5 dan ABZ6. *Makespan* yang diperoleh untuk ke-9 kasus tersebut berturut-turut adalah 713, 752, 682, 669, 593, 1010, 1044, 1347 dan 998 (Deepanandhini & Amudha, 2013).

Tabel 3: Solusi optimal, CGS-JSSP, GA dan AIS

No	Kasus	*Makespan*			
		Optimal	CGS	GA	AIS
1	FT06	55	55	55	55
2	LA01	666	666	666	702
3	LA02	655	663	716	708
4	LA03	597	621	638	672
5	LA04	590	598	619	644
6	LA05	593	593	593	593
7	LA16	945	990	1033	1124
8	FT10	930	993	1099	1208
9	ABZ5	1234	1238	1339	1434
10	ABZ6	943	972	1043	1084

Dibandingkan dengan algoritma CGS Deepanandhini & Amudha (2013), algoritma CGS-JSSP sebanding pada kasus LA05 (mampu mencapai solusi optimal) dan unggul pada 8 kasus lainnya. Perbedaan kualitas solusi yang dihasilkan oleh kedua algoritma ini dapat disebabkan oleh penggunaan nilai parameter yang berbeda. Selain itu perbedaan juga dapat diakibatkan oleh proses pembentukan strategi, dimana dalam penelitian ini dilakukan dengan dua cara, yaitu penggunaan bilangan acak dan pembentukan jadwal aktif. Pembentukan jadwal aktif dilakukan karena himpunan jadwal aktif merupakan himpunan terkecil yang mengandung jadwal optimal (Baker, 2001).

11 Kesimpulan

Penelitian ini menerapkan algoritma CGS pada masalah penjadwalan *job shop* untuk meminimasi makespan. Berdasarkan studi komputasional pada 10 kasus yang diuji, dapat disimpulkan bahwa algoritma CGS-JSSP yang dihasilkan dapat digunakan untuk memecahkan masalah penjadwalan *job shop*.

Kemampuan algoritma CGS-JSSP dalam mencapai solusi optimal cukup baik, dimana solusi optimal dapat dicapai pada 3 dari 10 kasus, dengan rata-rata selisih antara solusi yang dicapai algoritma yang dirancang dengan nilai optimal sebesar 2,15% untuk kesuluruhan kasus. Perbandingan solusi antara algoritma CGS-JSSP yang dirancang dengan metode metaheuritistik lainnya juga menunjukkan bahwa algoritma CGS-JSSP memiliki performansi yang unggul dalam memecahkan masalah penjadwalan *job shop*. Hal ini membuat penerapan algoritma

CGS pada masalah-masalah optimasi kombinatorial lainnya menjadi semakin menarik untuk diteliti.

Evaluasi performansi dalam penelitian ini dilakukan menggunakan nilai *makespan* terbaik yang dicapai, sesuai dengan data pembanding yang tersedia pada literatur yang diacu. Nilai rata-rata atau modus dari solusi yang diperoleh akan dapat memberikan hasil evaluasi yang lebih baik. Selain itu, penelitian lebih lanjut juga dapat menggunakan kriteria minimasi keterlambatan (*tardiness*), yang juga merupakan masalah yang dihadapi oleh perusahaan berstrategi *make to order* dengan sistem produksi *job shop*.

Daftar Pustaka

Baker, K.R. (2001). *Elements of Sequencing and Scheduling*. John Wiley & Sons.

Bedworth D.D, & Bailey J.E. (1987). *Integrated Production Control System*. John Wiley & Sons.

Bondal, A.A. (2008). *Artificial Immune Systems Applied to Job Shop Scheduling, Industrial and Systems Engineering*. Masters Thesis. Department of Industrial and Manufacturing Systems Engineering, Ohio University, Ohio.

Deepanandhini, D. & Amudha, T. (2013) *Solving Job Shop Scheduling Problems with Consultant Guided Search Metaheuristics*. International Journal of Software and Web Sciences, Vol. 3(1), 01-06.

Iordache, S. (2010). *Consultant-Guided Search A New Metaheuristic for Combinatorial Optimization Problems*. Proceedings of the 12th Genetic and Evolutionary Computation Conference 2010. ACM Press, New York, 225-232.

Morton, T.E. & Pentico, D.W. (1993). *Heuristic Scheduling Systems*. John Wiley & Sons.

Shah, N. (2004). *Using distributed computing to improve the performance of genetic algorithms for job shop scheduling problems*. Masters Thesis. Department of Industrial and Manufacturing Systems Engineering, Ohio University, Ohio.

Weckman, G., Bondal, A.A., Rinder, M.M., & Young, W.A. (2012). *Applying A Hybrid Artificial Immune Systems to the job shop scheduling problem*. Neural Computing & Applications, Vol. 21, 1465-1475.

Zhang, J., Hu, X., Tan, X., Zhong, J.H., & Huang, Q. (2006). *Implementation of an Ant Colony Optimization Technique for Job Shop Scheduling Problem*. Transactions of the Institute of Measurement and Control, Vol. 28, 93-108.