PENERAPAN ALGORITMA *TABU SEARCH* PADA PERMASALAHAN LINTASAN KESEIMBANGAN BENTUK U TIPE I DENGAN WAKTU PROSES STOKASTIK

Hotna Marina Sitorus; Cynthia P. Juwono; Pauline Ciputra

Industrial Engineering Department, Faculty of Engineering, Universitas Katolik Parahyangan
Jl. Ciumbuleuit No. 94 Bandung 40141
Email: nina@ unpar.ac.id, juwonocp@unpar.ac.id

ABSTRACT

This paper describes the application of Tabu Search to solve a U-shaped line balancing problem which seeks to obtain the minimum cycle time using the number of workers available (type I line balancing problem). To accommodate the differences between operators, this paper uses stochastic processing times. The performance of the proposed algorithm is analysed using various hypothetical cases. The cases are different in terms of the complexity of presedensi *structure, the desired cycle times, and the standard deviation of processing times. This paper also studies the impact of the Tabu Search parameters on the performance of the proposed algorithm. Experimental results show that the proposed algorithm is superior to Maximum Ranked Positional Weight method in all the cases and to Ant Colony System-based algorithm in several cases. This paper also finds that in several cases, the performance of the proposed algorithm is not influenced by the parameters of Tabu Search.*

Keywords: *u-shaped line balancing, tabu search, stochastic processing times*

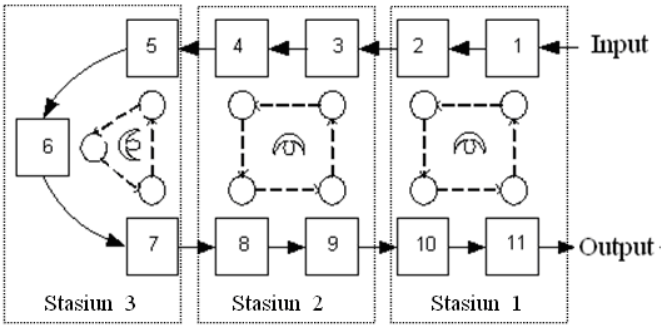
ABSTRAK

Penelitian ini membahas penerapan Tabu Search dalam memecahkan permasalahan keseimbangan lintasan bentuk U dengan tujuan meminimasi waktu siklus lintasan (masalah keseimbangan lintasan tipe I). Untuk mengakomodir perbedaan performansi operator stasiun kerja, penelitian ini menggunakan waktu proses stokastik. Performansi algoritma yang dikembangkan dianalisis menggunakan beberapa kasus hipotetik. Kasus-kasus tersebut berbeda dalam hal kompleksitas struktur presedensi, waktu siklus yang diinginkan dan standar deviasi waktu proses. Penelitian ini juga menganalisis sensitivitas performansi algoritma yang dikembangkan terhadap parameter Tabu Search. Berdasarkan eksperimen yang dilakukan diketahui bahwa algoritma yang dikembangkan memiliki performansi yang lebih baik dibandingkan dengan metode Maximum Ranked Positional Weight di seluruh kasus dan lebih baik dibandingkan algoritma berbasis Ant Colony System di beberapa kasus. Selain itu juga ditemukan bahwa pada beberapa kasus performansi algoritma yang dikembangkan tidak dipengaruhi oleh parameter Tabu Search yang digunakan.

Kata kunci: *keseimbangan lintasan bentuk U, tabu search, waktu proses stokastik*

PENDAHULUAN

Seiring dengan meningkatnya penerapan sistem *Just In Time,* semakin banyak perusahaan yang mengubah lintasan produksi mereka menjadi lintasan bentuk U. Lintasan bentuk U memiliki karakteristik utama yaitu aliran produk akan berbentuk huruf U, seperti ditunjukkan pada Gambar 1. Lintasan yang pertama kali diperkenalkan oleh Monden (1993) ini memungkinkan operasi pertama dan operasi terakhir dilakukan dalam satu stasiun yang sama. Hal ini membuat satu unit material akan masuk ke dalam lintasan pada saat satu unit material keluar, dan karena hal ini dilakukan oleh operator yang sama, besarnya *work-in-process* (WIP) dalam sistem dapat diusahakan selalu konstan (Monden, 1993). Selain itu lintasan bentuk U juga dapat meningkatkan visibilitas dan komunikasi antar operator, meningkatkan keragaman keahlian operator, menambah fleksibilitas pengaturan jumlah *output*, dan memiliki jumlah stasiun kerja yang tidak pernah lebih besar daripada lintasan tradisional garis lurus (Miltenburg dan Wijngaard, 1994).



Gambar 1 Contoh lintasan bentuk U

Permasalahan keseimbangan lintasan bentuk U pertama kali dikemukakan oleh oleh Miltenburg dan Wijngaard (1994) yang dalam penelitiannya menggunakan metode optimasi dan heuristik. Penelitian ini menyeimbangkan lintasan bentuk U dengan tujuan untuk meminimasi jumlah stasiun kerja, yang dikenal sebagai permasalahan keseimbangan lintasan tipe I. Kemudian Ajenblit dan Wainwright (1998) melakukan penelitian menggunakan algoritma genetika untuk memecahkan permasalahan yang sama. Erel et al. (2001) selanjutnya membahas penggunaan algoritma *Simulated Annealing*, sementara Martinez dan Duff (2004) mengembangkan metode heuristik yang digabungkan dengan algoritma genetika, juga untuk memecahkan permasalahan keseimbangan lintasan tipe I.

Permasalahan keseimbangan lintasan bentuk U semakin berkembang dengan dilibatkannya penggunaan waktu proses yang bersifat stokastik, sebagai usaha untuk mengakomodir perbedaan performansi operator stasiun kerja. Salah satu kelebihan dari penggunaan waktu proses stokastik dalam keseimbangan lintasan adalah dapat meminimasi probabilitas waktu aktual melebihi waktu siklus di suatu stasiun kerja (Becker & Scholl, 2006).

Guerriero dan Miltenburg (2003) meneliti penggunaan algoritma rekursif dalam keseimbangan lintasan bentuk U dengan waktu proses stokastik. Gamberini et al. (2004) menggunakan algoritma heuristik baru dan Sitorus et al. (2009) mengembangkan algoritma berbasis *Ant Colony System* untuk memecahkan masalah yang sama. Selain itu telah diteliti pula penggunaan algoritma genetika dalam keseimbangan lintasan bentuk U, juga dengan waktu proses stokastik, namun dengan tujuan meminimasi waktu siklus lintasan atau dikenal sebagai keseimbangan lintasan tipe II (Sitorus et al., 2007).

Penelitian ini bertujuan untuk mengembangkan algoritma berbasis *Tabu Search* untuk memecahkan permasalahan keseimbangan lintasan bentuk U tipe I. Untuk mengakomodir perbedaan yang ditimbulkan oleh tenaga kerja manusia, penelitian ini menggunakan waktu proses stokastik. Performansi algoritma yang disusun kemudian dibandingkan dengan performansi algoritma berbasis *Ant Colony System* yang dikembangkan Sitorus et al. (2009), selain dengan algoritma heuristik *Maximum Ranked Positional Weight* (Miltenburg dan Wijngaard, 1994).

Makalah ini disusun dalam alur sebagai berikut. Pemaparan diawali dengan penjelasan tentang konsep *Tabu Search*, dimana algoritma berbasis *Tabu Search* yang dikembangkan dijelaskan di bagian selanjutnya. Kemudian akan diuraikan hasil komputasi dan pembahasan atas performansi algoritma yang dikembangkan, yang kemudian ditutup dengan simpulan dan saran untuk penelitian lebih lanjut.

Tabu Search

Tabu Search merupakan suatu metode pencarian lokal iteratif yang biasa digunakan dalam optimasi kombinatorial. Konsep *Tabu Search* yang digunakan dalam penelitian ini diadopsi dari Glover dan Laguna (1997), yang terdiri atas tiga tahap: pencarian awal (*preliminary search*), intensifikasi dan diversifikasi. Pada tahap pencarian awal, algoritma *Tabu Search* menyerupai metode optimasi yang lain, di mana perbedaan utamanya hanyalah bahwa pada *Tabu Search* sebuah solusi dapat diterima meskipun kualitas dari solusi tersebut tidak lebih baik daripada solusi awal. Pada tahap intensifikasi dilakukan pergerakan atau pencarian solusi di area sekitar solusi yang telah ditemukan pada tahap pencarian awal, sedangkan tahap diversifikasi mencari solusi pada area-area baru (eksplorasi).

Istilah 'tabu' dalam *Tabu Search* berasal dari usaha untuk menghindari terjadinya *cycling* (kembali ke solusi awal), di mana beberapa pergerakan dinyatakan tabu atau tidak boleh dilakukan selama beberapa iterasi. Gerakan-gerakan yang tabu ini disimpan dalam daftar tabu atau *tabu list*. Dengan menggunakan *tabu list*, solusi yang tidak lebih baik dari solusi awal dapat diterima agar dapat menghindari kondisi terjebak dalam optimal lokal. Akan tetapi, dalam *Tabu Search* terdapat pula kondisi aspirasi (*aspiration condition*), di mana jika solusi dari gerakan yang terdapat pada *tabu list* merupakan solusi yang lebih baik dari semua solusi yang telah ditemukan, maka solusi ini akan diterima dan dibebaskan dari *tabu list*.

Pengembangan Algoritma Berbasis *Tabu Search*

Pengembangan algoritma dilakukan sesuai dengan tahap Tabu Search, yaitu tahap pencarian awal, tahap intensifikasi dan tahap diversifikasi.

Tahap Pencarian Awal

Tahap pencarian awal (*preliminary search*) dilakukan dengan langkah-langkah berikut.

Langkah 0 : Inisialisasi

Pertama, masukkan input. Input yang diperlukan adalah kendala presedensi, waktu siklus yang diinginkan dan waktu proses masing-masing operasi. Oleh karena dalam penelitian ini waktu proses yang digunakan bersifat stokastik, maka input waktu proses yang dibutuhkan terdiri dari waktu rata-rata pengerjaan operasi (μ) dan juga variansinya (σ^2). Waktu proses setiap operasi akan dihitung dengan persamaan berikut (Bedworth dan Bailey, 1987):

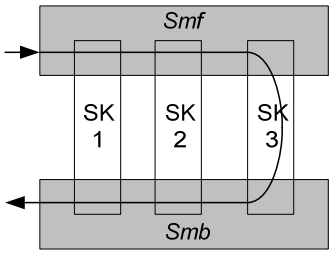
$$t = \left[\sum_{i=1}^{n} \mu_i\right] + (Z_\alpha)\left[\sum_{i=1}^{n} V(t_i)\right]^{1/2} \quad (1)$$

dimana:
n : jumlah operasi pada suatu stasiun
μ_i : rata-rata waktu proses untuk operasi i
$V(t_i)$: variansi dari waktu proses untuk operasi i
Z_α : nilai yang diperoleh dari standar distribusi normal

Kedua, tentukan nilai dari parameter yang digunakan. Berikut adalah parameter-parameter yang harus ditentukan nilainya terlebih dahulu: (1) *Tabu list size* (Ts), yaitu parameter yang mengatur berapa lama dan berapa jumlah *move* dalam kondisi tabu. (2) Iterasi maks (N_{maks}), yaitu parameter yang mengatur banyaknya iterasi maksimum yang dikehendaki. (3) Iterasi maksimum tanpa perbaikan (A_{maks}), yaitu parameter yang membatasi banyaknya iterasi tanpa adanya perbaikan pada solusi. (4) Jumlah *neighborhood* ($Jmlh_{maks}$), jumlah dari *neighborhood* yang diinginkan pun perlu ditentukan terlebih dahulu agar ruang pencarian tidak terlalu luas dan waktu yang dibutuhkan tidak terlalu lama. (5) Iterasi maksimum intensifikasi (ni_{maks}), yaitu parameter yang mengatur banyaknya iterasi selama melakukan proses intensifikasi. (6) Iterasi maksimum diversifikasi (nd_{maks}), yaitu parameter yang mengatur banyaknya iterasi selama melakukan proses diversifikasi.

Ketiga, tentukan solusi awal. Pencarian solusi awal dilakukan dengan menggunakan metode heuristik *Maximum Ranked Positional Weight* (Miltenburg & Wijngaard, 1994). Penelitian ini dikembangkan untuk memecahkan permasalahan keseimbangan lintasan bentuk U *single product*, di mana penugasan operasi bisa dilakukan secara *forward* atau *backward.* Oleh karena itu dalam satu stasiun terdiri dari : (a) Stasiun bagian depan (S_{mf}) yang berisi operasi-operasi yang ditugaskan karena pendahulunya sudah ditugaskan (penugasan secara *forward*). (b) Stasiun bagian belakang (S_{mb}) yang berisi operasi-operasi yang ditugaskan karena pengikutnya sudah ditugaskan (penugasan secara *backward*).

Penempatan stasiun S_{mb} dan juga stasiun S_{mf} dalam lintasan keseimbangan bentuk U diilustrasikan dalam Gambar 2.



Gambar 2 Penempatan stasiun bagian depan
dan belakang pada lintasan bentuk U

Ketiga, set nilai iterasi n = 0, nts_i = 0 dan A = 0. Sebagai permulaan jumlah iterasi akan dimulai dari 0 dan dalam setiap iterasi nilai n akan dinaikkan 1. Proses pencarian akan berhenti saat n sudah mencapai nilai maksimumnya. Selain jumlah iterasi yang dihitung, perlu juga ada suatu *counter* yang menandakan pada iterasi keberapa suatu operasi mulai berstasus tabu (nts_i). Suatu operasi dikategorikan tabu jika operasi tersebut merupakan operasi yang dipertukarkan (*swap*) atau disisipkan (*insert*) dalam pembentukan *candidate move*. Selain itu, perlu dihitung pula jumlah iterasi tanpa adanya perbaikan pada solusi (A). Jika A sudah mencapai batas maksimumnya maka proses pencarian

awal (tahap *preliminary search*) akan dihentikan karena dianggap sudah tidak efektif lagi dan akan masuk ke tahap selanjutnya yaitu intensifikasi dan diversifikasi.

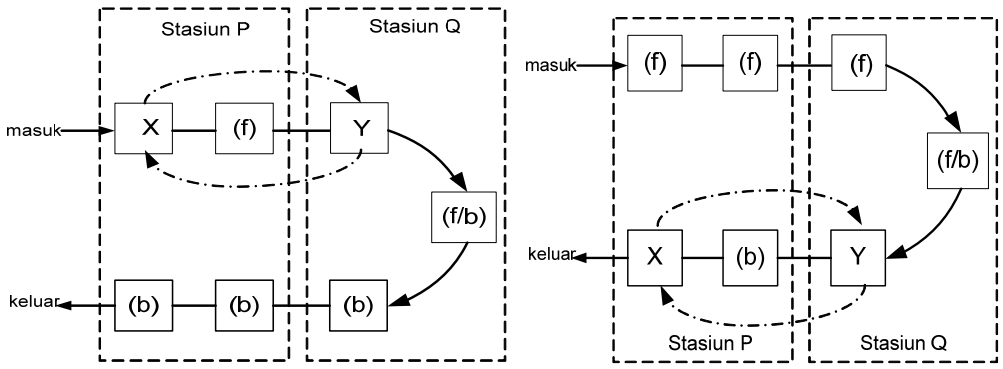
Langkah 1 : Pembentukan *Candidate Move*

Pembentukan *candidate move* dilakukan dengan langkah-langkah berikut. Pertama, bangkitkan *candidate move* dengan metode *neighborhood search*. Proses pencarian *candidate move* ini dilakukan sampai jumlah *candidate move* yang diinginkan telah diperoleh ($jmlh_{maks}$). Dalam proses pencarian *candidate move*, solusi dapat dihasilkan dengan menggunakan proses *swap* maupun *insert*. Proses ini masing-masing memiliki probabilitas yang sama untuk menghasilkan sebuah solusi. Jika penggunaan metode yang dipilih tidak menghasilkan solusi, maka proses dilanjutkan dengan metode lainnya. Sebagai contoh jika proses pencarian dengan menggunakan metode *insert* tidak dapat menghasilkan solusi, maka proses pencarian akan diganti dengan menggunakan metode *swap*, begitupun sebaliknya.

Langkah-langkah pencarian *candidate move* adalah sebagai berikut : (a) Set jmlh = 0. (b) Bangkitkan bilangan random (Rb) antara 0 sampai dengan 1. (c) Periksa apakah bilangan random tersebut ≤ 0.5. Jika ya, pencarian solusi tetangga dilakukan dengan proses *swap.* Jika tidak, pencarian dilakukan dengan proses *insert*. (d) Set jmlh = jmlh + 1. (e) Cek apakah jmlh = $jmlh_{maks}$. Jika ya, pencarian *candidate move* selesai, sedangkan jika tidak, kembali ke langkah b.

Pembentukan *candidate move* dapat dilakukan dengan proses *swap* maupun *insert.* Proses *swap* mempertukarkan dua buah operasi dari dua stasiun yang berbeda sehingga diperoleh sekumpulan *candidate move*. Dalam penelitian ini dikembangkan dua cara *swa*p yang diberi nama *Swap* 1 dan *Swap* 2. Dalam proses *insert*, 1 operasi (X) dari stasiun asal (P) dapat dimasukkan ke dalam maksimum 3 stasiun tujuan (Q).

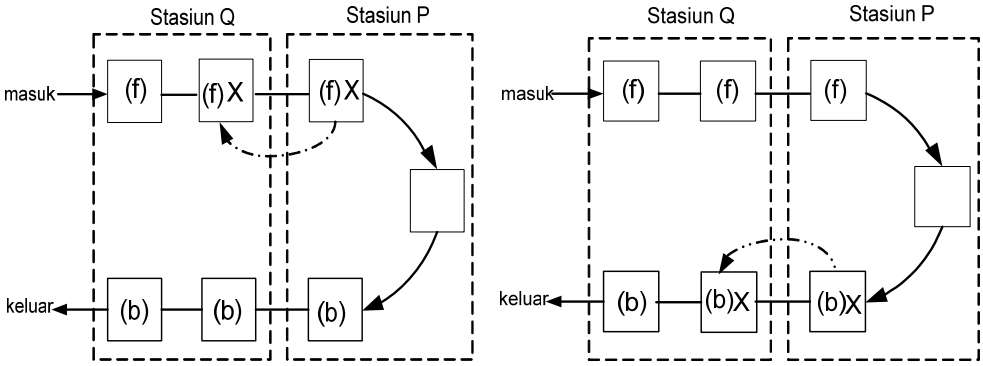
Dalam proses *Swap* 1, satu operasi (X) dari stasiun asal (P) dapat dipertukarkan dengan maksimum tiga operasi (Y) yang bertanda sama dari tiga stasiun tujuan (Q). Jika setelah operasi (X) dipertukarkan sebanyak tiga kali dengan operasi (Y) tetap tidak dihasilkan *candidate move* maka akan dicari operasi baru (X) dari stasiun asal baru (P) yang kemudian akan dipertukarkan kembali dengan maksimum tiga operasi (Y) dari tiga stasiun tujuan (Q). Stasiun asal (P) termasuk operasinya (X) akan diganti sebanyak maksimum tiga kali dan jika setelah tiga kali penggantian masih tidak diperoleh *candidate move* maka proses akan dilanjutkan dengan menggunakan metode *insert*. Gambar di bawah ini menunjukkan contoh proses *swap* berdasarkan cara *Swap* 1.



Gambar 3 Contoh proses pertukaran operasi dengan cara *Swap* 1

Dalam proses *Swap* 2, 1 operasi (X) dari stasiun asal (P) dapat dipertukarkan dengan maksimum tiga operasi (Y) dari tiga stasiun tujuan (Q). Pertukaran antara operasi X dan juga operasi Y tidak akan memperhatikan tanda, jadi pertukaran dapat dilakukan antara operasi yang bertanda sama maupun operasi yang bertanda berbeda. Jika setelah operasi (X) dipertukarkan sebanyak tiga kali dengan operasi (Y) tetap tidak dihasilkan *candidate move* maka akan dicari operasi baru (X) dari stasiun asal baru (P) yang kemudian akan dipertukarkan kembali dengan maksimum tiga operasi (Y) dari tiga stasiun tujuan (Q). Stasiun asal (P) termasuk operasinya (X) akan diganti sebanyak maksimum tiga kali dan jika setelah tiga kali penggantian masih tidak diperoleh *candidate move* maka proses akan dilanjutkan dengan menggunakan metode *insert.*

Dalam proses *insert*, satu operasi (X) dari stasiun asal (P) dapat dimasukkan ke dalam maksimum tiga stasiun tujuan (Q). Jika operasi (X) telah dimasukkan sebanyak tiga kali ke dalam stasiun tujuan (Q) tetap tidak dihasilkan *candidate move* maka akan dicari operasi baru (X) dari stasiun asal baru (P) yang kemudian akan dimasukkan ke dalam maksimum tiga stasiun tujuan (Q). Stasiun asal (P) termasuk operasinya (X) akan diganti sebanyak maksimum tiga kali dan jika setelah tiga kali penggantian masih tidak diperoleh *candidate move* maka proses akan dilanjutkan dengan menggunakan metode *swap*. Gambar 4 di bawah ini menunjukkan contoh proses *insert*.



Gambar 4 Contoh proses *insert*

Kedua, evaluasi setiap *candidate move*. Penelitian ini difokuskan pada permasalahan lintasan keseimbangan bentuk U tipe 1, dimana permasalahan lintasan keseimbangan tipe 1 bertujuan untuk meminimasi jumlah stasiun kerja. Maka nilai objektif pertama yang akan dihitung dari setiap *candidate move* adalah jumlah stasiun kerja yang digunakan. Selain itu dihitung pula fungsi objektif kedua yang akan mempermudah pemilihan solusi terbaik jika solusi mempunyai jumlah stasiun yang sama. Fungsi objektif kedua yang digunakan adalah efisiensi lintasan. Dalam menentukan *candidate move* terbaik dari sekumpulan *candidate move* yang ada tidak perlu dilakukan perbandingan terhadap solusi awal. Hal ini disebabkan dengan menggunakan metode *Tabu Search* solusi yang terpilih dapat merupakan solusi yang lebih buruk dari solusi awal dengan tujuan untuk menghindari pencarian terjebak pada solusi yang lokal optimum.

Langkah-langkah evaluasi *candidate move* adalah sebagai berikut : (a) Set i = 0. (b) Beri indeks i = 1 sampai jmlh maks untuk setiap *candidate move.* (c) Apakah i < i maks? Jika ya, lanjutkan ke langkah d. Jika tidak, simpan semua *candidate move* secara berurutan mulai dari yang terbaik. (d) Set i = i+1. (e) Hitung fungsi objektif pertama (jumlah stasiun kerja) dari *candidate move* ke – i (Si1'). Nilai Si1' menunjukkan nilai fungsi objektif pertama dari candidate move ke-i. (f) Hitung fungsi objektif kedua (efisiensi lintasan) dari *candidate move* ke-i (Si2'). Nilai Si2' menunjukkan nilai fungsi objektif kedua dari *candidate move* ke-i. (g) Kembali ke langkah c.

Langkah 2 : Pemilihan *Move* Terbaik

Pada langkah ini akan dipilih *move* terbaik dari beberapa *candidate move* yang telah diperoleh. Namun pertama-tama perlu dicek terlebih dahulu apakah semua *move* dari setiap *candidate move* yang ada berstatus tabu atau tidak. Jika semua *move* berstatus tabu maka pemilihan *candidate move* terbaik akan didasari pada kriteria aspirasi. Jika tidak semua *move* berstatus tabu maka pilih *candidate move* terbaik yang diperoleh berdasarkan hasil evaluasi *candidate move* pada langkah 1.

Langkah 3 : Kriteria Aspirasi

Kriteria aspirasi merupakan sebuah kriteria yang mengizinkan tetap dilakukannya sebuah *move* yang berstatus tabu. Dalam penelitian ini terdapat dua kriteria aspirasi yang digunakan yaitu : (a) Jika *move* tersebut menghasilkan solusi yang lebih baik dari solusi terbaik yang ada. Solusi terbaik dapat dilihat berdasarkan dua kriteria yaitu, jika jumlah stasiun lebih kecil dari jumlah stasiun awal dan jika jumlah stasiun sama dengan jumlah stasiun awal, namun nilai efisiensi lintasan lebih besar dari nilai efisiensi awal. (b) Jika semua *candidate move* mengandung *move* yang berstatus tabu maka *candidate move* yang terpilih adalah *candidate move* yang mengandung *move* yang paling lama berada dalam *tabu list*.

Langkah 4 : Evaluasi *Candidate Move* Terpilih

Langkah evaluasi terhadap *candidate move* yang terpilih adalah sebagai berikut: (a) Set *candidate move* (S') dari langkah 2 sebagai solusi sekarang (S). (b) Cek apakah S lebih baik dari S*. Jika tidak, set A = A + 1 kemudian lanjut ke langkah c. Jika ya, set S* = S dan A = 0 dan lanjut ke langkah d. (c) Cek apakah A = A maks. Jika tidak, lanjut ke langkah d sedangkan jika ya, lanjut ke langkah 5. (d) Set n = n + 1, nts = nts + 1. (e) Update *tabu list*. Operasi yang mengubah S menjadi S' dimasukkan ke dalam *tabu list*. (f) Cek apakah n = n_{maks}. Jika tidak, kembali ke langkah 1. Jika ya, lanjut ke langkah 5.

Langkah 5 : Kriteria Pemberhentian

Kriteria pemberhentian yang digunakan dalam penelitian ini adalah jumlah iterasi maksimum (n_{maks}). Akan tetapi jika sebelum iterasi mencapai nilai n_{maks} telah diperoleh nilai A_{maks}, maka tahapan *preliminary search* akan dihentikan dan akan masuk pada tahap selanjutnya yaitu tahap intensifikasi dan diversifikasi.

Tahap Intensifikasi

Tahap kedua dalam algoritma *Tabu Search* adalah tahap intensifikasi. Pada tahap intensifikasi proses pencarian akan lebih difokuskan pada solusi terbaik yang telah ditemukan. Proses intensifikasi dilakukan secara berulang sampai mencapai nilai iterasi intensifikasi maksimum (ni_{maks}).

Langkah-langkah dalam proses intensifikasi adalah sebagai berikut :

Langkah 0 : inisialisasi: set S = S*, kosongkan *tabu list* dan set ni = 0, nts_i = 0.

Langkah 1 : pembentukan *candidate move.*

Langkah 2 : pilih *move* terbaik.

Langkah 3 : kriteria aspirasi.

Langkah 4 : evaluasi *candidate move* terpilih.

Berikut adalah langkah-langkah dalam melakukan evaluasi terhadap *candidate move* yang terpilih: (a) Set *candidate move* (S') dari langkah 2 sebagai solusi sekarang (S). (b) Cek apakah S lebih baik dari S*. Jika tidak, lanjut ke langkah c sedangkan jika ya, set S* = S . (c) Set ni = ni + 1, nts = nts + 1. (d) Update *tabu list*. (e) Cek apakah ni = ni_{maks} .Jika tidak, kembali ke langkah 1 sedangkan jika ya, lanjut ke langkah 5

Langkah 5 : kriteria pemberhentian.

Kriteria pemberhentian yang dipakai dalam proses intensifikasi ini adalah jika jumlah iterasinya sudah mencapai jumlah maksimum intensifikasi (ni_{maks}).

Tahap Diversifikasi

Tahap ketiga dalam algoritma *Tabu Search* adalah tahap diversifikasi. Pada tahap diversifikasi, proses pencarian akan lebih meyebar untuk membuka daerah-daerah baru yang belum dikunjungi. Proses diversifikasi ini dilakukan terus hingga tercapai nilai iterasi diversifikasi maksimum (nd_{maks}). Langkah-langkah dalam proses diversifikasi adalah sebagai berikut.

Langkah 0 : inisialisasi

Terdiri dari : (a) Set solusi yang dihasilkan dari hasil proses intensifikasi ($S_{intensifikasi}$) sebagai solusi sekarang (S). (b) Kosongkan *tabu list.* (c) Set x *move* tersering berada dalam kondisi tabu. (d) Pilih secara random suatu *move*. (e) *Swap* / *insert move* terpilih pada solusi sekarang (S). (f) Cek waktu siklus. Jika pertukaran operasi *feasible* berdasarkan waktu siklus maka lanjut ke langkah g. Jika tidak *feasible*, kembali ke langkah d. (g) Cek kendala presedensi. Jika pertukaran operasi *feasible* berdasarkan kendala presedensi maka lanjut ke langkah h. Jika tidak *feasible*, maka kembali ke langkah d. (f) Set solusi yang dihasilkan (S_{random}) sebagai solusi sekarang (S). (g) Set nd = 0 dan nts_i = 0.

Langkah 1 : pembentukan *candidate move.*

Langkah 2 : pilih *move* terbaik.

Langkah 3 : kriteria aspirasi.

Langkah 4 : evaluasi *candidate move* terpilih.

Terdiri dari (a) Set *candidate move* (S') dari langkah 2 sebagai solusi sekarang (S). (b) Cek apakah S lebih baik dari S*. Jika tidak, lanjut ke langkah c. Jika ya, set S* = S. (c) Set ni = ni + 1, nts = nts + 1. (d) Update *tabu list*. (e) Cek apakah nd = nd_{maks}. Jika tidak, kembali ke langkah 1. Jika ya, lanjut ke langkah 5.

Langkah 5 : Kriteria pemberhentian.

Kriteria pemberhentian yang dipakai dalam proses diversifikasi ini adalah jika jumlah iterasinya sudah mencapai jumlah maksimum diversifikasi (nd_{maks}).

HASIL DAN PEMBAHASAN

Performansi algoritma berbasis *Tabu Search* yang dikembangkan dievaluasi dengan membandingkannya dengan metode heuristik *Maximum Ranked Positional Weight* yang dikemukakan oleh Miltenburg dan Wijngaard (1994) dan algoritma berbasis ACS yang dikembangkan oleh Sitorus et al. (2009). Metode *Maximum Ranked Positional Weight* merupakan modifikasi dari metode *Ranked Positional Weight* yang dikembangkan oleh Helgeson dan Birnie (Bedworth et al., 1987). Selain melakukan evaluasi performansi, penelitian ini juga menganalisis pengaruh parameter *Tabu Search* terhadap kualitas solusi yang dihasilkan.

Kasus yang Digunakan

Kasus-kasus yang digunakan dalam evaluasi performansi ini diadaptasi dari Sitorus et al. (2009) dan dapat dilihat pada Tabel 1. Pada keenam kasus tersebut terdapat variasi struktur presedensi, waktu siklus dan standar deviasi waktu proses. Dalam kasus-kasus tersebut digunakan dua jenis struktur presedensi: struktur cenderung serial dan cenderung paralel. Yang diistilahkan sebagai struktur presedensi cenderung serial adalah jika dalam diagram presedensi terdapat banyak operasi yang memiliki hubungan serial dengan operasi lainnya. Sebaliknya struktur presedensi diistilahkan sebagai cenderung paralel jika terdapat banyak operasi yang memiliki hubungan paralel dengan operasi lain.

Kasus 3, 4, 5, dan 6 dalam penelitian ini serupa dengan kasus 5, 6, 9 dan 10 dalam penelitian Sitorus et al. (2009). Jumlah iterasi maksimum (N_{maks}) yang digunakan adalah sebesar 500, sementara jumlah interasi maksimum intensifikasi (ni_{maks}) dan diversifikasi (nd_{maks}) masing-masing bernilai sama yaitu 50 iterasi.

Tabel 1 Kasus-kasus yang digunakan dalam evaluasi performansi

Kasus	$Jmlh_{maks}$	A_{maks}	TS	Struktur Presedensi	Waktu Siklus	σ
1	75	5	7	Cenderung serial	40 detik	1 σ
2				Cenderung paralel		
3	100	15	15	Cenderung paralel	30 detik	
4					50 detik	
5	125	25	25	Cenderung paralel	40 detik	0.5 σ
6					40 detik	1.5 σ

Perbandingan dengan Metode Maximum RPW

Dari perbandingan antara solusi yang dihasilkan dari metode *Maximum Ranked Positional Weight (Maximum RPW)* dengan solusi yang dihasilkan algoritma berbasis *Tabu Search* diperoleh bahwa kedua metode menghasilkan jumlah stasiun yang sama di seluruh kasus, namun berbeda dalam hal efisiensi lintasan. Data selengkapnya mengenai perbandingan nilai efisiensi lintasan yang dihasilkan dari kedua metode tersebut dapat dilihat pada Tabel 2.

Berdasarkan data perbandingan yang diperoleh, dapat disimpulkan bahwa solusi yang dihasilkan dengan menggunakan algoritma berbasis *Tabu Search* selalu lebih baik dibandingkan dengan metode *Maximum RPW*. Oleh karena metode *Maximum RPW* juga merupakan metode yang digunakan dalam pembentukan solusi awal pada algoritma berbasis *Tabu Search* yang dikembangkan, dapat disimpulkan pula bahwa selalu terjadi peningkatan kualitas solusi dari solusi awal yang digunakan.

Tabel 2 Perbandingan solusi Max RPW dengan algoritma berbasis *Tabu Search*

Kasus	*Max RPW*		*Tabu Search*		Selisih Rata-Rata (%)
	Rata-rata	Stdev	Rata-rata	Stdev	
Kasus 1	92.31	0.00	96.05	0.00	4.05
Kasus 2	90.87	0.00	98.35	0.35	8.23
Kasus 3	80.67	0.00	94.61	0.20	17.28
Kasus 4	90.58	0.00	99.54	0.29	9.89
Kasus 5	86.49	0.00	98.16	0.45	13.49
Kasus 6	92.95	0.00	98.90	0.09	6.40

Beberapa alasan yang menyebabkan solusi yang dihasilkan oleh algoritma berbasis *Tabu Search* selalu lebih baik daripada solusi metode *Maximum RPW* adalah karena *Tabu Search* merupakan metode heuristik multi-arah, dimana proses pencarian akan lebih menyebar dibandingkan proses pencarian yang dilakukan oleh metode *Maximum RPW* yang merupakan metode heuritik biasa. Pencarian solusi ke segala arah pada *Tabu Search* akan memperbesar peluang ditemukannya solusi yang relatif lebih baik.

Pada metode *Maximum RPW*, proses pencarian solusi hanya dilakukan sebanyak satu kali. Dengan menggunakan *Tabu Search* dimana terdapat tahap intensifikasi dan diversifikasi maka kemungkinan untuk memperoleh solusi yang optimal global semakin besar. Selain itu, dalam proses pembangkitan *candidate move* pemilihan operasi maupun stasiun kerja dilakukan secara random sehingga solusi yang dihasilkan sangat bervariasi dan kemungkinan untuk menghindari solusi yang lokal optimum semakin besar.

Perbandingan dengan Algoritma Berbasis ACS

Berdasarkan perbandingan antara solusi yang dihasilkan dari algoritma berbasis ACS dengan algoritma berbasis *Tabu Search*, diperoleh pula kondisi dimana kedua metode menghasilkan jumlah stasiun yang sama di seluruh kasus, namun berbeda dalam hal efisiensi lintasan. Data selengkapnya mengenai perbandingan nilai efisiensi lintasan yang dihasilkan dari kedua metode dapat dilihat pada Tabel 3.

Tabel 3 Perbandingan solusi algoritma berbasis ACS dengan *Tabu Search*

Kasus	ACS		*Tabu Search*		Hasil uji statistik
	Rata-rata	Stdev	Rata-rata	Stdev	
Kasus 1	96.05	0.00	96.05	0.00	Kedua metode sama
Kasus 2	98.75	0.42	98.35	0.35	ACS lebih baik
Kasus 3	94.54	0.42	94.61	0.20	*Tabu Search* lebih baik
Kasus 4	99.71	0.18	99.54	0.29	ACS lebih baik
Kasus 5	98.14	0.69	98.16	0.45	Kedua metode sama
Kasus 6	98.81	0.20	98.90	0.09	*Tabu Search* lebih baik

Hasil yang bervariasi dari penggunaan ACS dan *Tabu Search* menyebabkan sulit untuk menentukan secara langsung dalam kasus mana ACS lebih baik dari *Tabu Search* dan begitu pula sebaliknya. Untuk dapat menentukan hal itu perlu pembuktian terlebih dahulu bahwa hasil dari penerapan ACS dan hasil dari penerapan *Tabu Search* berasal dari dua populasi yang berbeda, untuk itu pada setiap kasus dilakukan uji t terhadap hasil dari algoritma berbasis ACS dan algoritma berbasis *Tabu Search*.

Dari Tabel 3 dapat terlihat bahwa untuk kasus 1, solusi yang dihasilkan oleh algoritma berbasis ACS dan *Tabu Search* tidak memiliki variasi dan menghasilkan nilai efisiensi lintasan yang sama sebesar 96.05%. Hal ini menandakan dalam menyelesaikan kasus yang dapat dikatakan masih bersifat sederhana baik algoritma berbasis ACS maupun *Tabu Search* menghasilkan solusi dengan kualitas yang sama.

Pada Kasus 2 dan 4, solusi yang dihasilkan oleh algoritma berbasis ACS memiliki nilai rata-rata efisiensi lintasan yang lebih baik dari solusi yang dihasilkan oleh algoritma berbasis *Tabu Search*. Berdasarkan hasil uji t yang dilakukan ternyata solusi dari algoritma berbasis ACS dan *Tabu Search* berasal dari dua populasi yang berbeda dan dapat disimpulkan bahwa untuk kedua kasus tersebut, algoritma berbasis ACS memberikan solusi yang lebih baik dibanding hasil dari algoritma berbasis *Tabu Search*.

Hasil uji t pada data solusi Kasus 3 dan 6 menunjukkan bahwa solusi dari kedua metode berasal dari populasi yang berbeda. Dengan demikian dapat disimpulkan bahwa untuk Kasus 3 dan 6, algoritma berbasis *Tabu Search* memberikan solusi yang lebih baik dibandingkan dengan algoritma berbasis ACS. Akan tetapi untuk Kasus 5, hasil uji t yang dilakukan menunjukkan bahwa solusi dari algoritma berbasis ACS dan *Tabu Search* berasal dari populasi yang sama atau dengan kata lain algoritma berbasis *Tabu Search* dan algoritma berbasis ACS menghasilkan solusi dengan kualitas yang sama.

Analisis Pengaruh Parameter *Tabu Search* terhadap Kualitas Solusi

Penelitian ini juga menganalisis pengaruh paramater *Tabu Search* terhadap kualitas dari solusi yang dihasilkan. Parameter yang dianalisis adalah maksimum tanpa perbaikan pada solusi (A_{maks}), jumlah *neighborhood* ($jmlh_{maks}$), *tabu list size* (TS). Untuk ketiga parameter tersebut diuji tiga buah nilai yaitu 5, 15 dan 25 untuk parameter A_{maks}, nilai 7,15 dan 25 untuk parameter TS dan nilai 75, 100 dan 125 untuk parameter $jmlh_{maks.}$. Parameter *Tabu Search* lainnya tidak dianalisis karena pengaruhnya terhadap kualitas solusi sudah cukup jelas.

Berdasarkan hasil pengujian ANOVA yang dilakukan, nilai parameter jumlah *neighborhood* ($jmlh_{maks}$), parameter iterasi maksimum tanpa ada perbaikan pada solusi (A_{maks}), parameter *tabu list size* (TS) dan juga kombinasi interaksi dari ketiga parameter tersebut tidak memberikan pengaruh terhadap nilai jumlah stasiun kerja. Kondisi yang berbeda ditemukan pada pengaruh parameter terhadap nilai nilai efisiensi lintasan, dengan pengaruh yang berbeda untuk kasus yang berbeda. Untuk kasus 1, 4 dan 6 dapat disimpulkan bahwa berdasarkan hasil uji ANOVA yang dilakukan, tidak ditemukan parameter yang berpengaruh secara signifikan terhadap kualitas solusi yang dihasilkan.

Pada kasus 2 hanya parameter A_{maks} dan TS saja yang berpengaruh secara statistik terhadap solusi. Penggunaan nilai A_{maks} terbesar sebesar 25 cenderung menghasilkan nilai yang lebih baik dari penggunaan nilai A_{maks} sebesar 5 dan 15. Penggunaan TS terbesar sebesar 25 cenderung menghasilkan nilai yang lebih baik dari penggunaan nilai TS sebesar 7 dan 15. Pada kasus 3 hanya parameter $jmlh_{maks}$ saja yang berpengaruh secara statistik terhadap solusi. Penggunaan nilai $jmlh_{maks}$ terbesar sebesar 125 cenderung menghasilkan nilai yang lebih baik dari penggunaan nilai $jmlh_{maks}$ sebesar 75 dan 100. Sementara pada kasus 5, hanya parameter $jmlh_{maks}$ saja yang berpengaruh secara statistik terhadap solusi. Penggunaan nilai $jmlh_{maks}$ terbesar sebesar 125 cenderung menghasilkan nilai yang lebih baik dari penggunaan nilai $jmlh_{maks}$ sebesar 75 dan 100.

SIMPULAN

Berdasarkan hasil eksperimen pada keenam kasus hipotetik yang digunakan, dapat disimpulkan bahwa algoritma berbasis *Tabu Search* yang dikembangkan dapat digunakan untuk memecahkan permasalahan keseimbangan lintasan bentuk U tipe I dengan waktu operasi stokastik. Analisis perbandingan yang dilakukan dengan metode heuristik *Maximum Ranked Positional Weight* menghasilkan kesimpulan bahwa algoritma berbasis *Tabu Search* yang dikembangkan selalu menghasilkan solusi yang lebih baik pada seluruh kasus. Hal ini juga menunjukkan kemampuan algoritma yang dikembangkan dalam menghasilkan solusi yanng lebih baik dibandingkan solusi awal, oleh karena metode *Maximum Ranked Positional Weight* juga digunakan untuk mencari solusi awal.

Perbandingan antara algoritma yang dikembangkan dengan algoritma berbasis ACS menghasilkan hasil yang berbeda untuk kasus yang berbeda. Berdasarkan uji t yang dilakukan, dapat disimpulkan bahwa pada Kasus 1 dan 5 tidak terdapat perbedaan kualitas solusi di antara kedua metode. Pada Kasus 2 dan 4, hasil uji t menunjukkan kedua data solusi berasal dari populasi yang berbeda dan dapat disimpulkan bahwa kualitas solusi dari algoritma berbasis ACS lebih superior, meski perbedaannya relatif kecil. Sebaliknya pada Kasus 3 dan 6, dengan metode analisis yang sama dapat disimpulkan bahwa kualitas solusi dari algoritma berbasis *Tabu Search* lebih superior, meski perbedaannya relatif kecil.

Analisis pengaruh parameter *Tabu Search* terhadap kualitas solusi yang dihasilkan memberikan kesimpulan yang berbeda untuk kriteria performansi yang berbeda. Berdasarkan hasil pengujian ANOVA yang dilakukan, nilai parameter jumlah *neighborhood* ($jmlh_{maks}$), parameter iterasi maksimum tanpa ada perbaikan pada solusi (A_{maks}), parameter *tabu list size* (TS) dan juga kombinasi interaksi dari ketiga parameter tersebut tidak memberikan pengaruh terhadap nilai jumlah stasiun kerja. Akan tetapi pada kriteria efisiensi lintasan ditemukan bahwa nilai parameter *Tabu Search* yang digunakan dapat mempengaruhi kualitas solusi, dengan pengaruh yang berbeda untuk kasus yang berbeda.

Penelitian ini hanya melibatkan satu jenis produk yang diproses pada satu lintasan. Permasalahan yang melibatkan multi produk dapat menjadi area penilitan yang menarik untuk dikaji. Selain itu penelitian lebih lanjut dapat pula melibatkan waktu transportasi yang tidak dipertimbangkan dalam penelitian ini. Penggunaan metode meta heuristik lainnya juga dapat dijadikan pertimbangan untuk memperkaya kajian keseimbangan lintasan bentuk U.

DAFTAR PUSTAKA

Ajenblit, D., Wainwright, R.L. (1998). Applying Genetic Algorithms to The U-Shaped Assembly Line Balancing Problem. *IEEE*. 9: 96-101.

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

Becker, C., Scholl, A. (2003). A Survey on Problems and Methods in Generalized Assembly Line Balancing. *European Journal of Operational Research.* 3(1): 694-715.

Erel, E., Sabuncuoglu, I., Aksu, B.A. (2001). Balancing of U-Type Assembly Systems Using Simulated Annealing. *International Journal of Production Research*, 39: 3003-3015.

Gamberini, R., Grassi, A., Gamberi, M., Manzini, R., Regattieri, A. (2004). U-Shaped Assembly Lines with Stochastic Tasks Execution Times: Heuristic Procedures for Balancing and Re-balancing Problems. *Proceedings of the Business and Industry Symposium, 2004 Advanced Simulation Technologies Conference*, 137-143.

Guerriero, F., Miltenburg, J. (2003). The Stochastic U-Line Balancing Problem. *Naval Research Logistics*. 50, 31-57.

Glover, F., Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.

Martinez, U, Duff, W.S. (2004). Heuristic Approaches to Solve the U-Shaped Line Balancing Problem Augmented by Genetic Algorithms. *Proceedings of the 2004 Systems and Information Enginneering Design Symposium*m, 287-293.

Miltenburg, G.J., Wijngaard, J. (1994). The U-Line Line Balancing Problem. *Management Science*, *40*, 1378-1388.

Monden, Y. (1993). *Toyota Production System: An Integrated Approach to Just-in-Time*. 2nd ed. Norcross, GA: Industrial Engineering and Management Press.

Sitorus, H. M., Juwono, C.P., Marlina D. (2007). The Application of Genetic Algorithm on U-Shaped Line Balancing with Stochastic Processing Times. *Proceedings of the 1st Asia Pacific Conference on Manufacturing Systems,* 405-411.

Sitorus, H. M., Juwono, C.P., Gani, L. (2009). Penerapan *Ant Colony System* dalam Menyelesaikan Permasalahan Keseimbangan Lintasan Bentuk U dengan Batasan Waktu Siklus dan Waktu Proses Stokastik. *Journal of Industrial Engineering & Management Systems. 2*(1): 23-32.