e-ISSN : 2722-4368

Modifikasi Algoritma A* untuk Pencarian Jalur Tercepat dengan Multi-Skema Prioritas, Tie Breaker dan Force Straight

Randi Farmana Putra
Universitas Pertamina, Jl. Teuku Nyak Arief, Kota Jakarta Selatan, Indonesia
Email : randi.putra@universitaspertamina.ac.id

Abstrak. *Pada robot yang bergerak 4 arah, waktu tempuhnya adalah akumulasi waktu perpindahan dari satu node ke node lain dan waktu rotasi untuk pergantian arah. Algortima A* dapat digunakan untuk menentukan jarak terpendek menuju suatu titik tetapi tidak bisa mendapatkan jalur dengan waktu tempuh tercepat karena adanya faktor rotasi. Dalam penelitian ini menggunakan modifikasi Algorima A* dengan menambahkan metode multi-skema prioritas, tie breaker dan force straight. Modifikasi Algoritma A* dengan multi-skema prioritas menghasilkan 4 jalur dengan waktu tempuh berbeda kemudian dipilih yang tercepat, selanjutnya tie breaker berperan menentukan node parent berdasarkan nilai aktual(g) dan heuristik(h), kemudian force straight saat backtracking bisa meminimumkan jumlah belokan. Modifikasi algoritma A* tersebut dapat digunakan pada robot yang bergerak 4 arah untuk memperoleh jalur dengan waktu tempuh tercepat.*

Kata Kunci : algortima A, force straight, multi-skema prioritas, tie breaker, waktu tempuh*

Abstract. *In a robot that moves in 4 directions, the travel time is the accumulated time for moving from one node to another and the rotation time for changing directions. The A* algorithm can be used to determine the shortest distance to a point but cannot get the path with the fastest travel time because of the rotation factor. In this study using a modification of the A* Algorithm by adding the multi-priority scheme method, tie breaker and force straight. Modification of the A* Algorithm with multi-priority schemes produces 4 paths with different travel times then the fastest is selected, then the tie breaker plays the role of determining the parent node based on the actual value (g) and heuristics (h), then force straight when backtracking can minimize the number of turns. The modification of the A* algorithm can be used on a robot that moves in 4 directions to get the path with the fastest travel time.*

Keyword : algortima A, force straight, multi-skema prioritas, tie breaker, time periode*

PENDAHULUAN

Pada setiap pergerakan robot baik dalam hal berpindah dari satu *node* ke *node* lain dan untuk melakukan pergantian arah pasti memerlukan waktu yang semuanya dihitung sebagai waktu tempuh. Oleh karena itu diperlukan suatu algoritma yang dapat menentukan jalur terpendek atau jalur paling dekat dari titik awal ke titik akhir dengan jumlah belokan paling sedikit sehingga menghasilkan waktu tempuh tercepat.

Robot yang dikaji adalah robot yang hanya dapat bergerak dalam 4 arah yaitu Up, Down, Right, Left dalam suatu area yang sudah diketahui terdiri dari area yang bisa dilewati dan area yang tidak bisa dilewati. Terdapat beberapa algoritma pencarian jalur diantaranya Dijkstra algorithm (Wang, 2012), A* algorithm (Erke dkk, 2020), Genetic algorithm (Alhayali, 2018) dan diantara algoritma tersebut Algoritma A* memiliki keunggulan yaitu efisiensi pencarian yang tinggi dan banyak digunakan dalam perencanaan jalur robot bergerak (Wang, 2020) (Y. Li, 2021) (R. Zuo, 2021), selain itu juga digunakan pada map dengan hambatan (Z. Yu, 2022).

Algoritma A* memiliki kemampuan yang sangat baik dalam proses pencarian jalur solusi *(way seeker)*. Algoritma ini akan mencari rute terpendek yang diambil dari titik awal ke objek berikutnya (ide bagus wahyu, 2018). Terdapat beberapa fungsi *heuristic* yang dapat digunakan dalam algoritma A* salah satunya adalah *Manhattan distance* yang umumnya digunakan dalam grafik berbasis grid untuk memperkirakan jarak ke *node* target dengan menghitung hanya gerakan vertikal dan horizontal,

berguna jika jalur pencarian hanya dapat dicapai secara vertikal dan horizontal saja (Garret, 2014). Penggunaan Algortima A* dengan *Manhattan distance* sebagai fungsi heuristic akan menghasilkan jalur dengan jarak terpendek tetapi belum tentu bisa mendapatkan jalur dengan waktu tempuh tercepat maka perlu dimodifikasi dengan menambahkan metode *tie breaker* guna optimalisasi pencarian node parent. Penelitian ini menggunakan jalur yang mencakup empat arah, sehingga tetangga yang perlu diperiksa maksimal berjumlah empat, maka penggunaan Manhattan distance diharapkan menghasilkan jalur yang tepat. Metode multi-skema prioritas menghasilkan 4 pilihan jalur yang dapat dipilih walaupun semuanya akan menghasilkan jumlah node path yang sama tetapi memungkinkan menghasilkan jumlah turn yang berbeda dan metode *force straight* guna mendapatkan jalur yang lebih lurus.

Pada penelitian Johan Reimon Betmetan mengenai pemilihan jalur tercepat dalam proses evakuasi, memperoleh hasil bahwa dengan menggunakan algoritma Ant-Colony dapat menentukan jalur tercepat dalam proses evakuasi letusan gunung berapi dimana titik-titik evakuasi dan jalur yang harus dilalui dapat ditentukan secara cepat dan efektif dalam proses evakuasi (Johan, 2016). Kemudian pada penelitian Fernando yang membahas tentang peneriapan Algoritma A* pada aplikasi pencarian fotografi memperoleh hasil berupa aplikasi yang dapat memberikan informasi mengenai lokasi foto yang ada di Bandar Lampung dengan dilengkapi maps sebagai penunjuk jalan yang diberikan Algoritma A* untuk menentukan rute terdekat menuju lokasi foto (Fernando, 2020). Tujuan dari penelitian ini adalah untuk dapat memperoleh jalur terpendek atau tercepat dalam perpindahan atau pergantian arah robot dengan menggunakan multi-skema prioritas, *tie breaker,* dan *force straight.*

METODOLOGI PENELITIAN

Algoritma A*

Algoritma A* adalah algoritma untuk mencari jalur terdekat dari node awal hingga node tujuan dengan tipe informed search dimana telah diketahui kondisi map baik node yang bisa dilewati ataupun tidak. Algoritma *A** menggunakan fungsi *f(n)* untuk pencarian jalur terdekat. Fungsi *f(n)* dituliskan sebagai berikut (Tran dkk, 2017):

$$f(n) = g(n) + h(n) \qquad (1)$$

Keterangan:
f(n) adalah harga yang dibutuhkan untuk berpindah dari *node parent* ke *node child* n
g(n) adalah harga dari *node* awal hingga ke *node child* n
h(n) adalah heuristic value merupakan estimasi harga dari *node child* n ke *node end*

Setiap melakukan evaluasi pada sebuah *node*, maka dihitung nilai f(n) di *node* tetangganya, selanjutnya akan dipilih *node child* dengan f(n) minimum untuk menjadi *node parent* berikutnya. Algoritma ini akan memilih *node* yang cenderung mengarahkan ke jalur terpendek secara keseluruhan dengan menggunakan *heuristic*.

Fungsi *Heuristic Manhattan Distance*

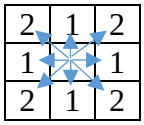
Fungsi *heuristic* digunakan dalam algoritma *A** untuk memperkirakan harga dari suatu node menuju *node* lainnya (Y. Tian, 2021). Terdapat beberapa fungsi heuristic yang dapat digunakan dalam algoritma A* salah satunya adalah Manhattan distance yang umumnya digunakan dalam grafik berbasis grid untuk memperkirakan jarak ke *node* target dengan menghitung hanya gerakan vertikal dan horizontal, berguna jika jalur pencarian hanya dapat dicapai secara vertikal dan horizontal saja (Garret, 2014). Karena dalam penelitian ini jalur hanya mencakup empat arah, sehingga tetangga yang perlu diperiksa maksimal berjumlah empat maka penggunaan *Manhattan distance* diharapkan menghasilkan jalur yang tepat.

e-ISSN : 2722-4368

Perhitungan jarak menggunakan *Manhattan Distance* dilakukan dengan menghitung jarak dari suatu node menuju node lain dengan nilai jarak dari kedua node adalah hasil penjumlahan dari selisih jarak kedua node tersebut pada sumbu X dan Y. Berikut adalah persamaan untuk menghitung *ManhattanDistance* (Chaudhari, 2017):

$$dx = abs\ (currentnode.x - endnode.x) \quad (2)$$
$$dy = abs\ (currentnode.y - endnode.y) \quad (3)$$
$$d = dx + dy \quad (4)$$

Keterangan:
dx adalah jarak sumbu x antara *node* sekarang dengan *node* tujuan
dy adalah jarak sumbu y antara *node* sekarang dengan *node* tujuan
d adalah penjumlahan dx dan dy
ilustrasi nilai d menggunakan Manhattan distance dapat dilihat pada gambar 1 di bawah



Gambar 1. Nilai d pada *Manhattan Distance*

Multi-Skema Prioritas

Algoritma A* bersifat heuristik sehingga bisa saja dihasilkan beberapa jalur dengan jarak yang sama (Tran dkk, 2017). Karena saat perhitungan fungsi f(n) menggunakan Manhattan distance bisa terdapat nilai f(n) yang sama diantara *node child* yang diperiksa, biasanya pemilihan *node parent* berikutnya dilakukan secara random, tetapi pada penelitian ini pemilihannya didasarkan pada tabel multi-skema prioritas dengan prioritas order sesuai arah jarum jam. Pada penelitian ini multi-skema prioritas terdiri dari 4 prioritas seperti tabel 1.

Tabel 1. Multi-skema prioritas

Urutan Prioritas	Prioritas			
	Up	***Right***	***Down***	***Left***
prioritas 1	*up*	*right*	*down*	*left*
prioritas 2	*right*	*down*	*left*	*up*
prioritas 3	*down*	*left*	*up*	*right*
prioritas 4	*left*	*up*	*right*	*down*

Tie Breaker

Tie Breaker terjadi saat semua nilai f *current child* bukan merupakan nilai f(n) minimum, maka *next parent* akan mengambil dari *child node* lain yang memiliki nilai f minimum, yang bisa jadi terdapat lebih dari 1 *node* dengan nilai f minimum (B. Liu, 2019). Pada penelitian ini, pemilihan *next parent* tidak dilakukan dengan random tetapi pemilihan *next parent* dilakukan dengan beberapa parameter yaitu maxg, maxh, ming atau minh. Perhitungan berhenti saat *parent node* merupakan *end node*.

Force Straight* Saat *Backtracking

Waktu tempuh adalah waktu total yang dibutuhkan dalam perjalanan, termasuk waktu berjalan dan waktu berotasi. *Backtracking* berguna untuk menentukan jalur dengan waktu tempuh tercepat yang dilakukan setelah *parent node* sudah mencapai *end node*. *Backtracking* dimulai dari *end node* sebagai *parent node backtracking*, hanya *parent node* yang akan dipilih sebagai jalur *backtracking*. *Node* kedua disesuaikan dengan multi-skema prioritas yang digunakan. Metode *force straight* diperlukan untuk menjadikan jalur lebih lurus (J. Y. Choi, 2019). Force straight dilakukan pada *node* berikutnya

yang dipilih berdasarkan arah dari 2 *node* sebelumnya, jika selisih koordinat *node last parent* dan koordinat *node current parent* sama jumlahnya dengan selisih koordinat *node current parent* dan koordinat *node next parent* berarti 2 *node* sebelumnya searah dengan *node parent next*, maka *node parent next* akan menjadi *node parent current* selanjutnya. Hal ini diperlukan untuk memaksa pemilihan *node next parent* membentuk garis lurus untuk meminimalkan jumlah rotasi pada jalur.

Penentuan Waktu Tempuh Terbaik

Penentuan waktu tempuh robot tercepat diantara semua skema prioritas dilakukan dengan formula sebagai berikut :

Waktu tempuh = Waktu berpindah antar *Node**Jumlah *Node Path* + Waktu untuk bertukar arah*Jumlah belokan

Dengan asumsi, waktu berpindah antar *node* adalah 1 detik dan waktu bertukar arah adalah 1 detik.

HASIL DAN PEMBAHASAN

Pencarian Jalur Tercepat dengan Map yang Lebih Besar

Simulasi dilakukan oleh robot hanya bisa berjalan dalam 4 arah, pada map dengan total 900 *node* dengan jumlah *node open* 611(66,9%) dan *node obstacle* 289(32,1%), pencarian jalur dimulai dari *node* awal (0,29) dan *node* akhir (29,0). Dari tabel 3 dapat dilihat bahwa semua skema prioritas bisa memperoleh jalur terpendek dari titik awal ke titik akhir dengan nilai sama yaitu sejauh 59 *node*, tetapi semua skema prioritas bisa saja menghasilkan *node path* yang berbeda atau sama. Karena pencarian jalur dilakukan dari sudut kiri bawah menuju ke sudut kanan atas, maka penentuan *parent* menggunakan multi-skema prioritas akan mengarah ke *child node right* dan ke *child node up*, sebab jalur ke arah tersebut akan mendekatkan ke *node end*.

Misalkan untuk skema *up*, jika keempat *child node* tetangga memiliki nilai f yang sama akan dipilih *child up* sebagai *parent*, tetapi jika *child up* memiliki nilai f yang lebih besar dari *child* tetangga lain dan *child* tetangga lain memiliki nilai yang sama maka *child right* tetangga akan menjadi *parent*, begitu seterusnya hingga *child left* tetangga yang menjadi *parent*. Tetapi jika tidak ada *child* tetangga yang memiliki f minimum global, maka dipilih *child* dengan f minimum global untuk menjadi *parent*, jika terdapat >1 *child* dengan f minimum diperlukan g dan h sebagai *tie breaker*.

Tabel 3. Waktu tempuh yang diurutkan secara siklis berdasarkan skema prioritas

No	Skema Prioritas	*Tie Breaker*	*Node Parent*	*Node Child*	Persentase Jelajah	Jumlah Belokan	*Node Path*	*Waktu Tempuh*
1	*up*	maxg	136	49	30.28	17	59	76
2	*right*	maxg	136	49	30.28	17	59	76
3	***down***	**maxg**	**162**	**36**	**32.41**	**11**	**59**	**70**
4	*left*	maxg	124	46	27.82	23	59	82
5	*up*	minh	136	49	30.28	17	59	76
6	*right*	minh	136	49	30.28	17	59	76
7	***down***	**minh**	**162**	**36**	**32.41**	**11**	**59**	**70**
8	*left*	minh	124	46	27.82	23	59	82
9	*up*	maxh	324	46	60.56	13	59	72
10	*right*	maxh	324	46	60.56	13	59	72
11	*down*	maxh	327	43	60.56	13	59	72

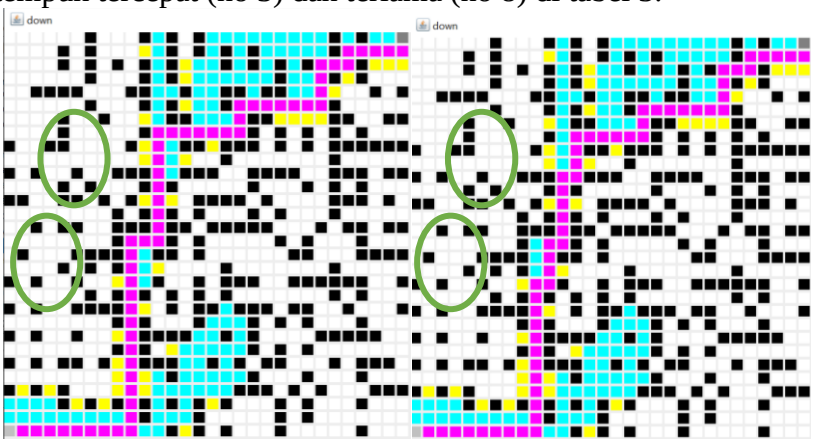
e-ISSN : 2722-4368

No	Skema Prioritas	*Tie Breaker*	*Node Parent*	*Node Child*	Persentase Jelajah	Jumlah Belokan	*Node Path*	*Waktu Tempuh*
12	*left*	maxh	325	42	60.07	14	59	73
13	*up*	ming	324	46	60.56	13	59	72
14	*right*	ming	324	46	60.56	13	59	72
15	*down*	ming	327	43	60.56	13	59	72
16	*left*	ming	325	42	60.07	14	59	73

Dari tabel 3, time travel yang dihasilkan oleh *tie breaker* maxg dan minh bernilai sama pada skema prioritas yang sama dan juga menghasilkan jalur yang sama, begitu juga dengan ming dan maxh. waktu tempuh tercepat dihasilkan oleh skema *down* dengan *tie breaker* maxg (no. 3) dan *tie breaker* minh (no. 7) dengan nilai 70 detik, sedangkan waktu tempuh terlama dihasilkan oleh skema *left* dengan *tie breaker* maxg (no. 4) dan *tie breaker* minh (no. 8) dengan nilai 82 detik. Persentase jelajah terkecil dihasilkan oleh skema *left* dengan *tie breaker* maxg (no. 4) dan *tie breaker* minh (no. 8) sebesar 27,82 %, sedangkan Persentase jelajah terbesar dihasilkan oleh skema *up*, *right* dan *down* dengan *tie breaker* maxh (no. 9,10,11) dan *tie breaker* ming (no. 13,14,15) sebesar 60,56 %. Tabel 3 adalah jalur yang dihasilkan dengan metode *force straight*, sehingga jumlah turn sudah dioptimalisasi.

Perbandingan Penggunaan *Force straight* dan Tanpa *Force straight*

Selanjutnya dibandingkan jalur yang dihasilkan dengan *force straight* dan tanpa *force straight*, diambil data waktu tempuh tercepat (no 3) dan terlama (no 8) di tabel 3.



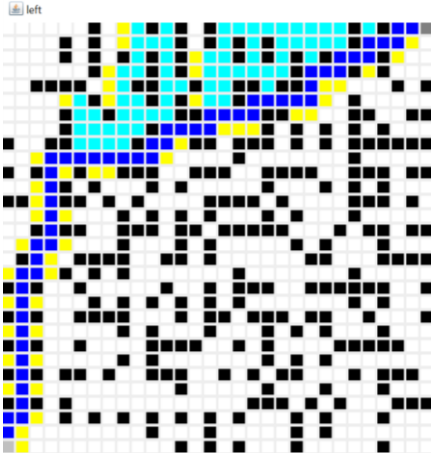
Gambar 5. Jalur dengan waktu tempuh tercepat(magenta) dengan *force straight*(kiri) dan tanpa *force straight*(kanan) dengan optimalisasi jalur pada bulatan merah

Tabel 4. Perbandingan skema prioritas dengan *force straight* dan tanpa *force straight* diurutkan berdasarkan waktu tempuh

Tie Breaker	Skema Prioritas	*Node Parent*	*Node Child*	Persentase Jelajah	Waktu Tempuh	Jumlah Belokan	*Force Straight*
maxg	*down*	162	36	32.41	70	11	*Yes*
maxg	*down*	162	36	32.41	74	15	*no*
minh	*left*	124	46	27.82	82	23	*Yes*
minh	*left*	124	46	27.82	82	23	*no*

Berdasarkan tabel 4, bisa diketahui hasil perbandingan skema prioritas dengan *force straight* dan tanpa *force straight* dengan waktu tempuh tercepat dihasilkan oleh skema *down* dengan *tie breaker* maxg dan *tie breaker* minh dengan nilai 70 detik dengan jumlah turn 11, sedangkan tanpa

menggunakan *force straight*, waktu tempuh yang dihasilkan adalah 74 detik dengan jumlah turn 15, seperti gambar 5 (kiri), optimalisasi bisa dilakukan karena jalur garis magenta masih mempunyai ruang untuk dioptimalisasi menjadi lurus. Waktu tempuh terlama 82 detik yaitu skema *left* dan *tie breaker* maxh, tanpa *force straight* dan menggunakan *force straight* tidak menurunkan waktu tempuh karena jalur biru memang tidak mempunyai ruang untuk dapat di optimalisasi seperti pada gambar 6.



Gambar 6. Jalur dengan waktu tempuh terlama(biru) tanpa dan dengan *force straight*

KESIMPULAN

Setelah dilakukan simulasi pencarian jalur oleh robot yang hanya bisa berotasi ke 4 arah dalam map yang terdiri dari 900 node yang sudah diketahui posisi node rintangan dapat disimpulkan bahwa:

1. Penggunaan Algoritma A* yang dimodifikasi dengan metode multi-skema prioritas, tie breaker dan force straight menghasilkan 4 jalur dimana semua skema prioritas memperoleh jalur terpendek dengan jarak yang sama yaitu 59 node tetapi dengan waktu tempuh berbeda dimana waktu tempuh tercepat diperoleh dari skema down dengan tie breaker maxg dan minh dan menerapkan force straight yaitu selama 70 detik dengan jumlah belokan 11 dan juga bisa ditentukan jumlah node yang telah dievaluasi yang ditunjukkan oleh persentase jelajah yang menjadi faktor penentu jalur terbaik jika waktu tempuh tercepat bernilai sama.
2. *Tie breaker* berperan untuk penentuan *node next parent* jika f *child* bukan minimum global sehingga *next parent* adalah child dengan f minimum, dan jika terdapat >1 child dengan f minimum digunakan parameter g dan h, jalur yang dihasilkan dengan tie breaker maxg akan sama dengan minh, begitu juga dengan jalur dengan *tie breaker maxh* akan sama dengan ming sehingga cukup gunakan maxg dan maxh atau ming dan minh saja.
3. *Force straight* yang dilakukan saat *backtracking* dapat memaksa jalur menjadi lurus yang mengurangi waktu rotasi sehingga juga mengurangi waktu tempuh.

DAFTAR PUSTAKA

[1]. Alhayali S., Ucan O., Bayat O., 2018. Genetic Algorithm for finding shortest paths Problem. ICEMIS '18: Proceedings of the Fourth International Conference on Engineering & MIS. Juni 2018. ISBN 978-1-4503-6392-1. Hal. 1-6. https://doi.org/10.1145/3234698.3234725

[2]. B. Liu and L. Li, "A Novel Tie-breaking Strategy for the A* Algorithm," Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics, pp. 2724-2729, 2019.

[3]. Batmetan, J. R. (2016). Algoritma Ant Colony Optimization (ACO) untuk Pemilihan Jalur Tercepat Evakuasi Bencana Gunung Lokon Sulawesi Utara. AITI, 13(1), 31-48.

[4]. Chaudhari, A. M., Apsangi, M. R., Kudale, A. B. 2017. Improved A-star algorithm with least turn for robotic rescue operations. Communications in Computer and Information Science. Vol. 776:614–627. https://doi.org/10.1007/978-981-10-6430-2_48

[5]. Dalem, I. B. G. W. A. (2018). Penerapan algoritma A*(Star) menggunakan graph untuk menghitung jarak terpendek. Jurnal RESISTOR (Rekayasa Sistem Komputer), 1(1), 41-47.
[6]. Erke, S., Bin, D., Yiming, N., Qi, Z., Liang, X., Dawei, Z. 2020. An improved A-Star based path planning algorithm for autonomous land vehicles. International Journal of Advanced Robotic Systems. Vol. 17(5): 1–13. https://doi.org/10.1177/1729881420962263
[7]. Garret D. F. 2014. Aircraft Route Optimization Using the a-Star Algorithm. Air Force Institute of Technology. Hal 11-13. https://apps.dtic.mil/sti/pdfs/ADA600125.pdf
[8]. J. Y. Choi, H. Kim, and B. C. Jung, "A* Algorithm with Force Straight: Enhancing Pathfinding Efficiency in Grid Maps," Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2947-2952, 2019.
[9]. Mustaqov, M. A., & Megawaty, D. A. (2020). Penerapan Algoritma A-Star Pada Aplikasi Pencarian Lokasi Fotografi Di Bandar Lampung berbasis Android. Jurnal Teknoinfo, 14(1), 27-34.
[10]. R. Zuo, H. Zhang, and S. Li, "Enhancing A* algorithm with multi-criteria priorities for robot path planning," Robotics and Autonomous Systems, vol. 139, pp. 103720, 2021.
[11]. Tran, H. A. M., Ngo, H. Q. T., Nguyen, T. P., Nguyen, H. 2018. Implementation of vision-based autonomous mobile platform to control by A∗ algorithm. 2018 2nd International Conference on Recent Advances in Signal Processing, Telecommunications and Computing (SIGTELCOM). Januari 2018. Hal. 39–44. https://doi.org/10.1109/SIGTELCOM.2018.8325802
[12]. Wang, S. X. 2012. The improved Dijkstra's shortest path algorithm and its application. Procedia Engineering. Vol. 29:1186–1190. https://doi.org/10.1016/j.proeng.2012.01.110
[13]. Wang, X., Liu, X., Wang, Y., & Liang, S. 2020. Research on Path Planning of Mobile Robot Based on Improved A* Algorithm. IE&EM 2019:153–161. https://doi.org/10.1007/978-981-15-4530-6_16
[14]. Y. Bjornsson and A. I. Maier, "Speeding Up Grid Pathfinding with Hierarchical Expansion, Multiple Searches, and Forced Paths," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, no. 1, pp. 526-532, 2014.
[15]. Y. Li, H. Li, and X. Zhang, "A fast A* algorithm for global path planning of mobile robot," Measurement, vol. 181, pp. 25-35, 2021.
[16]. Y. Tian, L. Lu, and W. Zhang, "A Fast and Adaptive A* Algorithm with Multiple Heuristics for Robot Path Planning," Applied Sciences, vol. 11, no. 7, pp. 3038, 2021.
[17]. Z. Yu, L. Cheng, and Y. Zhang, "An Efficient A* Algorithm for Robot Path Planning Based on Obstacle Distance Map," Journal of Intelligent & Robotic Systems, pp. 1-14, 2022.