IJCCS, Vol.11, No.1, January 2017, pp. 21~30
ISSN: 1978-1520

Analisis Perbandingan Algoritma Perencanaan Jalur Robot Bergerak Pada Lingkungan Dinamis

Tonny Suhendra*[1], Tri Kuntoro Priyambodo[2]
[1]Universitas Maritim Raja Ali Haji, Kepulauan Riau
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]tonny.suhendra@gmail.com,** [2]mastri.ugm@gmail.com

Abstrak

Dengan semakin berkembangnya teknologi dan semakin kompleks sebuah lingkungan (lingkungan dinamis) penggunaan algoritma dalam perencanaan jalur menjadi hal yang penting untuk dilakukan, masalah yang akan diselesaikan oleh algoritma perencanaan jalur adalah jalur yang dihasilkan merupakan jalur yang aman (bebas tabrakan), kedua adalah jarak tempuh, yaitu, panjang jalur yang dihasilkan dari posisi permulaan robot sampai ke posisi target saat ini dan yang ketiga adalah waktu tempuh, yaitu, waktu yang dibutuhkan robot untuk sampai ketujuan. Penelitian ini menggunakan algoritma ACO dan algoritma A-star untuk mengetahui pengaruh rintangan (lingkungan sederhana) dan juga perbedaan pola pergerakan target (linier dan sinusoidal) terhadap kemampuan algoritma dalam perencanaan jalur untuk menemukan jalur terpendek. Hasil pengujian memeprlihatkan bahwa untuk lingkungan sederhana dimana keadaan target dan rintangan masih statis, diperoleh hasil bahwa algoritma A-star lebih baik dari algoritma ACO baik itu dari waktu temuh dan juga dari jarak tempuh, pengujian tanpa rintangan, dilihat dari jarak tempuh perbedaan selisih yang diperoleh sebesar 0,57% , sedangkan untuk pengujian dengan rintangan selisihnya sebesar 9%. Pada pengujian lingkungan kompleks dimana target dan rintangan bergerak secara dinamis dengan pola tertentu, dari ketiga kondisi lingkungan yang telah diujikan, algoritma ACO lebih baik dari pada algoritma A-star dimana algoritma ACO dapat menemui jalur dengan jarak optimal atau jarak terpendek.

***Kata kunci** – Perencanaan jalur, robot bergerak, algoritma ACO, algoritma A-star*

Abstract

Development of technology and complexity of an environment (dynamic environtment), the use of algorithms in path planning becomes an important thing to do, problem to be solved by the path planning is safe patch (collision-free), second is the distance traveled, ie, the path length is generated from the robot start position to the current target position and the thirdtravel time, ie, the timerequired by the robot to reached its destination.this research uses ACO algorithm and A-star Algorithm to determine the influence of obstacles (simple environment) and also differences in the pattern of the target motion (linier and sinusoidal)on the ability of the algorithm in pathplanning for finding the shortest path. The test results show that for a simple environtment where the state of target and obstacles still static,the resukt that A-star algorithm is betterthan ACO algorithm both in terms of travel time and travel distance. Testing with no obstacles, seen from the distance travelled differences obtained of 0,57%, whereas for testing with obstacles difference of 9%. Testing in a complex environtment where the targets and obstacles which movesdinamically with a certain pattern, from the three environmental conditions that has been tested, ACO algorithm is better than A-star algorithm where the ACO algorithm can find a path with optimal distance or the sortest distance.

***Keywords** – path planning, mobile robot, ACO algorithm, A-star Algorithm*

Received November 2nd,2016; Revised January 30th, 2017; Accepted January 30th, 2017

1. PENDAHULUAN

Robot bergerak (*mobile robot*) merupakan mesin pintar yang dapat bekerja dibawah kondisi tertentu dan tanpa campur tangan manusia. Robot membawa navigasi dan tugas tertentu pada saat bersamaan, seperti robot pelayan, robot penyelamatan dan juga robot yang dapat melakukan eksplorasi [1]. Supaya robot dapat bergerak dengan baik diperlukan navigasi yang baik pada robot, navigasi robot bergerak (*mobile robot*) pada lingkungan dinamis mempunyai dua komponen utama yaitu lokalisasi [2] dan perencanaan jalur [3]. lokalisasi merupakan penentuan posisi dan orientasi robot dengan melihat kondisi lingkungan sekitarnya, sementara perencanaan jalur lebih ke bagaimana robot dapat menghindari tabrakan (*collision free*) dan menemukan jalur terpendek menuju target. Selain dua komponen utama di atas (lokalisasi dan perencanaan jalur), menurut [4] ada 3 (tiga) permasalahan yang juga menjadi perhatian pada navigasi robot, yaitu efisiensi (*efficiency*), aman (*safety*) dan akurasi (*accuracy*).

Menurut [5], bahwa perencanaan jalur merupakan salah satu permasalahan yang penting pada navigasi robot bergerak, dikarenakan untuk mendapatkan jalur yang optimal robot harus mengetahui bahwa pergerakan berikutnya yang diambil akan menghasilkan jalur dengan biaya minimal (*minimal cost*). Banyak hal yang mempengaruhi untuk mendapatkan jalur yang optimal.Sifat lingkungan (*environment*) apakah bersifat *global* atau *local*, sifat rintangan dan target (*obstacle* dan *goal*) apakah bersifat dinamis atau statis bahkan kombinasi statis dan dinamis. Menurut [6], bahwa pada perencanaan jalur robot haruslah bebas dari tabrakan (*collision free path*) dari titik awal ke tujuan dan robot menempuh jalur terpendek.Selain menghindari rintangan, pada perencanaan jalur dilingkungan dinamis dimana target dan rintangan bergerak, yang juga harus diperhatikan adalah akurasi, terutama untuk memprediksi pergerakan target menuju ke tujuan akhir, sehingga robot dapat dengan segera mencapai tujuan (target).

Penelitian yang dilakukan Via mencapai target lebih cepat pada pola pergerakan target linier. Namun pada pola pergerakan sinusoidal, hasil penelitian yang dilakukan oleh [7] lebih lama 0.9 detik dari penelitian Jaradat. Hal ini dikarenakan kemampuan metode yang dikembangkan oleh Via dalam memprediksi posisi target yang bergerak, dimana faktor perhitungan prediksi posisi target belum akurat untuk pola pergerakan *sinusoidal.* Penelitian ini akan mencoba menganalisis kinerja algoritma perencanaan jalur (*path planning algorithm*) dalam memprediksi posisi target yang bergerak dengan pola pergerakan linier dan sinusoidal dimana rintangan dan target yang bergerak sepanjang waktu, sehingga diketahui algoritma mana yang lebih baik, seberapa besar pengaruh rintangan dan target bergerak pada kemampuan (kinerja) algoritma perencanaan jalur. Metode perencanaan jalur yang akan dibandingkan yaitu menggunakan algoritma *Ant Colony Optimization* dan algoritma *A-Star.*

2. METODE PENELITIAN

Penelitian ini menggunakan simulasi sebagai sarana pengujian untuk mengetahui kemampuan dari algoritma yang akan digunakan pada setiap pengjujian. Kemampuan algoritma ketika melakukan perencanaan jalur (*path planning*) akan dianalisisi dan dibandingkan sehingga nantinya dapat diambil kesimpulan. Proses simulasi akan menggunakan *tools* yang bernama Netlogo, merupakan aplikasi bebas berbasis agen yang dikembangkan oleh Uri Wilensky di *Center for Connected Learning and Compute-based Modeling Northwestern University.*

2.1 Perencanaan Jalur Robot Bergerak

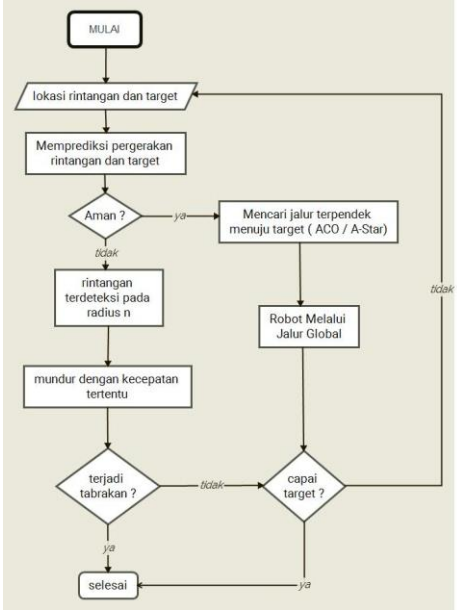
Seperti yang terlihat pada Gambar 1 merupakan perancangan perencanaan jalur robot bergerak dapat dikelompokkan menjadi 4 bagian utama, berikut adalah tahap perancangan bagian-bagian tersebut.

2.1.1. Lokasi dan pergerakan agen (rintangan dan target)

Karena penelitian ini menggunakan *tools* simulasi berbasis agen (Netlogo) sehingga setiap objek (robot, target atau semut (aco)) dapat berdiri sendiri sebagai agen. Ketika agen

robot bergerak, robot akan mendapatkan informasi koordinat posisi target dan rintangan, sehingga robot nantinya dapat menentukan apakah lingkungan disekitarnya aman apa tidak dengan melihat lokasi target dan rintangan berada. Robot, rintangan dan target pada simulasi akan menempati *patch* tertentu, yang memiliki posisi spesifik terhadap koordinat *y* dan koordinat *x* [7].

Selain bersifat statis, rintangan dan target pada penelitian ini juga dapat bergerak atau bersifat dinamis. Untuk rintangan pergerakannya adalah horizontal dan vertikal, sementara itu untuk target selain vertikal dan horizontal, juga dapat bergerak secara sinusoidal. Pergerakan sinusoidal menggunakan rumus gelombang sinus, yaitu $y = A \sin \omega T$, dimana A merupakan amplitudo gelombang, ω adalah kecepatan sudut dan T merupakan periode getaran (waktu).



Gambar 1 Diagram Alir Perencanaan Jalur Robot Bergerak

2.1.2. Prediksi pergerakan target

Pola pergerakan yang akan diprediksi pada penelitian adalah pola pergerakan pada target sementara itu pergerakan rintangan akan menggunakan fungsi radius dalam jangkauan tertentu. Setelah mengetahui posisi agen target, selanjutnya adalah memprediksi pergerakannya. Konsep yang digunakan untuk memprediksi target, menggunakan konsep yang juga digunakan oleh [7] pada penelitiannya yaitu memprediksi untuk satu langkah didepannya, ada dua informasi yang harus didapat dari agen (target), yaitu informasi posisi agen pada saat ini dan posisi agen sebelumnya atau satu langkah sebelumnya. Persamaan yang digunakan untuk memprediksi posisi target ditunjukkan oleh persamaan (1) – (2) :

$$P_{(n+1)x} = 2P_{nx} - P_{(n-1)x} \quad (1)$$

$$P_{(n+1)y} = 2P_{ny} - P_{(n-1)y} \quad (2)$$

$$P_{(n+1)} = [\ P_{(n+1)x}\ \ P_{(n+1)y}\ ] \quad (3)$$

$P_{(n+1)x}$ dan $P_{(n+1)y}$ merupakan posisi koordinat *x* dan posisi koordinat *y* pada langkah (n + 1), $P_{(n-1)x}$ dan $P_{(n-1)y}$ merupakan posisi *x* dan *y* pada langkah (n - 1), sementara itu P_{nx} dan P_{ny} merupakan posisi *x* dan *y* pada langka (n). Persamaan (1), (2) dan (3) cocok untuk memprediksi pola pergerakan vertikal, horizontal dan juga pergerakan diagonal.

2.1.3. Menghindari rintangan

Agen (robot, rintangan dan target) memiliki posisi spesifik terhadap lingkungan simulasi, untuk dapat menghindari rintangan maka pada simulasi ini robot harus mengetahui bahwa posisi saat ini berada, apakah dalam kondisi aman atau kondisi sebaliknya. Akan dibuat suatu kondisi

dengan menggunakan jarak (radius) yang nantinya akan menentukan batas aman dan batas tidak aman, ketika robot akan memasuki batas tidak aman maka robot akan berusaha untuk menghindari rintangan. Akan digunakan fungsi *in-radius* untuk mengetahui keadaan lingkungan disekitar robot.

2.1.4. Mencari jalur terpendek menggunakan algoritma pencarian jalur

Perencanaan jalur merupakan hal yang mendasar dari navigasi robot bergerak, dimana keakuratan jalur bergantung pada pemetaan lingkungan dan lokalisasi [8]. Pada perencanaan jalur, robot harus mencapai target tanpa terjadi tabrakan terhadap rintangan dan jalur merupakan jalur yang terpendek. Beberapa algoritma telah digunakan pada perencanaan jalur, pada penelitian ini akan menggunakan dua algoritma perencanaan jalur yang memiliki karakteristik yang sama yaitu algoritma yang berbasis heuristik atau informasi heuristik. Algoritma tersebut adalah algoritma *incremental search A-star* dan algoritma *ant colony optimozation*, algoritma ini digunakan untuk mencari jalur terpendek. Diharapkan dengan membandingkan kedua algoritma berbasis heuristik ini akan mendapatkan informasi menarik mana yang lebih optimal dalam hal pencarian jalur robot bergerak (*mobile robot*).

Penelitian ini akan menggunakan konsep dari algoritma pencarian yang telah dikembangkan oleh peneliti sebelumnya, untuk algoritma *incremental search A-star* akan menggunakan konsep algoritma yang dikembangkan oleh [9,10], yaitu *incremental search adaptive A-star* dan untuk algoritma *ant colony optimization* akan menggunakan konsep algoritma yang dikembangkan oleh [5], yaitu *improve ant colony system*.

Tabel 1 Perkembangan Algoritma ACO (Dorigo and Stutzle)

ACO algorithm	TSP	Referensi
Ant System (AS)	Ya	Dorigo (1992); Dorigo, Maniezzo & Colorni (1991a,b, 1996)
Elitist AS	Ya	Dorigo (1992); Dorigo, Maniezzo & Colorni (1991a,b, 1996)
Ant-Q	Ya	Gambardella & Dorigo (1995); Dorigo & Gambardella (1996)
Ant Colony System	Ya	Dorigo & Gambardella (1997a,b)
Max-Min AS	Ya	Stutzel & hoos (1996, 2000); Stutzel (1999)
Rank-based AS	Ya	Bullnheimer, Hartl, & Strauss (1997, 1999c)
ANTS	Tidak	Maniezzo (1999)
Hyper-cube	Tidak	Blum, Roli & Dorigo (2001) ; Blum & Dorigo (2004)

2.2. Algoritma Pencarian Jalur ACO

Algoritma *ant colony optimization* (ACO) merupakan algoritma yang digunakan untuk mendapatkan jalur optimal yang berdasarkan pada prilaku dari semut dalam mencari makanan. Algoritma ini merupakan keluarga dari *ant colony algorithm* yang diperkenalkan oleh Marco Dorigo pada awal dekade 90-an, yang digunakan untuk melakukan pencarian jalur optimal pada *graph*. Teori pertama yang diusulkan dalam pengembangan dari *ant algorithm* adalah *ant system* yang menjadi dasar dari model-model *ant algorithm* lainnya dan merupakan dasar dari ACO atau bisa juga disebut sebagai ACO *system,* Tabel (1) merupakan perkembangan algoritma ACO.

Terdapat dua langkah utama untuk melakukan perencanaan jalur menggunakan *ACS*, yaitu *tour* building dan *pheromone updating.*

a. Tour building (n_{iD})

Semut memilih next *patch* menggunakan aturan transisi keadaan probabilitas yang disebut dengan *pseudorandom propotional rule*, seperti yang terlihat pada persamaan (4).

$$J = \begin{cases} \arg max_{j \in \mathcal{N}_i^k} \{(\tau_j)^{\alpha} \text{ x } (n_{jD})^{\beta}\}, if\ q \leq q0 \\ j, lainnya \end{cases} \quad (4)$$

q merupakan angka random antara 0 dan 1, *q0* merupakan parameter yang menentukan apakah akan melakukan eksploitasi atau melakukan eksplorasi, n_{jD} merupakan nilai *invers* jarak antara patch *j* dan tujuan *D*, ketika jarak dekat maka nilai n_{jD} akan besar. *Patch j* merupakan *patch* selanjutnya yang akan dipilih oleh semut untuk langkah selanjutnya, τ_j merupakan jumlah *pheromone* dari *patch j* dan bisa berubah jumlahnya sepanjang waktu. $\mathcal{N}_i^k$ merupakan *patch* yang belum dikunjungi oleh semut *k*, sedangkan α dan β merupakan parameter yang mempengaharui derajat *pheromone* dan jarak. Nilai dari α dan β ditentukan berdasarkan *trial* dan *error* dan biasanya nilai β lebih besar dari nilai α.

Eksploitasi, terjadi ketika nilai *q* lebih kecil dari nilai *q0*, semut akan memutuskan untuk memilih *patch* dimana *patch* tersebut berdasarkan ketertarikan (*most atractive*) memiliki jumlah pheromone yang besar dan juga memiliki jarak lebih dekat ke tujuan sebagi *next patch*. Persamaan yang digunakan untuk melakukan eksploitasi adalah persamaan (4)

$$p_{ij}^k = \frac{(\tau_j)^\alpha (n_{jD})^\beta}{\sum_{j \epsilon \mathcal{N}_i^k} (\tau_j)^\alpha (n_{jD})^\beta} \quad (5)$$

Eksplorasi, terjadi ketika nilai *q* lebih besar dari *q0, patch j* dipilih berdasarkan persamaan (5), sama dengan *random-propotional rule* pada AS, persamaan (6) digunakan untuk menghitung jarak dari koordinat saat ini ke koordinat tujuan.

$$n_{jD} = \frac{1}{d_{jD}},\ d_{jD} = \sqrt{(x_j - x_D)^2 + (y_j - y_D)^2}, \quad (6)$$

Dimana p_{ij}^k merupakan probabilitas *patch j* sebagai *patch* selanjutnya (*next patch*) dimana semut *k* berada pada *patch i*, (x_j, y_j) merupakan koordinat dari *patch j* dan (y_D, x_D) koordinat tujuan *D*. *Patch* dengan jumlah pheromone banyak dan dengan jarak terpendek memiliki probabilitas yang tinggi. Namun, pemilihan *patch j* dengan jarak terdekat dan memiliki jumlah pheromone besar akan melalui proses probabilitas, sehingga tidak semata-mata pemilihan *patch j* berdasarkan dari tingginya tingkat probabilitas, sebagai *next patch.* Proses probabilitas disini adalah dengan membangkitkan nilai acak atau angka random antara 0 dan 1, angka ini akan di jadikan acuan sebagai pembanding, jika nilai probabilitas lebih besar atau sama dengan nilai acak, maka semut akan segera menuju ke *patch* tersebut, sehingga tidak selalu *patch* yang memiliki probabilitas tertinggi yang akan dipilih. Jika pada proses pembandingan tidak ditemukan nilai probabilitas lebih besar dari nilai acak, maka semut akan memilih *patch* dengan nilai probabilitas tertinggi. Jalur terpendek sebagai jalur global terbaik akan dibentuk setelah semua semut selesai melakukan tur-nya, persamaan (7) digunakan untuk mencari jarak terpendek untuk setiap iterasinya.

$$if \min \{L^k\} < L^+, L^+ = L^k \quad (7)$$

L^k merupakan jarak yang dihasilkan oleh semut selama tur, *min* L^k merupakan jarak terpendek pada iterasi saat ini.

b. Pheromone updating

Pembaharuan *pheromone* dapat dilakukan secara global dan lokal. Pembaharuan global dilakukan ketika semua semut telah mencapai tujuan, sedangkan pembaharuan lokal dilakukan ketika semut masih dalam proses pencarian jalur.

1) Penguapan (*evaporation*), penguapan feromon pada semua *patch* dengan menggunakan persamaan (8), fungsi penguapan.

$$\tau_j = (1 - \rho)\, \tau_j, \rho \epsilon (0,1) \quad (8)$$

dimana ρ adalah derajat penguapan dan τ_j jumlah pheromone pada *patch j*. Proses penguapan akan menghindari terjadinya permasalahn lokal optimal ketika proses pencarian jalur berlangsung. *Evaporation* termasuk dalam pembaharuan global, karena dilakukan ketika semua semut telah mencapai tujuan pada setiap iterasinya.

2) Pemberian feromon (*deposit*), proses *deposit* pada ACS terdiri dari pembaharuan global dan pembaharuan lokal. Pada pembaharuan global, pemberian feromon hanya pada jalur global terbaik utuk setiap iterasinya. Persamaan (9) merupakan fungsi pembaharuan, yaitu kombinasi dari penguapan dan deposit.

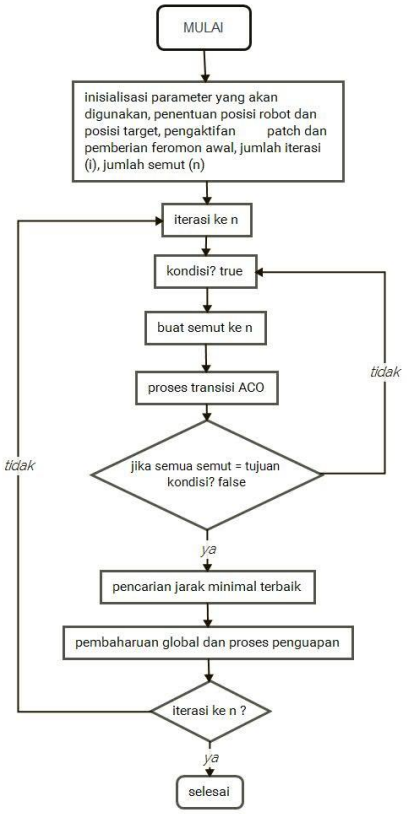
$$\tau_j = (1 - \rho) \tau_j + \rho \Delta \tau_j^{best} \quad (9)$$

$$\Delta \tau_j^{best} = \frac{Q}{L^{best}} \quad (10)$$

Q persamaan (10), merupakan parameter yang berhubungan dengan kecepatan konvergensi, Q biasanya bernilai 1, 100 dan 300. Pembaharuan lokal, dapat dilihat pada persamaan (10), L^{best} merupakan panjang jarak terbaik selama semut melakukan turnya [11]

$$\tau_j = (1 - \rho) \tau_j + \rho \tau_0 \quad (11)$$

τ_0 merupakan feromon awal, feromon akan di perbaharui ketika semut bergerak ke *next patch*. Untuk meningkatkan kemampuan algoritma, hal menarik yang dilakukan oleh Hsu, yaitu melakukan sedikit perubahan pada proses pemberian feromon (*deposit*). Pada pembaharuan global pada persamaan (11), Hsu pada penelitiannya tidak hanya melakukan *deposit* pada jalur terbaik tapi juga memberikan feromon disekitar jalur terbaik. Pemberian feromon dilakukan pada 4 arah disekitar *patch*, yaitu (atas, bawah, kiri dan kanan), menurut beliau pemberian feromon tidak dilakukan untuk 8 arah sekitar, karena bisa menyebabkan semut melakukan eksplorasi secara berlebihan dan mengakibatkan terjadinya lokal optimal. Gambar 2 merupakan diagram alir proses penarian jalur terpendek dengan menggunakan algoritma ACO.



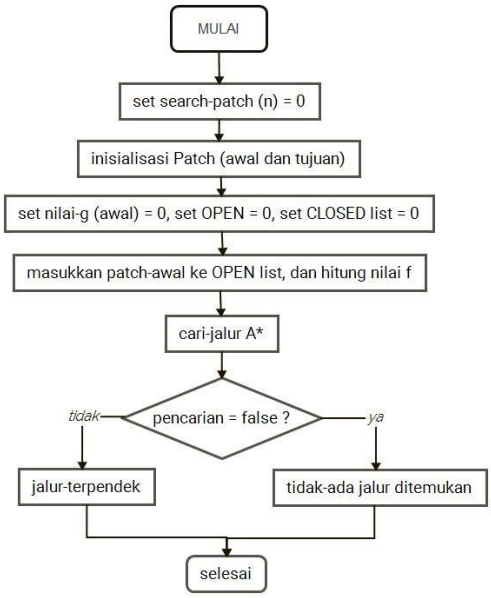
Gambar 2 Diagram Alir Pencarian Jalur Terpendek ACO

2.3. Algoritma Incremental Search A-star

Algoritma *incremental search* merupakan algoritma yang menggunakan kembali informasi dari pencarian sebelumnya untuk meningkatkan pencarian saat ini dan dengan demikian dapat menemukan jalur terpendek untuk serangkaian pencarian yang sama lebih cepat daripada dengan memecahkan setiap masalah pencarian secara mandiri dari awal (Koenig et al.), algoritma *Incremental search* dikembangkan berdasarkan pada metode pencarian sebelumnya dimana proses perencanaan kembali (*replan*) dimulai dari awal lagi atau dari permulaan, hal ini tidak efisien untuk ruang lingkup dengan frekuensi perubahan yang terus terjadi.

Algoritma *incremental search* terbagi menjadi 2 kelas utama berdasarkan informasi heuristik :

1. Kelas pertama, memperbaharui nilai heuristik (*h-values*) pada pencarian saat ini dengan menggunakan informasi (*h*-values) dari pencarian sebelumnnya sehingga menyebabkan pencarian saat ini lebih informasi dan pencarian selanjutnya akan lebih fokus.
2. Kelas kedua, memperbaharui *previous search tree* (pohon pencarian sebelumnya) dari pencarian sebelumnya pada pencarian saat ini, sehingga pada pencarian saat ini tidak perlu dilakukan dari awal.



Gambar 2 Pencarian Jalur Terpendek Algoritma A-star

Penelitian ini akan menggunakan kelas pertama pada proses pencarian jalur, yang saya gunakan adalah metode *Adaptive A-star* yang dikembangkan oleh [9]. Gambar 3 merupakan diagram alir proses penarian jalur terpendek pada algoritma A-star. Agoritma A* merupakan dasar dari algoritma *incremental search* yang akan digunakan, A* memiliki empat nilai untuk setiap *node* $s \in S$ (Sun) :

1. *Heuristic value* (*h-value*) *h*(*s*) dari *s* memperkirakan *dist**(*s*, s_{goal}), biaya minimal pergerakan dari *s* ke s_{goal} dari setiap pencarian.
2. Biaya minimal pergerakan dari *node* posisi awal ke *s*, *g-value g(s).*
3. Biaya minimal pergerakan dari *node* posisi awal melalui *s* menuju *node* tujuan (target), *f(s) := g(s) + h(s).*
4. *Parent pointer, parent(s) points* ke salah satu dari *node* pendahulunya s' dari *s*, s' merupakan *parent* dari *s* pada pohon pencarian (*search tree*). *Parent pointer* digunakan untuk menghasilkan jalur setelah pencarian selesai.

Robot selalu mengikuti jalur terpendek dari posisi saat ini ke posisi target saat ini, sampai robot mencapai posisi target. *S* merupakan sekumpulan *node unblocked*, $s_{awal} \epsilon S$ merupakan posisi robot saat ini dan posisi awal pencarian, $s_{tujuan} \epsilon S$ merupakan posisi saat ini target dan posisi tujuan dari pencarian. $N(s) \subseteq S$ merupakan bagian dari *node* tetangga dari *node* $s \in S$, $c(s,s') > 0$ merupakan jarak *node* $s \epsilon S$ ke *node* $s' \epsilon N(s)$. A* memiliki dua data struktur :

1. Daftar OPEN, berisi semua *node* yang akan dikunjungi, pada mulanya hanya berisi *node* awal dengan *g-value* kosong dan *parent* NULL. A* secara berulang akan memindahkan *node s* yang memiliki nilai *g-value* dan *h-value* terkecil dari daftar OPEN dan dimasukkan ke dalam daftar CLOSED,
2. Daftar CLOSED, berisi semua *node* yang telah dihapus dari daftar OPEN, pada awalnya kosong.

Adaptive A-star termasuk kedalam kelompok algoritma pencarian bertahap (*incremetal search*) menggunakan proses pembelajaran heuristik, membutuhkan *h-values* yang konsisten menuju posisi target atau tujuan untuk setiap pencarian. Adaptive A-star secara berulang mencari jalur dengan biaya minimal (*cost-minimal*) dari posisi saat ini robot menuju posisi saat ini, nilai *h* akan selalu diperbaharui.

3. HASIL DAN PEMBAHASAN

Proses pengujian dilakukan dengan menggunakan *software* simulator Netlogo 5.3.1 dengan menggunakan 2 tahap pengujian, yaitu pengujian sederhana dan pengujian kompleks. Pada pengujian lingkungan kompleks terdiri dari tiga kondisi lingkungan yang memiliki skenario pengujian yang berbeda.

3.1. Hasil Pengujian Lingkungan sederhana ACO dan A-star

Pada pengujian lingkungan akan di *set* dengan keadaan *max-pxcor* 50 dan *max-pycor 50* dan ukuran *patch* adalah 4, dengan demikian banyaknya jumlah *patch* pada lingkungan adalah 10.201 *patch.* posis robot pada koordinat [0 0] dan posisi target pada koordinat [0 45], pada pengujian lingkungan sederhana terdiri dari dua tipe lingkungan yaitu lingkungan tanpa rintangan dan lingkungan dengan rintangan, posisi rintangan diletakkan di antara posisi robot dan posisi target dengan tujuan menghalangi jalur terpendek, jarak tempuh dan waktu tempuh akan dilihat pada pengujian ini, jumlah semut yang digunakan pada algoritma ACO sebanyak 20 semut dan jumlah iterasi 2.

Tabel 2 Pengujian Lingkungan Sederhana Tanpa Rintangan

Pencarian Tanpa Rintangan		
Algoritma	**Jarak Tempuh**	**Waktu Tempuh**
ACO	1040,88561	14,726
A-star	1035	9,0993

Tabel 2 merupakan hasil pengujian lingkungan sederhana tanpa rintangan, dari hasil pengujian terlihat bahwa jarak tempuh yang dihasilkan oleh algoritma A-star lebih baik dari algoritma ACO, begitu juga dengan waktu tempuh.

Tabel 3 Pengujian Lingkungan Sederhana Dengan Rintangan

Pencarian Dengan Rintangan		
Algoritma	**Jarak Tempuh**	**Waktu Tempuh**
ACO	1152,39084	16,5005
A-star	1052,6443	9,3912

Hal yang sama terjadi pada pengujian dengan rintangan Tabel 3 dimana algoritma A-star lebih baik dari algoritma ACO baik itu dilihat dari jarak tempuh mapunu dari waktu tempuh, hal ini karena sifat dari algoritma ACO ketika semut melakukan pencarian dipengaruhi oleh pembatasan ruang lingkup pencarian sehingga ketika melakukan eksplorasi tidak terlalu keluar dari jalur terbaik pada pencarian sebelumnya sehingga kemungkinan untuk mendapatkan jalur terbaik kecil. Sedangkan pada algoritma A-star ruang lingkup pencarian lebih luas sehingga kemungkinan untuk mendapatkan jalur dengan jarak terpendek lebih besar.

3.2. Hasil Pengujian Lingkungan Kompleks ACO dan A-star

Pengujian lingkungan kompleks terdiri dari tiga skenario pengujian, pada pengujian ini rintangan bersifat statis dan dinamis. Tabel 4 merupakan penempatan posisi robot, target dan rintangan pada ketiga kondisi lingkungan yang digunakan, RD merupakan rintangan dinamis, RS rintangan statis, *heading* merupakan arah pergerakan rintangan. Pada pengujian sebelumnya (lingkungan sederhana) sifat dari target adalah diam (statis), pada pengujian lingkungan kompleks target dan rintangan dapat bergerak. Sifat rintangan ada yang bersifat statis dan dinamis, pada rintangan dinamis pola pergerakan yang digunakan hanya bersifat linier.

Pola pergerakan target pada pengujian ini akan digunakan dua pola gerak, yaitu gerak linier dan sinusoidal, sehingga setiap lingkungan pengujian akan dilakukan dua kali pengujian,

data hasil pengujian akan dianalisa sehingga dapat diambil kesimpulan. Pada pengujian ini data yang diambil hanya data jarak tempuh.

Tabel 4 Pengujian Lingkungan Kompleks

KL	RD	RS	Posisi Robot	Posisi Target
K1	Patch [-26 16, -18 8], heading [90, 90]	Patch [-13 14, -14 14, -15 14, -16 14, -17 14] Patch [-13 13, -14 14, -15 14, -16 14, -17 14]	Patch [0 0]	Patch [-24 24]
K2	Patch [-18 12, -15 5, 12 18], heading [90,90,-90]	Patch [-2 16, -3 16, -14 16, -5 16] Patch [-5 7, -6 7, -7 7, -8 7]	Patch [-15 0]	Patch [3 24]
K3	Patch [0 14, -14 6, 8 20, -20 0], heading [-90, -90, -90, -90]	Patch [-20 17, -19 17, -18 17, -17 17] Patch [-25 11, -24 11, -23 11, -22 11] Patch [-20 3, -19 3, -18 3, -17 3 -16 3]	Patch [-18 -5]	Patch [-23 23]

Tabel 5 Pengujian Lingkungan Kompleks

Pola Gerak	Algoritma (Jarak Tempuh (m))					
	ACO			A-star		
	K1	K2	K3	K1	K2	K3
Linier	1441,36066	2838,58473	1744,47681	1474,0643	2802,30822	1737,53778
Sinusoidal	1609,06568	2331,54793	1733,20699	1644,25867	2367,31672	1769,36198

Tabel 5 merupakan rata-rata jarak tempuh dari 10 kali pengujian, K1, K2 dan K3 merupakan kondisi lingkungan pengujian, dari tabel tersebut dapat terlihat bahwa untuk setiap kondisi lingkungan dilihat dari rata-rata jarak tempuh, nilai rata-rata terbaik bisa saja diperoleh oleh algoritma A-star bisa juga diperoleh oleh algoritma ACO, keadaan ini dipengaharui oleh posisi penempatan dan arah pergerakan target. Pada kondisi lingkungan K1 dimana posisi target bergerak mendekati posisi robot, seolah-olah robot akan memotong jalur target, baik pergerakan linier maupun sinusoidal algoritma ACO lebih baik dari A-star. Pada kondisi lingkungan K2 dimana posisi target bergerak menajuhi posisi robot, seolah-olah robot mengejar target, pada pergerakan linier algoritma A-star lebih baik dari algoritma ACO namun pada pergerakan sinusoidal algoritma ACO lebih baik dari A-star. Pengujian kondisi lingkungan K3 sama seperti pengujian kondisi lingkungan K2, pada pergerakan linier Algoritma A-star lebih baik dari algoritma ACO sedangkan pada pergerakan sinusoidal ACO lebih baik dari A-star.

4. KESIMPULAN

Berdasarkan hasil penelitian yang didapat, serta pembahasan tentang algoritma perencanaan jalur yang telah dibuat dapat diambil kesimpulan, yaitu sebagai berikut :

1. Simulasi pengujian algoritma perencanaan jalur robot bergerak dapat dilakukan dengan menggunakan aplikasi Netlogo.
2. Kedua algoritma baik itu algoritma ACO maupun algoritma A-star dapat menemukan solusi dengan baik.
3. Pada pengujian lingkungan sederhana, algoritma A-star lebih baik dari pada algoritma ACO, pengujian lingkungan tanpa rintangan perbedaan selisih yang diperoleh sebesar 0,57%, pada pengujian lingkungan dengan rintangan selisihnya sebesar 9%.
4. Pada pengujian ini dengan tiga kondisi lingkungan yang berbeda algoritma ACO lebih baik dari pada algoritma A-star untuk mendapatkan jalur dengan jarak optimal baik itu pada kondisi lingkungan dengan gerak linier maupun gerak sinusoidal.

5. Pola gerak target baik itu linier dan sinusoidal dapat mempengaharui algoritma pencarian jalur mendapatkan jalur optimal. Selisih diantara kedua algoritma untuk mendaptkan jalur terpendek (jalur optimal) tidak terlalu jauh, algoritma ACO lebih baik daripada algoritma A-star untuk mendapatkan jalur optimal. Pada kondisi lingkungan K1, selisih pada gerak linier adalah 3,73%, sedangkankan pada gerak sinusoidal sebesar 2,51%. Pada kondisi lingkungan K2, selisih gerak linier sebesar 0,36%, sedangkan pada gerak sinusodial sebesar 2,37%. Pada kondisi lingkungan K3, selisih gerak linier pada kedua algoritma sebesar 3.96% dan pada gerak sinusoidal sebesar 3.19%.
6. Metode yang digunakan untuk menghindari rintangan dinamis belum maksimal, sehingga tabrakan masih bisa terjadi.

5.SARAN

1. Perlu dilakukan pengembangan lebih lanjut untuk mengantisipasi pergerakan rintangan dinamis, sehingga robot dapat terus bergerak selama proses pencarian.
2. Pada algoritma A-star bisa dilakukan pengembangan lebih lanjut sehingga tidak terjebak pada kondisi lokal optimal.
3. Pada algoritma ACO bisa dilakukan pengembangan lebih lanjut sehingga hasil setiap pengujian bisa lebih konsisten.

DAFTAR PUSTAKA

[1] Minguez, Javier, and Montano, L., "Sensor-Based Robot Motion Generation in Unknown, Dynamic and Troublesome Scenarios." *Robotics and Autonomous Systems* 52.4 (2005): 290–311. Web. 13 Feb. 2015.

[2] Filliat, David, and Meyer, J. A., "Map-Based Navigation in Mobile Robots:"*Cognitive Systems Research* 4.4 (2003): 243–282. Web. 11 Apr. 2015.

[3] Meyer, J. A, and Filliat, D., "Map-Based Navigation in Mobile Robots:" *Cognitive Systems Research* 4.4 (2003): 283–317. Web. 11 Apr. 2015.

[4] Han, K. M., "Collision Free Path Planning Algorithms for Robot Navigation Problem ProQuest Dissertations & Theses Global - ProQuest." *ProQuest Dissertations & Theses Global*. N.p., 2007. Web. 11 Apr. 2015.

[5] Hsu, C. C, Hou, R. Y, and Wang, Y. W., "Path Planning for Mobile Robots Based on Improved Ant Colony Optimization." *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013. 2777–2782. Web. 31 May 2015.

[6] Zhang, L., "Global Path Planning for Mobile Robot Based on A∗ Algorithm and Genetic Algorithm." *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012. 1795–1799. Web. 11 May 2015.

[7] Via, Y. V., "OPTIMASI PENCAPAIAN TARGET PADA SIMULASI PERENCANAAN JALUR ROBOT BERGERAK DI LINGKUNGAN DINAMIS." *JUTI: Jurnal Ilmiah Teknologi Informasi* 10.1 (2012): 15. Web. 11 May 2015.

[8] Goyal, J. K, and Nagla, K. S., "A New Approach of Path Planning for MobileRobots." *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2014. 863–867. Web. 11 Apr. 2015.

[9] Xiaoxun, S., "Incremental Search-Based Path Planning for Moving Target Search - ProQuest." N.p., 2013. Web. 5 July 2015.

[10] Xiaoxun, S, Yeoh, W, and Koenig, S., "Efficient Incremental Search for Moving Target Search." *IJCAI International Joint Conference on Artificial Intelligence*. N.p., 2009. 615–620. Web.

[11] Dorigo, M., and Di C.G., "Ant Colony Optimization: A New Meta-Heuristic." *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 2. IEEE, 1999. 1470–1477. Web. 13 Jan. 2016.