Perencanaan Rute dan Kecepatan AGV pada Sistem Pergudangan Menggunakan Algoritma *Ant Colony Optimization*

Anugrah K. Pamosoaji[1]
[1)]Program Studi Teknik Industri, Universitas Atma Jaya Yogyakarta
Gedung Bonaventura, Jl. Babarsari 43, Sleman, Yogyakarta 55281
Email: kusumo_pamosoaji@mail.uajy.ac.id

ABSTRAK

Paper ini membahas proses perencanaan rute dan kecepatan satu grup Automated Guided Vehicle (AGV) pada sebuah sistem pergudangan. Diberikan sebuah grup AGV yang bertugas membawa material keluar gudang. Algoritma ini harus memutuskan rute titik-ke-titik dan kecepatan yang diterapkan selama perpindahan sehingga semua AGV dapat mencapai titik target dan tidak saling menabrak. Selain itu juga waktu perjalanan dari AGV yang paling lambat harus diminimalisasi.Setiap AGV memiliki batas kecepatan maksimum. Gudang yang digunakan adalah jenis gudang sederhana yang dimodelkan sebagai matriks 15 titik. Metode yang digunakan menggunakan algoritma Ant Colony Optimization (ACO) yang dimodifikasi. Dalam algoritma ACO yang dimodifikasi ini, dua solusi dapat diselesaikan: penentuan rute dari titik ke titik dan kecepatan yang harus diterapkan AGV pada rute antara dua titik.Untuk pemilihan titik, algoritma ini menggunakan jarak terpendek antara titik terkini dan titik tujuan. Algoritma ACO digunakan ketika menentukan kecepatan perjalanan antar-titik.Dengan metode ini, tabrakan antar kendaraan dapat dihindari. Hasil simulasi menunjukkan bahwa kinerja algoritma yang diusulkan untuk gudang sederhana cukup memuaskan. Hal ini terlihat dari konvergensi waktu tempuh minimum yang dihasilkan dari kendaraan paling lambat dan rute yang dihasilkan memenuhi syarat bebas benturan.
Kata kunci: Perencanaan rute dan trayektori, automated guided vehicle (AGV), rute titik-ke-titik, pencegahan tabrakan, sistem pergudangan, ant colony optimization.

ABSTRACT

This paper discusses a modifiedalgorithm of route planning and speed of a group of Automated Guided Vehicle (AGV) in a warehousing system. Given an AGV group that carries material out of the warehouse. This algorithm has to decide which routes and speed are applied so that all AGVs can reach the target point without any collision to each other. In addition, the slowest travelling time from AGV must be minimized. Each AGV has a maximum speed limit. The warehouse used as a case study is a simple warehouse that is modeled as a 15-point matrix. The method applied is a modified Ant Colony Optimization (ACO) algorithm. In the ACO algorithm, two solutions can be combined: route determination from point to point and speed that must be applied by the AGV on the route between any two points. The ACO algorithm is used to determine the speed of point-to-point travel. By this method, collision among vehicles can be avoided. The simulation results revealthe performance of the proposed algorithm for a simple warehouse is quite satisfying. This conclusion is indicated by the convergence of the minimum travel time generated from faster vehicles and the routes resulting from collision-free requirements.
Keywords:*Path and trajectory planning, automated guided vehicle (AGV), point-to-point routes, collision avoidance, warehouse systems, ant colony optimization.*

Pendahuluan

Latar Belakang Masalah

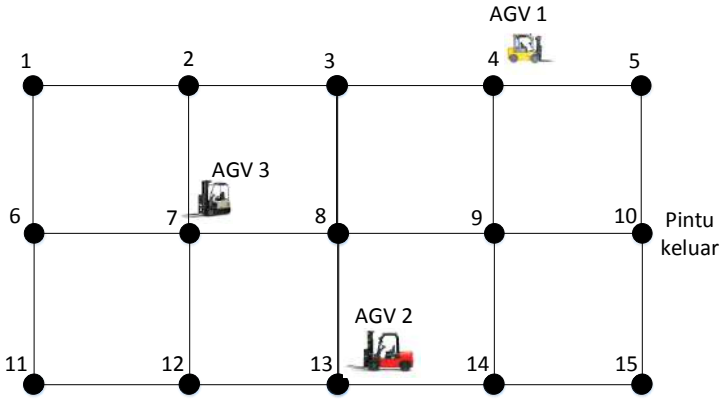
Pembangunan infrastruktur Indonesia yang sangat gencar ditambah dengan semakin membaiknya manajemen logistik menyebabkan arus barang menjadi lebih mudah. Hal ini berimbas pada munculnya kebutuhan sistem pergudangan yang mampu menjadi penyangga (*buffer*) pada sistem rantai pasok. Sebuah gudang terdiri atas
Pada gudang berskala besar, kebanyakan perusahaan menggunakan metode penanganan material (*material handling*) tertentu. Salah satu metode penanganan material tersebut adalah dengan menggunakan *automated-guided vehicle*.Pada studi ini, konfigurasi gudang yang akan dijadikan contoh kasus adalah gudang yang digambarkan pada Gambar ggg1.
Dalam penerapannya, multi-AGV pada sistem pergudangan mencakup beberapa aspek:

1. *Task assignment* (penugasan) , yaitu proses pemberian tugas pada AGV (biasanya dilakukan dari server terpusat). Tugas yang dimaksud adalah permintaan kepada AGV untuk mengambil barang dari rak menuju pintu gerbang atau sebaliknya. Mekanisme ini harus diatur sedemikian rupa sehingga tidak terjadi konflik penugasan pada sistem multi-AGV.
2. Proses *load/unload*, yaitu proses pengambilan barang oleh AGV dari rak menuju pintu gerbang atau sebaliknya. Proses ini lebih bersifat mekanik, sehingga untuk peramalan waktu penyelesaiannya membutuhkan analisis kinematika dan dinamika.
3. *Docking/parking*, yaitu proses kembalinya AGV menuju stasiunnya masing-masing jika sudah tidak mendapat tugas dari server. Selain itu, *docking* yang digunakan untuk AGV harus melakukan *recharging* baterai.

Dalam rangka mengestimasi waktu penyelesaian proses *load/unload*, maka perencanaan pada sistem pergudangan perlu menjadi perhatian. Penelitian ini merupakan penelitian tahap awal dalam rangka mengestimasi waktu penyelesaian proses *load/unload* tersebut. Skenario yang digunakan dalam penelitian ini adalah skenario dari proses *unload* menuju gerbang keluar.

Rumusan Masalah

Pada studi ini dimisalkan terdapat N_t buah AGV pada sebuah gudang dengan *layout* yang digambarkan pada Gambar ggg1. Setiap AGV berangkat dari titik awal masing-masing yang berbeda. Setiap AGV harus melalui sebuah target yang sama (yang dalam studi ini dianggap sebagai titik keluar gudang setelah semua AGV melakukan loading barang dari gudang). Tujuan utama dari studi ini adalah merancang algoritma yang dapat membangkitkan penjadwalan AGV sedemikian sehingga waktu yang ditempuh oleh AGV yang paling lambat dapat diminimalkan. Secara formal, masalah ini dapat diekspresikan secara matematis sebagai berikut (Gambar 1):



Gambar 1. Skenario pick-and-place dengan tiga AGV.

$$\min J = \{\max(T_1, T_2, \ldots T_N)\} \quad (1)$$

yang tunduk pada beberapa ketentuan berikut ini.

1)$0 \leq v_i \leq v_{max}$, untuk semua$t \geq 0$,

2) Untuk semua waktu dengan rentang $\mathbf{0} \leq \boldsymbol{t} \leq \boldsymbol{T_i}$, jumlah maksimum AGV yang dapat menempati sebuah titik adalah 1.

3) Untuk semua waktu dengan rentang $\mathbf{0} \leq \boldsymbol{t} \leq \boldsymbol{T_i}$, jika$\boldsymbol{f}(\boldsymbol{e_{i,j}}, \boldsymbol{t}) > 0$, maka set $\boldsymbol{f}(\boldsymbol{e_{i,j}}, \boldsymbol{t}) = \mathbf{0}$, dan sebaliknya.

dengan $T_i$didefinisikan sebagai waktu tempuh dari AGV ke-*i*.

Dalam paper ini sebuah gudang dengan layout sederhana dijadikan contoh kasus seperti diperlihatkan pada Gambar ggg1. Dimisalkan gudang tersebut mempunyai 15 titik P . Beberapa titik terhubung sehingga membentuk kumpulan jalur segiempat. Diasumsikan bahwa jarak antara titik-titik yang bersebelahan dan terhubung adalah sama. Dalam contoh kasus ini, tiga AGV digunakan untuk melakukan skenario berjalan ke titik keluar (target yang sama).

Tinjauan Pustaka

Menurut Arjman dkk [1], ada empat sub-masalah pada masalah *order picking problem* pada sistem pergudangan, yaitu *sequencing*, *routing*, *batching*, dan *assignment*. *Assignment* adalah masalah pelimpahan tugas pengangkutan barang kepada kendaraan tertentu. *Batching* adalah masalah mengelompokan barang-barang yang akan diambil, sehingga dapat mengurangi jarak tempuh dari kendaraan yang akan mengangkut barang-barang tersebut. *Routing* adalah masalah penyusunan barang-barang yang akan diangkut *Sequencing* adalah masalah mengurutkan *batch* yang diberikan kepada sebuah *picker*/kendaraan pengangkut untuk meminimisasi keterlambatan. Semua sub-masalah tersebut akan bermuara pada masalah pengurangan waktu keterlambatan.

Penelitian ini difokuskan pada sub-masalah *routing,* atau biasa dikenal dengan masalah pencarian rute kendaraan (*vehicle routing problem*/VRP). Berbeda dengan masalah VRP pada umumnya, dalam studi ini tidak hanya masalah jarak tempuh yang akan diminimalisasi, tetapi juga waktu tempuh. Dengan demikian, solusi yang akan dihasilkan dari algoritma yang diajukan dalam paper ini adalah bukan hanya rute perjalanan AGV, tetapi juga pencarian kecepatan setiap AGV yang menghasilkan minimalisasi waktu tempuh kendaraan yang paling lama.

Oleh karena itu, dalam paper ini diajukan sebuah algoritma perencanaan rute dan kecepatan pada sebuah kelompok AGV dalam sistem pergudangan. Motivasi studi ini terutama didasarkan pada pengalaman di lapangan bahwa terdapat beberapa titik yang akan potensial menjadi lokasi tabrakan antar-AGV [6][7]. Dengan demikian, salah satu pendekatan yang dilakukan adalah dengan menerapkan sebuah referensi rute dan kecepatan virtual yang dapat diikuti oleh AGV secara otomatis [8].

Berbagai solusi, terutama dari yang menggunakan pendekatan metaheuristik, telah dilakukan [2][9][12].Menurut Wang et al [10], metode meta-heuristik digunakan untuk menghasilkan solusi suboptimal dengan waktu perhitungan yang dapat diterima. Salah satu pendekatan metaheuristik yang cukup sukses untuk memecahkan masalah-masalah pencarian rute adalah *Ant Colony Optimization* (*ACO*). ACO pertama kali diperkenalkan oleh Dorigo dkk [3] yang bertujuan untuk memecahkan masalah traveling salesman (TSP). Pengembangan pendekatan ini untuk masalah TSP juga semakin banyak[4][5][9]. Lebih jauh, ACO juga diterapkan dalam masalah-masalah pencarian rute di bidang logistik [10][12] dan manufaktur [11].

Metodologi Penelitian

Metode penelitian yang digunakan dalam studi ini dapat dijabarkan ke dalam beberapa langkah berikut ini:

1. Melakukan studi pustaka mengenai masalah penjadwalan AGV pada sistem pergudangan multi-AGV.
2. Mendesain formulasi solusi akhir untuk masalah penjadwalan AGV pada sistem pergudangan multi-AGV.
3. Mendesain sebuah teknik pencarian solusi berbasis Ant Colony Optimization (ACO) yang mengakomodasi formulasi solusi akhir yang telah ditentukan.
4. Melakukan simulasi pengujian untuk kesuksesan 1 AGV melakukan tugas menggunakan perangkat lunak MATLAB.
5. Melakukan simulasi pengujian untuk kesuksesan semua AGV melakukan tugas.
6. Melakukan simulasi pencarian solusi waktu tempuh terlama minimum dengan N iterasi.
7. Pengambilan kesimpulan performa algoritma ACO yang dirancang dengan melihat perkembangan pencarian waktu tempuh terlama minimum selama N iterasi.

Pembahasan

Model Ruang Pencarian (Search Space) dan Model Solusi

Ruang pencarian dapat dimodelkan sebagai berikut. Didefinisikan sebuah graf $\mathbf{G}(\mathbf{P}, \mathbf{E})$ yang terdiri atas sebuah himpunan titik-titik $\mathbf{P} = \{p_i\}$, dengan $i = 1, \dots, N_\mathrm{v}$, N_v adalah jumlah titik-titik pada $\mathbf{P}$, dan sebuah himpunan jalur $\mathbf{E} = \{e_{i,j}\}$, dengan i dan j didefinisikan secara berurutan sebagai label untuk titik awal dan titik akhir; p_i merepresentasikan titik ke-i dan $e_{i,j}$ merepresentasikan jalur penghubung antara titik ke-i dan titik ke-j. Didefinisikan pula sebuah *adjacency matrix* $\mathbf{A} = \{a_{i,j}\}$ dengan $a_{i,j} = 1$ jika titik ke-i dan titik ke-j saling terhubung dan$a_{i,j} = 0$ jika tidak terhubung. Lalu didefinisikan pula sebuah fungsi $f(e_{i,j}) = \{0,1,2, \dots, N_t\}$ yang merepresentasikan jumlah AGV yang menempati jalur penghubung titik ke-i dan titik ke-j dengan arah gerak dari titik ke-i menuju titik ke-j pada satu waktu t. Sebagai catatan, secara umum $f(e_{i,j}) \neq f(e_{j,i})$, yang berarti jumlah kendaraan yang bergerak dari titik ke-i menuju titik ke-j tidak selalu sama dengan jumlah kendaraan yang bergerak dalam arah sebaliknya.

Model ruang pencarian pada studi ini dikembangkan lebih lanjut pada penambahan variabel kecepatan yang ditetapkan untuk setiap jalur $e_{i,j}$, yaitu $v_{i,j}$. Didefinisikan $\mathbf{p}_k^{s,l}$ sebagai komponen solusi posisi titik ke-k yang dilalui oleh AGV ke-l; $\mathbf{P}^{s,l} = \{\mathbf{p}_i^{s,l}, \mathbf{p}_j^{s,l}, \dots \mathbf{p}_{N_\mathrm{v}}^{s,l}\}$ adalah solusi lengkap posisi yang merupakan rangkaian titik-titik persinggahan $\mathbf{p}_k^{s,l}$ yang dilalui oleh AGV ke-l dari titik awal ($\mathbf{p}_1^{s,l}$) sampai ke titik target ($\mathbf{p}_{N_\mathrm{v}}^{s,l}$). Selanjutnya didefinisikan pula $\mathbf{v}_k^{s,l}$ yang merupakan komponnen solusi kecepatan AGV ke-l yang bergerak dari titik ke-k menuju titik ke-(k+1); $\mathbf{V}_s = \{\mathbf{v}_1^{s,l}, \mathbf{v}_2^{s,l}, \dots, \mathbf{v}_{N_i-1}^{s,l}\}$ adalah solusi lengkap kecepatan AGV ke-l dalam pergerakan dari titik awal sampai ke titik target. Akhirnya, solusi keseluruhan dari masalah ini dimodelkan ke dalam himpunan posisi-dan-kecepatan $\mathbf{S} = \{\mathbf{P_s}, \mathbf{V}_\mathrm{s}\}$.

Dalam proses penentuan kecepatan pada jalur $e_{i,j}$, didefinisikan pula *centroid* kecepatan yang disimbolkan dengan $ctr(e_{i,j})$. Sebuah *centroid* kecepatan mewakili satu nilai kecepatan. Nilainya akan berubah setiap kali ada kecepatan yang menghasilkan waktu tempuh AGV terlama yang lebih cepat.

Algoritma Ant Colony Optimization (ACO)

Algoritma Ant Colony Optimization (ACO) pertama kali diperkenalkan oleh Dorigo, dkk [9]. Algoritma ini meniru perilaku koloni semut dalam mencari jalur dari sarang menuju lokasi makanan. Setiap semut akan meninggalkan zat kimia yang disebut *pheromone* setiap kali melangkah. *Pheromone* ini akan menjadi tanda bagi semut lain bahwa sebelumnya pernah ada semut yang melewati jalur yang mengandung *pheromone* tersebut. Semakin banyak kandungan *pheromone* yang ditinggalkan oleh semut-semut sebelumnya, maka semut-semut setelahnya akan cenderung mengikuti arah *pheromone* tersebut.

Algoritma ACO secara garis besar dapat dijabarkan melalui *pseudo-code* berikut ini:

1. Inisialisasi parameter-parameter
2. Selama kondisi terminasi belum tercapai, lakukan langkah-langkah berikut:
3. Mengkonstruksi *Ant Solution*.
4. *Daemon Actions* (opsional)
5. Pembaruan nilai *pheromone.*
6. Selesai.

Sebagai contoh, pada masalah *Traveling Salesman Problem* (TSP) untuk sebuah kendaraan (AGV) di mana setiap titik pada graf hanya boleh dilewati satu kali dan semua titik harus dilewati oleh AGV, konstruksi *Ant Solution* merupakan solusi lengkap yang dibentuk oleh satu semut. Solusi lengkap ini kemudian akan dilihatnilai *fitness*-nya, yaitu nilai fungsi obyektif yang ditetapkan sebelumnya. Beberapa langkah opsional dapat dilakukan (*Daemon Actions*) tergantung pada masalah. Kemudian setelah satu semut selesai mengkonstruksi solusi, maka dilakukan pembaruan nilai *pheromone* dengan persamaan sebagai berikut:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho \sum_{s \in \mathbf{S}} F(s), \quad (2)$$

Dengan $\tau_{i,j}$ merupakan jumlah *pheromone* pada jalur dari titik ke-i menuju titik ke-j;ρ adalah tingkat penguapan (*evaporation rate*); dan $F(s)$ adalah fungsi obyektif untuk komponen solusi s.

Algoritma Ant Colony Optimization (ACO) untuk Pencarian Rute dan Kecepatan

Algoritma ACO biasa digunakan untuk menghasilkan solusi berupa rute. Dalam memecahkan masalah pencarian rute dan kecepatan dalam paper ini, diusulkan sebuah modifikasi algoritma ACO dengan ruang pencarian yang mencakup rute dan kecepatan $\mathbf{S} = \{\mathbf{P_s}, \mathbf{V_s}\}$. Untuk tujuan itu maka dalam paper ini diusulkan sebuah algoritma ACO yang dimodifikasi sebagai berikut:

1. Inisiasi parameter pada algoritma ini mencakup inisialisasi koordinat titik-titik **P**, *adjacency matrix* **A**, jumlah semut N_t, jumlah iterasi.
2. Untuk setiap iterasi, lakukan proses berikut ini:
3. Mengkonstruksi *Ant Solution*:
   Untuk setiap semut ke-l, lakukan proses berikut ini:
   Mencari titik tujuan berikut. Didapat $\mathbf{p}_i^{s,l}$. Lakukan penambahan komponen solusi titik:$\mathbf{p}^{s,l} \leftarrow \mathbf{p}_i^{s,l}$
   Menentukan Kecepatan dari titik terkini sampai ke titik tujuan berikutnya. Didapat $\mathbf{v}_1^{s,l}$. Lakukan penambahan komponen solusi kecepatan: $\mathbf{v}^{s,l} \leftarrow \mathbf{v}_1^{s,l}$.
   Menghitung waktu tempuh dari titik terkini sampai ke titik tujuan berikutnya.
   Selesai
5. Pembaruan nilai *pheromone* pada jalur yang dilalui setiap semut.
6. Selesai.

Algoritma ACO untuk menentukan jalur dan kecepatan AGV dapat dijelaskan sebagai berikut. Sebagai informasi awal, program membutuhkan koordinat titik-titik pada himpunan **P**, dengan *adjacency matrix* **A** yang bersesuaian, jumlah semut N_t yang digunakan untuk pencarian, dan jumlah iterasi. Kemudian ditempatkan sejumlah semut artificial dengan jumlah yang sama dengan jumlah AGV. Pada saat awal, setiap semut artifisial ditempatkan pada posisi awal AGV. Selanjutnya setiap titik tetangga (titik yang terhubung langsung dengan titik yang ditempati setiap AGV) akan dievaluasi untuk ditentukan kelayakannya untuk dipilih menjadi titik persinggahan berikutnya. Kandidat titik persinggahan harus memenuhi syarat perlu (*necessary condition*) sebagai berikut:
1. Kandidat titik persinggahan tidak ditempati oleh semut artifisial yang lain.
2. Jika kandidat titik persinggahan sedang ditempati oleh semut artifisial yang lain, maka jika tujuan berikut dari semut artifisial tersebut adalah titik yang sedang ditempati oleh

semut yang sedang dievaluasi, maka kandidat titik persinggahan tersebut dikeluarkan dari *list*.

Jika syarat perlu tersebut telah dipenuhi, maka pemilihan titik persinggahan dilanjutkan dengan mempertimbangkan sebuah kriteria pemilihan yaitu

$$F_i(\mathbf{p}_n^{s,l}) = \max_{m \in \mathrm{ctr}\,(e_{i,n})}\left(\| \mathbf{p}_n^{s,l} - \mathbf{p}^t \|\right)(\Pr(\mathrm{v}_{i,n}^{c,m})), \quad (3)$$

dengan $\|.\|$ merupakan operator jarak antar dua titik, $\mathrm{v}_{i,n}^{c,m}$ adalah alternatif kecepatan ke-m pada jalur penguhubung titik ke-i dan titik ke-n (salah satu tetangga dari titik ke-i). Operator Pr (.) adalah operator probabilitas yang menunjukkan peluang kecepatan tersebut dipilih. Alternatif dengan nilai F_i terbesar akan dipilih sebagai titik persinggahan berikutnya.

Setiap kali semua semut selesai menjalani rute, maka prosedur pembaruan nilai *pheromone* dilakukan, yaitu dengan menambahkan kandungan *pheromone* pada *centroid* kecepatan yang digunakan oleh setiap semut artifisial. Jadi dengan demikian, setiap centroid menyimpan nilai *pheromone* yang berbeda untuk setiap semut artifisial. Kandungan *pheromone* pada setiap centroid akan digunakan sebagai dasar perhitungan probabilitas. Rumusan untuk probabilitas sebuah *centroid* untuk dipilih adalah sebagai berikut:

$$\Pr\left(\mathrm{v}_{i,j}^{c,m}\right) = \frac{\tau_{i,j}/T_{i,j}}{\sum \tau_{i,j}/T_{i,j}} \quad (4)$$

Hasil Pengujian Algoritma

Simulasi dilakukan dengan melakukan 5 kali percobaan. Setiap percobaan dilakukan dalam 30 iterasi menggunakan 3 semut artificial yang merepresentasikan 3 buah AGV. Kecepatan maksimum yang diizinkan untuk setiap AGV adalah 5 m/s. Jarak dari setiap pasangan titik yang terhubung adalah 20 m. Semua AGV harus mencapai target yaitu titik 10.

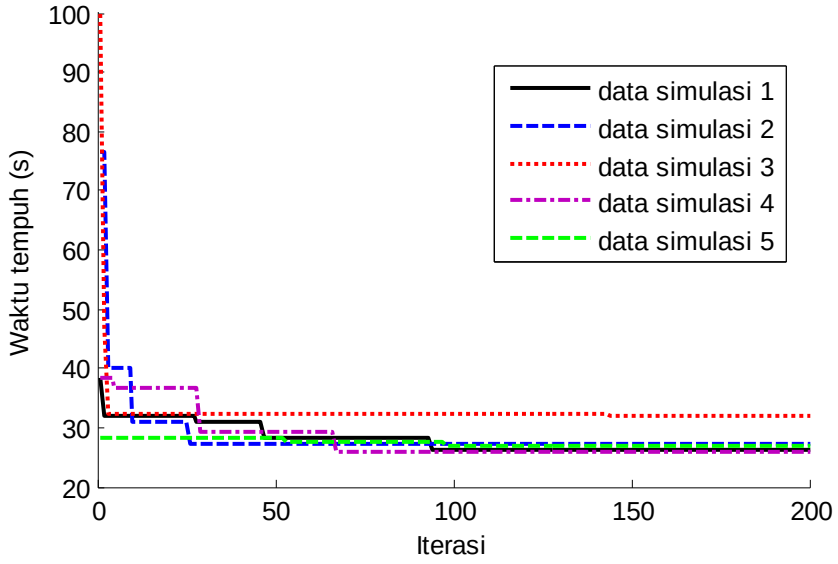
Seperti yang ditunjukkan pada Tabel 1, hasil setiap simulasi menunjukkan bahwa rute yang ditempuh oleh setiap AGV sama, yaitu:
1) AGV 1: titik 4 → 5 → 10,
2) AGV 2: titik 13 →14 → 9 → 10,
3) AGV 3: titik 8 → 9 → 10.

Yang berbeda dalam kelima simulasi tersebut adalah nilai kecepatan yang dihasilkan. Deret kecepatan pada kolom "Kecepatan" dapat dibaca sebagai berikut: angka kecepatan yang pertama adalah kecepatan yang diterapkan dari titik pertama ke titik kedua; angka kecepatan kedua adalah kecepatan yang diterapkan dari titik kedua ke titik ketiga, dan seterusnya. Misalnya, dalam percobaan no. 1, AGV 1 bergerak dari titik 4 ke titik 5 dengan kecepatan 4,8442 m / s; dari titik 5 ke titik 10 dengan kecepatan 4.9713 m/s, dan seterusnya. Secara keseluruhan, semua uji coba memberikan waktu tempuh kendaraan paling lambat minimum rata-rata 26.3741 detik.

Performa yang diuji dari algoritma ACO yang dimodifikasi ini adalah konvergensi. Seperti diperlihatkan pada Gambar 2, secara umum 4 dari 5 percobaan menghasilkan nilai konvergensi yang berdekatan, yaitu di sekitar 26-27 detik. Hanya pada percobaan 3, nilaikonvergensinya berada pada angka yang relative jauh, yaitu 32.0311 detik. Hal ini menunjukkan bahwa secara umum penggunaan ACO untuk masalah ini dapatdipertimbangkan. Munculnya hasil yang berbeda dapat diduga merupakan efek dari adanya area *local maximum* pada ruang pencarian.

Tabel 1. Hasil simulasi penentuan jalur dan kecepatan menggunakan algoritma ACO yang dimodifikasi.

Percobaan nomor ke-	AGV nomor ke-	Rute	Kecepatan (m/detik)	Waktu tempuh AGV terlama/ Waktu pemrosesan (detik)
1	1	4 -- 5 -- 10	4.8442 -- 4.9713	26.3741 (3.4008)
	2	13 -- 14 -- 9 -- 10	4.9873 -- 4.4188 -- 4.3003	
	3	8 -- 9 -- 10	4.0985 -- 4.2982	
2	1	4 5 10	4.3706 -- 4.2594	27.1086 (3.6972)
	2	13 14 9 10	4.6327 -- 4.1199 -- 4.5633	
	3	8 9 10	4.4283 -- 4.6070	
3	1	4 5 10	4.1332 --1.7894	32.0311 (3.6816)
	2	13 14 9 10	4.6897 -- 4.2647 -- 4.1535	
	3	8 9 10	4.9346 -- 4.9346	
4	1	4 5 10	4.8968 -- 3.9813	25.8601 (3.6816)
	2	13 14 9 10	4.6825 -- 4.8233 -- 4.4324	
	3	8 9 10	3.8701 -- 3.6158	
5	1	4 5 10	4.3153 -- 4.9967	26.9392 (3.5100)
	2	13 14 9 10	4.8103 -- 4.9865 -- 3.7729	
	3	8 9 10	4.0312 -- 4.6918	



Gambar 2. Hasil pengujian konvergensi pencarian solusi.

Kesimpulan

Artikel ini mempresentasikan masalah pencarian rute kendaraan (*vehicle routing problem* / VRP) dengan solusi berupa rute dan kecepatan diterapkan pada beberapa kendaraan berpemandu otomatis (AGV) untuk sistem pergudangan. Fitur utama dari metode yang

digunakan adalah modifikasi Ant Colony Optimization (ACO) yang dirancang untuk menghasilkan kecepatan untuk semua rute AGV. Algoritma yang diusulkan telah disimulasikan di gudang sederhana dengan 15 titik. Hasil simulasi dari algoritma yang diusulkan menunjukkan bahwa waktu perjalanan kendaraan paling lambat dapat diminimalkan menjadi nilai konvergen. Juga, metode yang diusulkan terbukti menghasilkan lintasan bebas tabrakan. Pekerjaan mendatang akan difokuskan pada penambahan beberapa aspek yang lebih realistis, seperti jarak kendaraan ke kendaraan, akselerasi, dan desain pengontrol pelacakan untuk setiap kendaraan.

Ucapan Terima Kasih

Penelitian ini di-*support* oleh Laboratorium Pemodelan dan Optimasi, Program Studi Teknik Industri, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta.

Daftar Pustaka

[1] Arjman, E., Shakeri, H., Singh, M., and Bajgiran, O. S. 2018. Minimizing Order Picking Makespan with Multiple Pickers in Wave Picking Warehouse. *International Journal of Production Economics*, 206. pp. 169-183.

[2] Chen, F., Wang, H., Xie, Y., and Qi, C. 2016. An ACO-Based Online Routing Method for Multiple Order Pickers with Congestion Consideration in Warehouse. *Journal of Intelligent Manufacturing*, 27(2). pp. 389-408.

[3] Dorigo, M., Maniezzo, V., and Colorni, A. 1996. Ant System: Optimization by A Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26(1). pp. 29-41.

[4] Dorigo, M. and Gambardella, L. 1997. Ant Colony: A Cooperative Learning Approach to the Travelling Salesman Problem. *IEEE Transactions on Evolutionary Computing*, 1(1). pp. 53-66.

[5] Dorigo, M., and Stutzle, T. 2010. Ant Colony Optimization: Overview and Recent Advances. Dalam Gendreau, M., Potvin, J. Y. (eds), *Handbook of Metaheuristics*. International Series in Operations Research and Management Science, 146, Springer, Boston, MA, USA.

[6] Pamosoaji, A. K. and Hong, K.-S. 2015. Group-Based Particle Swarm Pptimization for Multiple-Vehicles Trajectory Planning. Makalah disajikan dalam *15th International Conference on Control, Automation, and Systems*, Busan, Republic of Korea, 13-16 Oktober. pp.862-867.

[7] Pamosoaji, A. K., Piao, M., and Hong, K.-S. 2019. PSO-Based Minimum-Time Motion Planning for Multiple-Vehicle Systems Considering Acceleration and Velocity Limitations. *International Journal of Control, Automation, and Systems*, DOI: https://doi.org/10.1007/s12555-018-0176-9.

[8] Pamosoaji, A. K. 2018. Trajectory Tracking Control Strategy using Co-Reference for Rear-Steered Vehicle. Makalah disajikan dalam *3rd International Conference on Control and Robotics Engineering*, IEEE, Nagoya, Japan. pp.74-78.

[9] Schyns, M. 2015. An Ant Colony System for Responsive Dynamic Vehicle Routing. *European Journal of Operational Research*, 245. pp. 704-718.

[10] Wang, X., Choi, T.-M., Liu, H., and Yue, X. 2016. Novel Ant Colony Optimization Methods for Simplifying Solution Construction in Vehicle Routing Problems. *IEEE Transactions on Intelligent Transportation Systems*, 17(11). pp. 3132-3141.

[11] Xiang, W. and Lee, H. P. 2008. Ant Colony Intelligence in Multi-Agent Dynamic Manufacturing Scheduling. *Engineering Applications of Artificial Intelligence*, 21. pp. 73-85.

[12] Yuan, Y., and Wang, D. 2009. Path Selection Model and Algorithm for Emergency Logistics Management. *Computers and Industrial Engineering*, 56(3). pp. 1081-1094.