frontiers | Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 19 August 2022
DOI 10.3389/frobt.2022.958930

OPEN ACCESS

EDITED BY
Allahyar Montazeri,
Lancaster University, United Kingdom

REVIEWED BY
Khoshnam Shojaei,
Islamic Azad University of Najafabad, Iran
Meysam Yadegar,
Qom University of Technology, Iran
Nargess Sadeghzadeh-Nokhodberiz,
Qom University of Technology, Iran

*CORRESPONDENCE
Daehee Lim,
dhlim@vessel21.com

SPECIALTY SECTION
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

RECEIVED 01 June 2022
ACCEPTED 11 July 2022
PUBLISHED 19 August 2022

CITATION
Lim D and Jo J (2022), Path planning with the derivative of heuristic angle based on the GBFS algorithm. *Front. Robot. AI* 9:958930. doi: 10.3389/frobt.2022.958930

Path planning with the derivative of heuristic angle based on the GBFS algorithm

Daehee Lim* and Jungwook Jo

R&D 2 Team, Vessel Aerospace Co., Ltd., Suwon-si, South Korea

Robots used in extreme environments need a high reactivity on their scene. For fast response, they need the ability to find the optimal path in a short time. In order to achieve this goal, this study introduces WA*DH+, an improved version of WA*DH (weighted A* with the derivative of heuristic angle). In some path planning scenes, WA*DH cannot find suboptimal nodes with the small inflation factor called the critical value due to its filtering method. It is hard to develop a new filtering method, so this study inflated the suboptimality of the initial solution instead. Critical values vary in every path planning scene, so increasing the inflation factor for the initial solution will not be the solution to our problem. Thus, WA*DH + uses the GBFS algorithm with the infinitely bounded suboptimal solution for its initial solution. Simulation results demonstrate that WA*DH + can return a better solution faster than WA*DH by finding suboptimal nodes in the given environment.

KEYWORDS

greedy best first search, heuristic angle, heuristic search, node-based algorithm, path planning, trajectory planning

Introduction

Path planning is an essential part of self-moving machines such as self-driving cars, unmanned aerial vehicles, or other robotics systems (De Momi, Elena et al., 2016; Boulares and Barnawi, 2021; Mac, Thi Thoa et al., 2016; Fu, Bing et al., 2018; Duan, Chao, et al., 2020; Liu, Zhiyuan et al., 2020; Hou, Mengxue et al., 2021). Path planning aims to find the path that has the lowest path cost in the shortest time because self-moving machines need fast reactions on their scene. Here, the path cost means the distance from the start to the target. Many path planning algorithms were developed to achieve this goal. Bioinspired algorithms such as genetic algorithm (GA), particle swarm optimization (PSO) (Das et al., 2016; Song et al., 2016), sampling-based algorithms such as rapidly exploring random tree (RRT), Voronoi, and artificial potential field (APF) algorithms are the examples (Yang, Liang et al., 2014; Yang, Liang et al., 2016). However, these algorithms sometimes show poor performances due to some limitations, such as the local minimum situation.

In order to avoid these threats, the A* algorithm (Hart et al., 1968) motivated by Dijkstra's algorithm and other node-based algorithms were developed. The main characteristic of A* and all variations of A* is the heuristic. The heuristic means the

estimated distance from the current node to the target. The heuristic is the most important factor in A* and all variations of A* because the heuristic can change the performance of algorithms (Jing and Yang, 2018). The concept of the heuristic is used not only for node-based algorithms but also for other algorithms, such as the ACO algorithm (Dai, Xiaolin et al., 2019).

The heuristic is a powerful method for finding the optimal path. However, algorithms using the heuristic have a time-consuming problem. This problem made A* hard to use in real-time systems, so many researchers developed various methods to get a result of heuristic-using algorithms in a short time. With these trials, weighted A*(WA*), the bounded suboptimal search algorithm, was developed (Pohl, 1970). WA* uses the heuristic multiplied by the inflation factor ($\epsilon > 1$). The concept of inflating the heuristic solved a time-consuming problem. However, the inflated heuristic cannot guarantee the optimality of the result anymore.

Many researchers focused on this side effect, and as a result, many variations of WA* were developed. For example, dynamically weighted A*(DWA*) uses the *d(root)* as a depth bound (Pohl, 1973) and A^*_ϵ uses the desired suboptimality bound to build the focal list from a node selected to expand (Pearl and Kim, 1982; Thayer and Ruml, 2008; Thayer and Ruml, 2009). Also, Aine Sandip et al. (2016) suggested two versions of multi-heuristic A*(MHA*), which uses multiple heuristic functions to find the path; independent MHA*(IMHA*), which uses independent *g* and *h* values for each search, and shared MHA*(SMHA*), which uses different *g* but a single *g* value for all the searches. Ying et al. (2018) suggested the evolutionary heuristic A*(EHA*), which uses GA to automatically design, calibrate, and optimize multi-weighted heuristic functions to maximize the performance of the algorithm (Yiu et al, 2018).

Anytime algorithms were also used in various path planning environments. Anytime algorithms have a flexible time cost and can return a sub-optimal solution in a short time and progressively optimize it till the time limit expires. The naïve anytime A*(ATA*) returns its result by iteratively reducing the inflation factor; however, it repeats previous works (Zhou and Hansen, 2002). To address this inefficient work, anytime repairing A*(ARA*) reuses the previous work to optimize the path efficiently (Likhachev et al., 2003; Li et al., 2012). The concept of reusing the previous work was adapted to D*. Thus, anytime dynamic A* was developed (Likhachev, Maxim et al., 2005). However, they need an understanding of complex mathematical logic, which can make users reluctant to use these algorithms.

Recently, weighted A* with the derivative of the heuristic angle (WA*DH), motivated by the concept of anytime algorithms, was suggested (Lim et al., 2020). WA*DH returns its path by getting an initial solution with a certain inflation factor and partially replans the path with the same inflation factor used in the initial solution. Because WA*DH improves the initial solution only with the direction of the path, it does not require complicated mathematical logic. Thus, WA*DH has the advantage that users can easily understand how WA*DH can improve its initial solution. However, we found that the performance of WA*DH at a certain range of inflation factors is worse than that of the larger inflation factor. We supposed that this is because of the quality of the initial solution of WA*DH and this problem makes WA*DH hard to use in the self-moving vehicles used in extreme environments.

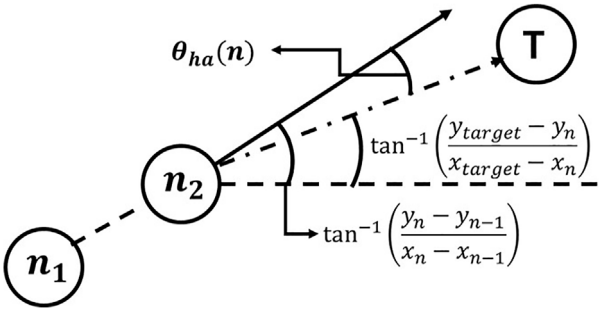

FIGURE 1
Schematic diagram for the heuristic angle.

In order to address this problem, this study introduces WA*DH+. WA*DH+ is motivated by WA*DH, so the overall procedures of WA*DH+ are the same as those of WA*DH. The difference between WA*DH and WA*DH+ is that WA*DH+ uses the GBFS algorithm to get the initial solution, whereas WA*DH uses the WA* to get the initial solution. We confirmed from the simulations that the suggested method not only reduces the elapsed times but also removes the probability of the degradation of the performance of the algorithm. From the simulation results, we believe that WA*DH+ will make self-moving vehicles used in extreme environments more responsive.

Weighted A* with the derivative of the heuristic angle

As stated in the introduction, WA*DH uses the derivative of the heuristic angle (hereinafter referred to as DH) to refine the initial solution. The heuristic angle can be defined as (1), and its schematic diagram is stated in Figure 1. In Eq. 1, $\{T, n_1, n_2\} \in \mathcal{P}$ denotes the target node, a current node, and the parent node of a current node, respectively. $\mathcal{P}$ denotes the set of nodes that consist of the path. T denotes the target, and subscripts $\{target, n, n-1\}$ are subscripts of $\{T, n_1, n_2\}$, respectively:

$$\theta_{ha} = \left|\tan^{-1}\left(\frac{y_{target}-y_n}{x_{target}-x_n}\right) - \tan^{-1}\left(\frac{y_n-y_{n-1}}{x_n-x_{n-1}}\right)\right| \quad (1)$$

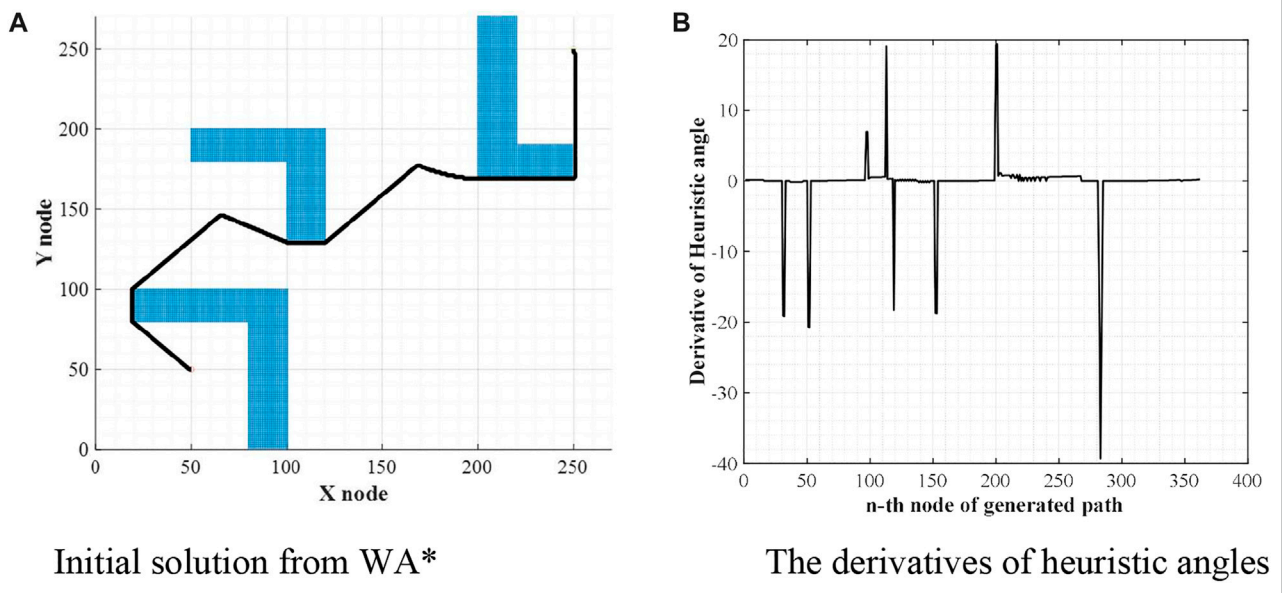

FIGURE 2
Initial solution and their derivatives of heuristic angles. **(A)** Initial solution from WA*. **(B)** The derivatives of heuristic angles.

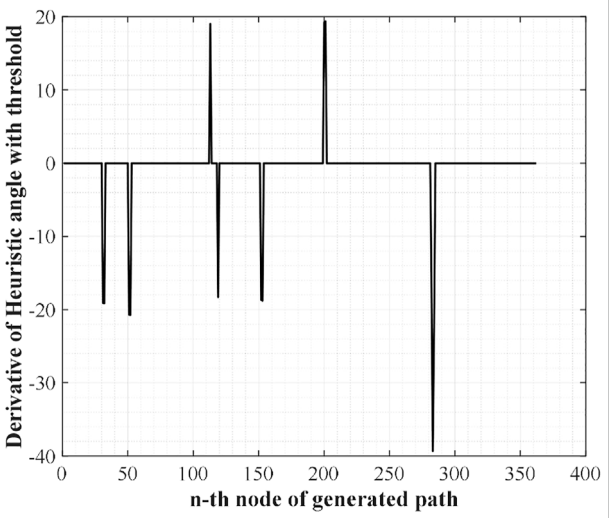

FIGURE 3
Filtered DH with a threshold.

The first step of WA*DH is getting an initial solution from WA*. Once the procedures of WA* are executed, coordinates of nodes that consist of the initial solution will be listed. With this list, WA*DH calculates heuristic angles and their derivatives. The shape of the initial solution and the derivatives of heuristic angles are stated in Figures 2A,B.

From Figure 2B, there are some noise-shaped patterns near 0. These elements can be considered suboptimal nodes by the definition of suboptimal nodes, so they must be eliminated. To do so, WA*DH filters them with a threshold defined as (2). With these methods, Figure 2B will be changed to Figure 3. In Eq. 2, $\theta'_{ha}(n)$ denotes the derivative of the heuristic angle of n_{th} node of the initial solution and m denotes the total number of nodes that consist of the initial solution:

$$\boldsymbol{Threshold} = \sum_{n=1}^{m} \left(\boldsymbol{\theta'_{ha}(n)}\right) \div \boldsymbol{m} \tag{2}$$

The next step of WA*DH is searching suboptimal nodes. The suboptimal node contains two nodes: occurrence and escape. The occurrence node is defined as a node with a positive DH, and the escape node is defined as a node with negative DH. Also, the escape node must satisfy one more condition; there must be at least one occurrence node between the start and a node with negative DH. By the definition of suboptimal nodes, we can find two occurrence nodes near the 110th and 200th nodes and three escape nodes near the 120th, 150th, and 280th nodes from Figure 3.

The purpose of searching suboptimal nodes is to make the node-set. The node-set contains the start and the target nodes for local replanning. Local replanning needs to be executed for the number of occurrence nodes. In the example environment, there are two occurrence nodes, so local replanning needs to be executed twice. The start and the

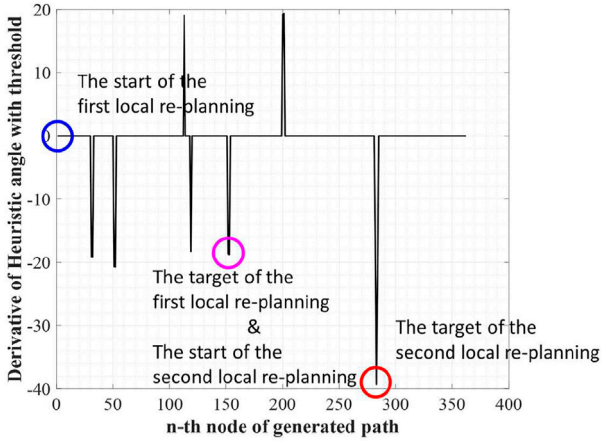

FIGURE 4
The start and the target for local replanning of the example.

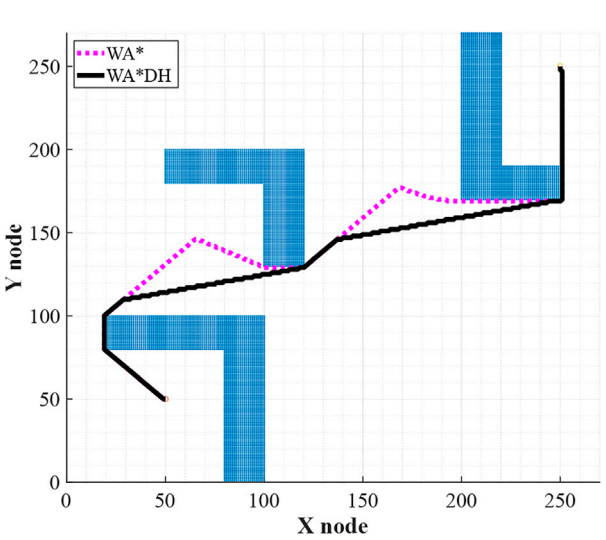

FIGURE 5
Initial and final solution of WA*DH.

target for local replanning can be determined as follows and Figure 4 shows the start and the target for local replanning of the example environment:

1) The start of the first local replanning is the start of the initial solution, and the target of the first local replanning is the first escape node. However, if there are two or more escape nodes between two occurrence nodes (or target of the initial solution), the last escape node will be chosen for the target of the local replanning.
2) The start and the target of the second and the subsequence local replanning are targets of the previous local replanning and n_{th} escape node. If there are two or more escape nodes between two occurrence nodes (or target of the initial solution), the last escape node will be chosen for the target of the local replanning.

Next, WA*DH executes WA* multiple times based on the node-set. The number of executions is the same as the number of occurrence nodes, and the inflation factor in each execution is equal to the initial solution. After that, WA*DH replaces the initial solution with the locally replanned paths. The procedure of the replacement contains not only replacing nodes but also updating $g(n)$ and $h(n)$. As a result, WA*DH returns its result. Figure 5 states the initial solution and final solution of WA*DH in a dotted line and a full line, respectively.

WA*DH+: Locally replans paths based on the infinitely bounded suboptimal solution

Critical values on WA*DH

Theoretically, the path cost of WA*DH decreases with the decreasing inflation factor. However, we found that the path cost of WA*DH with a certain inflation factor is higher than that with a larger inflation factor. This is stated in Figure 6 and Table 1. In Figures 6B,D,F, $\theta'_{ha\,f}$ denotes the filtered DH with a threshold; a dotted line and a full line in Figures 6A,C,E denote the initial and final solution of WA*DH, respectively. Table 1 states the elapsed times and path costs of each simulation.

From the initial solution in Figures 6A,C,E, it is intuitively clear that there are two detouring sections, so we can expect that there will be two escape nodes. When $\epsilon = 2$, WA*DH detected two escape nodes correctly, so the quality of the path cost is equal to A*. However, when $\epsilon = 1.8$, WA*DH detected only one escape node near the 300_{th} node and the path cost of WA*DH is higher than the path cost of WA*DH when $\epsilon = 2$. We first supposed this is because WA*DH cannot detect all suboptimal nodes. However, as stated in Figure 6D, WA*DH with $\epsilon = 1.5$ found only one escape node at the same location in Figure 6D, but its path cost is equal to A*. From these results, we concluded that the fault detection of suboptimal nodes is not the cause of the poor performance. Also, we defined inflation factors that make performance degradation critical value.

Get an initial solution from the greedy GFS algorithm

It is hard to avoid the threat of critical values because they are unpredictable. We thought that using a large inflation factor would avoid the threat of the critical value. However, it is hard to decide on a large inflation

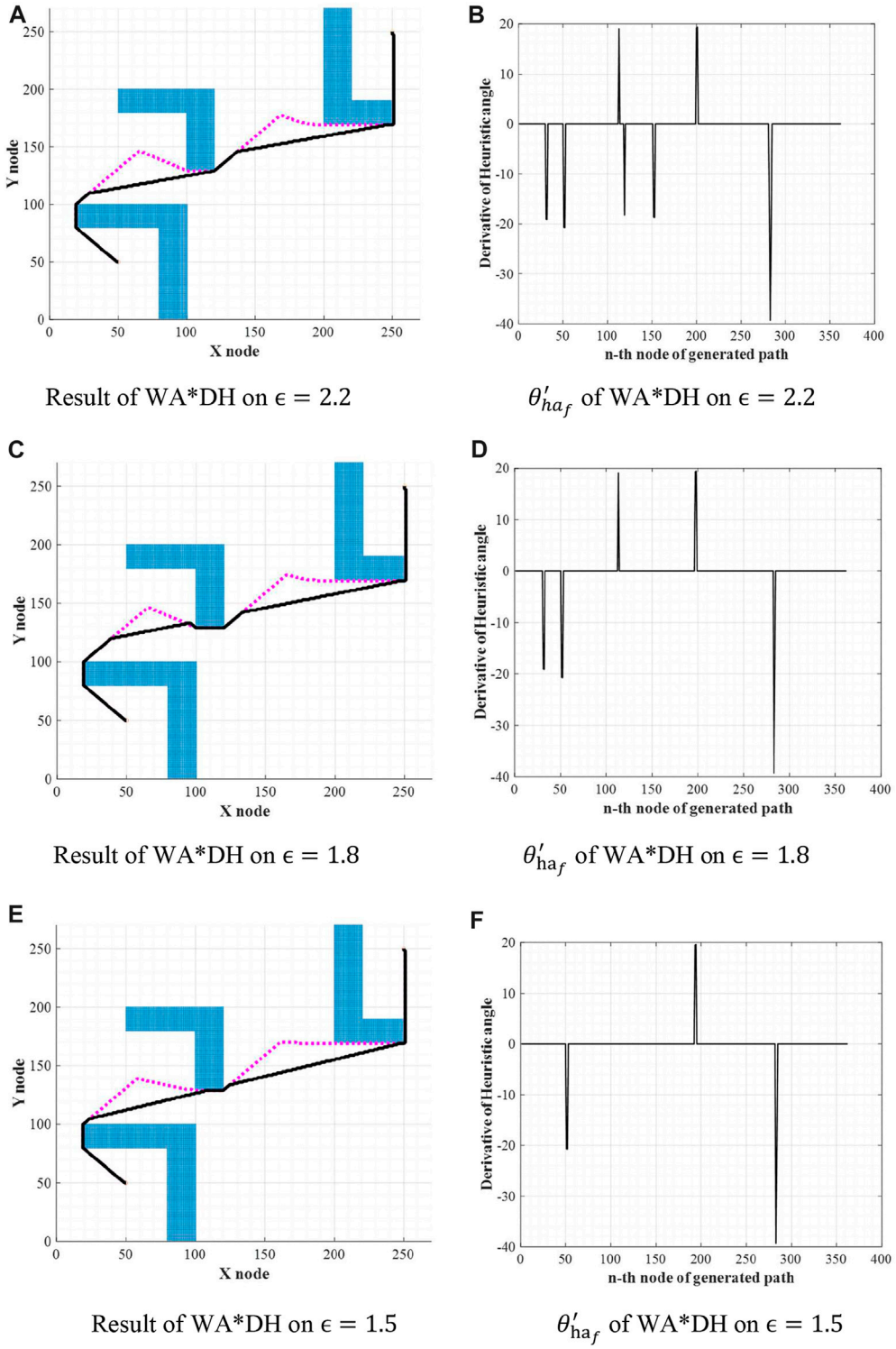

FIGURE 6
Results of WA*DH on the different inflation factors. **(A)** Result of WA*DH on ϵ = 2.2. **(B)** θ'_{ha_f} of WA*DH on ϵ = 2.2. **(C)** Result of WA*DH on ϵ = 1.8. **(D)** θ'_{ha_f} of WA*DH on ϵ = 1.8. **(E)** Result of WA*DH on ϵ = 1.5. **(F)** θ'_{ha_f} of WA*DH on ϵ = 1.5.

TABLE 1 Elapsed times and costs Figure 4.

Inflation factor	Time (s)	Cost (node)
2.2	2.04	404.84
1.8	2.49	408.15
1.5	4.08	404.84
1 (= A*)	15.40	404.84

factor because the criteria for big and small are different for each person. From this, we hypothesized that an extremely high inflation factor, such as an infinite inflation factor, will be enough to call a large inflation factor. Therefore, we suggest the greedy best-first search (GBFS) algorithm as the algorithm for the initial solution. GBFS is an algorithm that searches nodes with only heuristic, so we thought that using the GBFS algorithm is equal to using WA* with the infinite inflation factor. The cost function of GBFS is stated in Eq. 3, where *f(n)* denotes the cost function of a node of the GBFS algorithm and *h(n)* denotes the heuristic of a node:

$$f(n) = h(n) \tag{3}$$

Theorem. If the inflation factor is extremely high (or infinite), then the effect of the *g* term will be disappeared. Here, *g* is the cost of the path from the start node to the n_{th} node.

Proof.

Let the cost function of an algorithm be

$$f(\epsilon) = g + \epsilon \times h \tag{4}$$

We can change Eq. 4 to

$$\frac{f(\epsilon)}{\epsilon} = \frac{g}{\epsilon} + h \tag{5}$$

Taking the limit on both sides of Eq. 5,

$$\lim_{\epsilon\to\infty}\left(\frac{f(\epsilon)}{\epsilon}\right) = \lim_{\epsilon\to\infty}\left(\frac{g}{\epsilon}\right) + \lim_{\epsilon\to\infty}(h) \tag{6}$$

$$\frac{f(\epsilon)}{\epsilon} \approx h \tag{7}$$

$$\therefore f(\epsilon) \approx \epsilon \times h \tag{8}$$

The role of the inflation factor on the cost function, such as Eq. 4, is deciding the influence of the heuristics compared to the cost of the path, *g(n)*. However, there is no need for the inflation factor because Eq. 8 does not contain *g(n)* anymore. so we can remove ϵ from Eq. 8 With these procedures, we can derive Eq. 3 as a result.

Using the GBFS algorithm gives us some advantages, as stated in Figure 7. Figure 7 states escape nodes found from the result of WA* with $\epsilon = 1.8$, $\epsilon = 2.2$, and the GBFS algorithm, and circles in Figures 7A,C,E state the locations of occurrence nodes.

WA*DH in Figure 7B detected two occurrence nodes, so we can expect there would be two escape nodes. However, WA*DH in Figure 7 detected only one escape node. Also, in Figure 7D, WA*DH detected two occurrence nodes and two escape nodes. Moreover, WA*DH in Figure 7F found three occurrence nodes and escape nodes. In fact, considering the placement of obstacles in the simulation environment of Figure 7, there must be three occurrence nodes and escape nodes. However, Figures 7B,D could not detect all suboptimal nodes due to the relatively high optimality of WA*. From these results, we can prove that the high inflation factor can detect suboptimal nodes clearly.

The path planning with the GBFS algorithm is also very useful in terms of elapsed time. The elapsed time of WA* gets shorter as the inflation factor increases. This means that an infinite inflation factor can theoretically get a result of WA* in the fastest time. Thus, we can expect that the GBFS algorithm can reduce the elapsed time of our algorithm, WA*DH+. This will be simulated in Section 4.

Procedures of WA*DH+

This section introduces our algorithm, WA*DH+, and how WA*DH + gets its result. Procedures of WA*DH + are similar to those of WA*DH: getting an initial solution, finding escape nodes, locally replanning the paths, and connecting them. The details of procedures of WA*DH + are stated below.

First of all, WA*DH + gets an initial solution from the GBFS algorithm. Unlike WA*DH, WA*DH + does not need to decide the inflation factor for the initial solution. After getting an initial solution, then WA*DH+ calculates θ'_{haf} with the moving median filter and the threshold.

The next step of WA*DH+ is finding the suboptimal nodes to decide the start and the targets for local replanning. Next, WA*DH+ executes the local replanning. In this procedure, WA*DH+ needs to decide on an inflation factor.

The final step of WA*DH+ is path-connecting. In this procedure, WA*DH + needs to update only heuristics, whereas WA*DH needs to update the actual costs, *g(n)*, and heuristics. Also, WA*DH+ needs an additional procedure to calculate the path cost, whereas WA*DH can get its path cost from *g(n)* of the target. After calculating the path cost, WA*DH+ can finally get its result. The pseudocode and the flowchart of WA*DH+ are stated in Figures 8, 9. In the flowchart, when $i = 1$, the $(i-1)_{th}$ node is equal to the original start node.

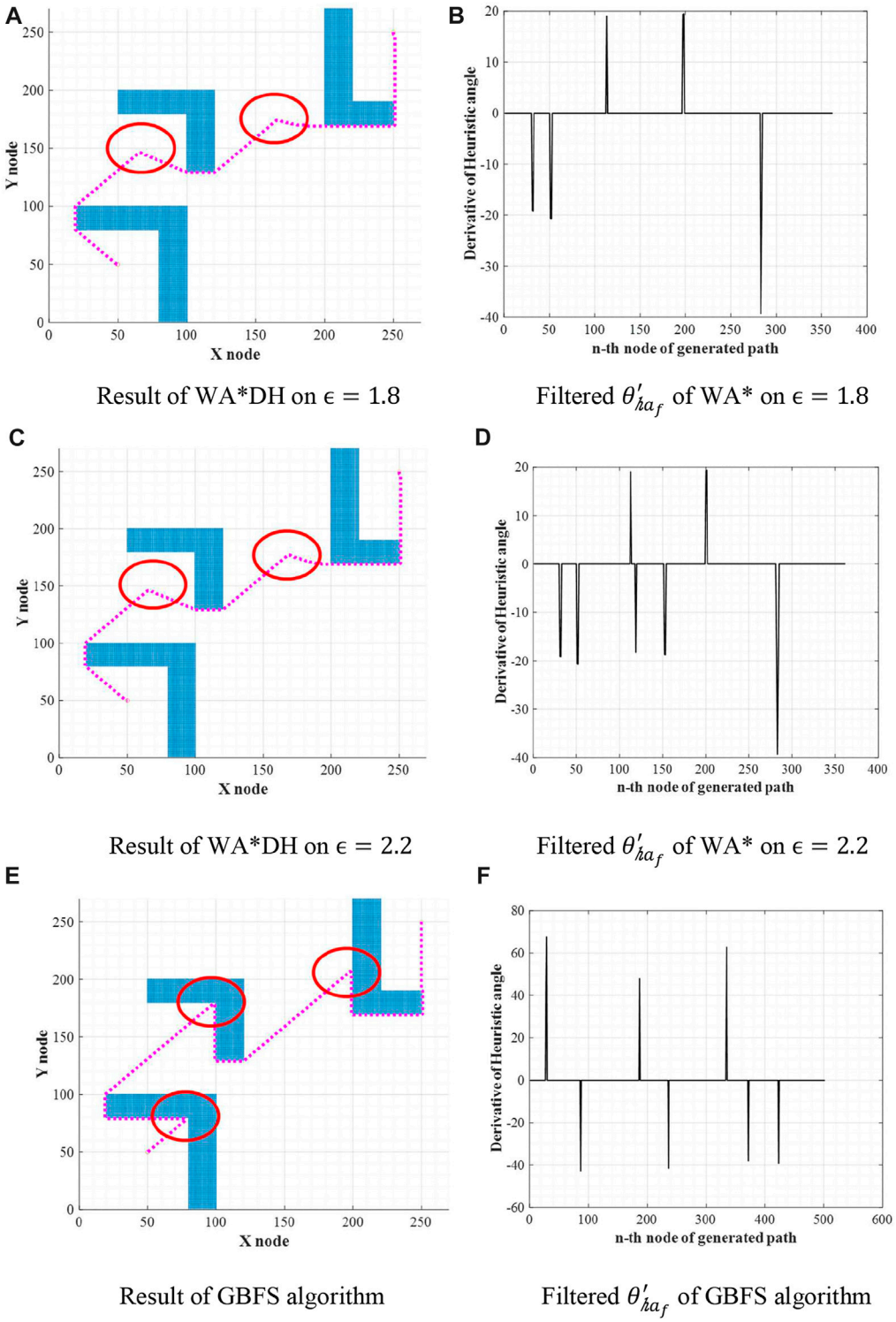

FIGURE 7
Escape nodes of WA* on different inflation factors. **(A)** Result of WA*DH on ϵ = 1.8. **(B)** Filtered θ'_{ha_f} of WA* on ϵ = 1.8. **(C)** Result of WA*DH on ϵ = 2.2. **(D)** Filtered θ'_{ha_f} of WA* on ϵ = 2.2. **(E)** Result of the GBFS algorithm. **(F)** Filtered θ'_{ha_f} of the GBFS algorithm.

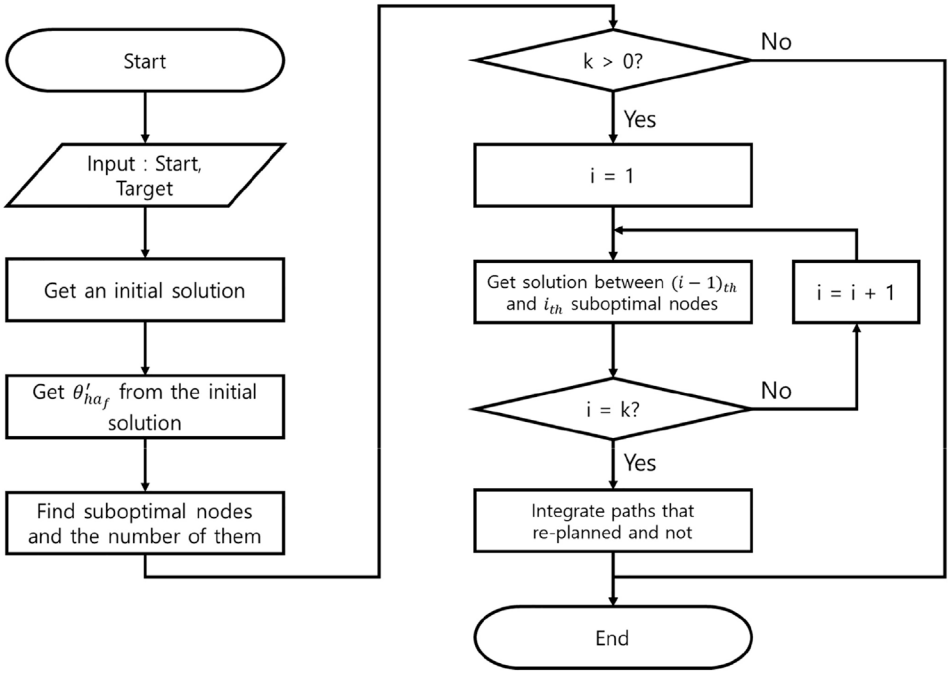

FIGURE 8
Flowchart of WA*DH+.

Simulation results

Simulation environments

In order to compare the performances with WA*DH, obstacles were placed the same as in the simulation environments of WA*DH, and the sizes of all simulation environments are also the same as those of the simulation environments of WA*DH (270 x 27 nodes) as stated in Figure 10. All simulations were conducted in MATLAB with Windows 10, i7-9700 CPU with 32 GB RAM, and there are no acceleration methods or parallel processes such as Graphics Processing Unit (GPU) and parallel processing with CPU cores.

Performances of WA*DH+

To validate the performance of WA*DH+, we simulated WA*DH+, WA*DH, and WA* in terms of the path cost and elapsed time in each environment. Simulations were conducted by reducing the inflation factor from 3 to 1 by 0.2 to compare performances near critical values. Results of simulations are stated in Figures 11–14, and quantitative comparisons are stated in Tables 2–5.

From Table 2, the path cost of WA*DH with a relatively large inflation factor (> 2.2) is the same path cost of A*'s. However, when ϵ is in the range from 2.2 to 1.6, the path cost of WA*DH is about 0.81% larger than that with larger inflation factors. In contrast, the path cost of WA*DH + does not change with varying inflation factors, and its quality keeps the same as the path cost of A*.

Table 3 shows that path costs of WA*DH and WA*DH+ with relatively high inflation factors (> 2.0) are the same as the path cost of A*. However, when the inflation factor is lower than 2, the path cost of WA*DH is the same as WA* until the inflation factor decreases to 1, whereas the path cost of WA*DH + does not change with varying inflation factors.

In the case of simulation case 3, the path cost of WA*DH+ is about 5.04% lower than that of WA*DH. Thus, we can see that WA*DH + not only removes the risk of a critical value but also returns a lower path cost at a relatively high inflation factor. Moreover, we also confirmed that the path cost of WA*DH + keeps the same quality as the path cost of A* regardless of the inflation factor.

Unlike other results of simulation cases, when $\epsilon = 3$ in case of simulation case 4, the path cost of WA*DH+ is about 0.5% higher than that of WA*DH, and this difference rises to about 1.47% when $\epsilon = 1.6$. However, it is hard to say that the quality of WA*DH+ is bad because the purpose of using WA*DH+ is to avoid the threat of the critical value.

Besides being free from the threat of the critical value, we also found that WA*DH + has an advantage in terms of the

Algorithm 1. WA*DH+(Weighted A* with the Derivative of Heuristic angle +)

Procedure Main()

GBFS()
$get\ \theta'_{ha}\ from\ all\ nodes\ on\ the\ path$
$node\_set = detect\_nodes(\theta'_{ha})$
$replannd\_path = local\_replanning(node\_set)$
$Result = connect\_paths(replanned\_path, original\_path)$
$return\ Result$

Procedure GBFS()

$h(n_{start}) = c(start, target);$
$OPEN = OPEN\ \cup (n_{start})$
$CLOSED = \emptyset;$
$while\ (n_n! = n_{target})$
    $remove\ n\ with\ the\ smallest\ f(n) from\ OPEN$
  $CLOSED = CLOSED \cup \{n\};$
  for each successor n′ of n
    $if\ n'$ was not visited before then
      $insert\ n' into\ OPEN\ with\ f(n')$

Procedure WA^*()

$g(n_{start}) = 0;$
$OPEN = CLOSED = \emptyset;$
$while\ (n_n! = n_{target})$
    $remove\ n\ with\ the\ smallest\ f(n) from\ OPEN$
  $CLOSED = CLOSED \cup \{n\};$
  for each successor n′ of n
    $if\ n'\ was\ not\ visited\ before, then$
      $g(n') = \infty;$
    $if\ g(n') > g(n) + c(n, n')$
      $g(n') = g(n) + c(n, n')$
      $if\ n' \notin CLOSED$
        $insert\ n' into\ OPEN\ with\ f(n')$

Procedure detect_nodes()

$node_{set} = \emptyset;\ N = \emptyset;$
$i = 0;\ flag = 0;\ size\_n = 0;$
$while\ i < size(\theta'_{ha}())$
  i++;
  $if\ \theta'_{ha}(i) > 0$
    flag = 1;
  $while\ flag = 1\ \&\&\ i < size(\theta'_{ha}())$
    $i++;$
    $if\ \theta'_{ha}(i) < 0$
      size_n++;
      N(size_n)=i;
      $while\ \theta'_{ha}(i) < 0\ \&\&\ i < size(\theta'_{ha}())$
        i++;
        $if\ \theta'_{ha}(i) < 0$
          N(size_n)=i;
        flag=0;
for n=1:size(N)
  if n=1
    node_set(n)=[start_node original_path(N(n))];
  else
node_set(n)=[original_path(N(n-1) original_path(N(n))];
return node_set;

Procedure local_replanning()

replanned_path=∅;
for j=1:size(col(node_set))
  if j=1
    replanned_path = WA^*_ϵ(node_set(j));
  else
    local_path = WA^*_ϵ (node_set(j));
    replanned_path = [replanned_path; local_path];
return replanned_path

Procedure connect_paths()

nr_path = original_path(node_set(end):end);
for all h of nodes of replanned_path
  f(n)=c(n, goal);
update f of replanned_path with changed h
final_path=[replanned_path; nr_path];
return final_path

FIGURE 9
Pseudocode of WA*DH+.

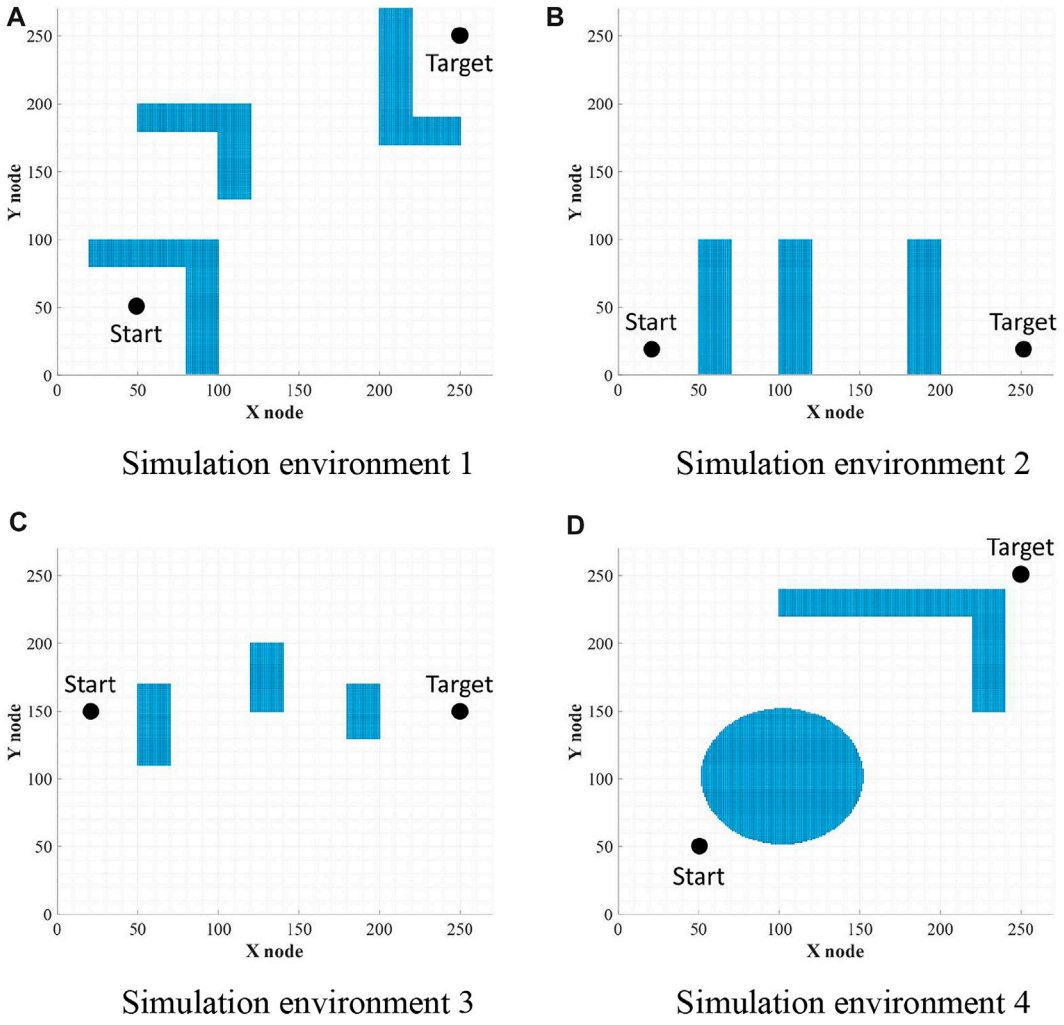

FIGURE 10
Environments for simulations. **(A)** Simulation environment 1. **(B)** Simulation environment 2. **(C)** Simulation environment 3. **(D)** Simulation environment 4.

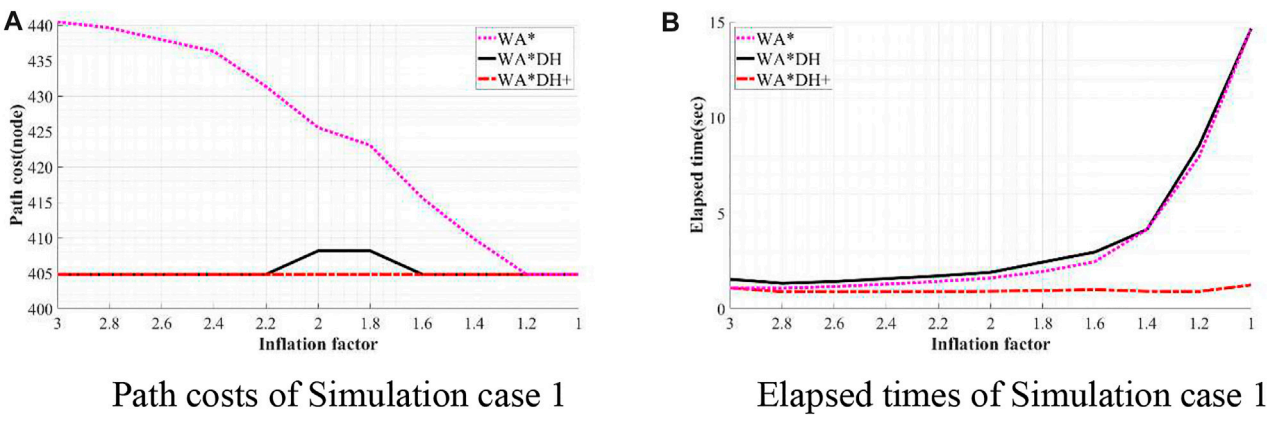

FIGURE 11
Result of simulation case 1. **(A)** Path costs of simulation case 1. **(B)** Elapsed times of simulation case 1.

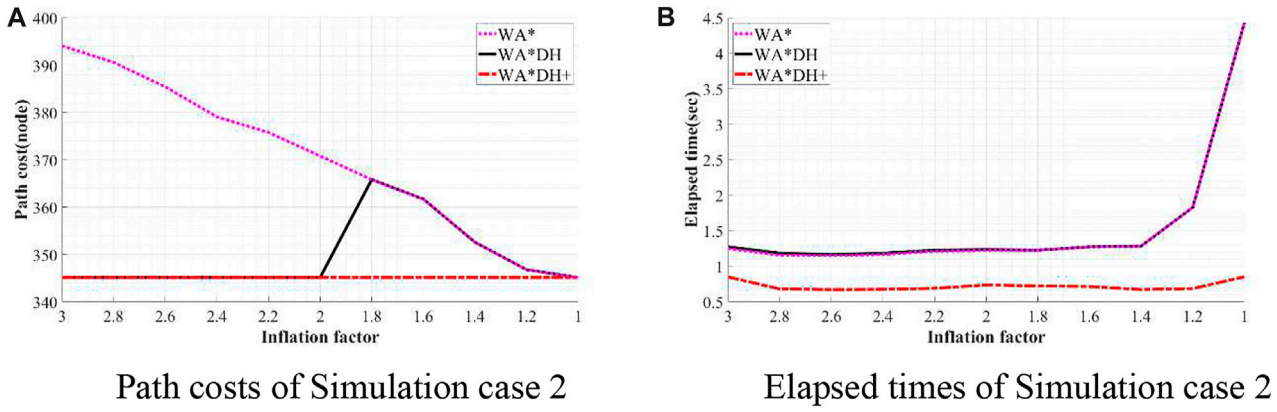

FIGURE 12
Result of simulation case 2. **(A)** Path costs of simulation case 2. **(B)** Elapsed times of simulation case 2.

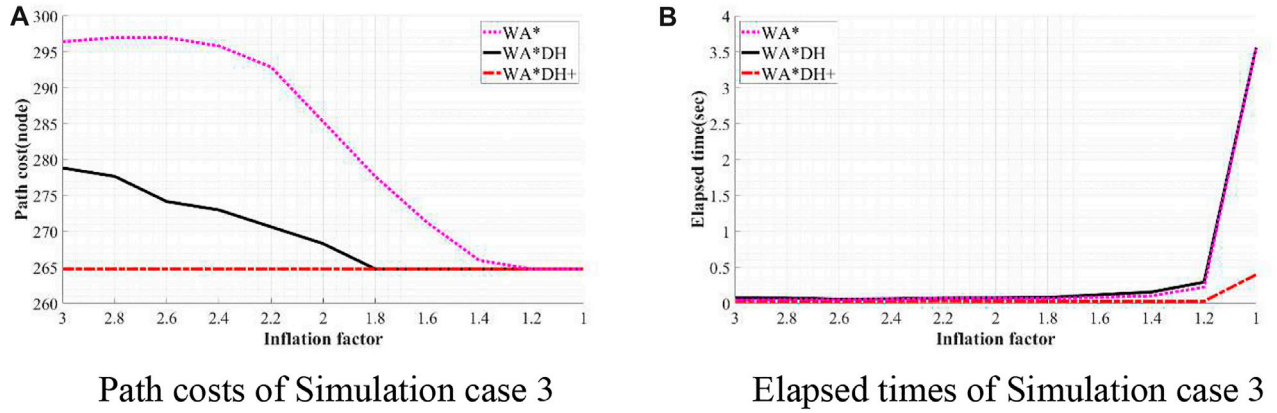

FIGURE 13
Results of simulation case 3. **(A)** Path costs of simulation case 3. **(B)** Elapsed times of simulation case 3.

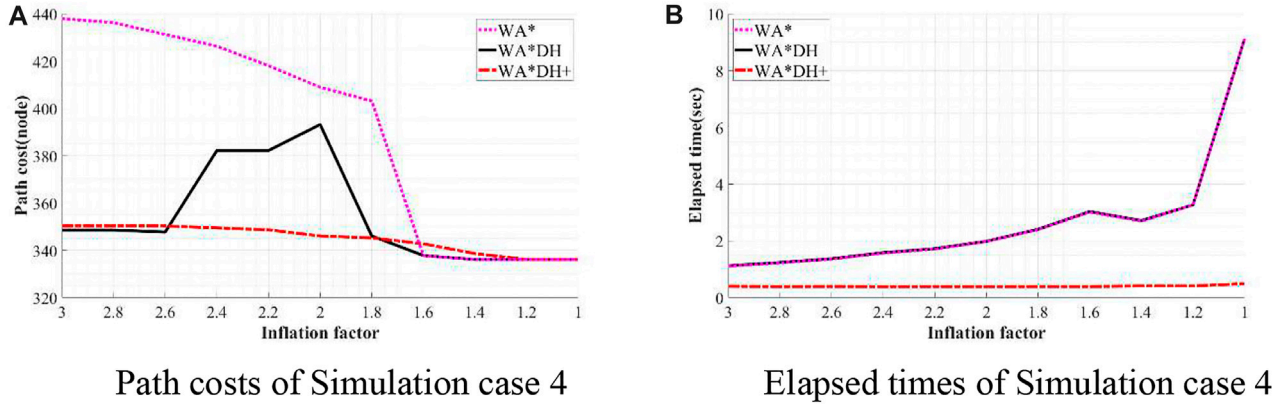

FIGURE 14
Result of simulation case 4. **(A)** Path costs of simulation case 4. **(B)** Elapsed times of simulation case 4.

TABLE 2 Quantitative comparison of the performances of algorithms on simulation 1.

€	WA*DH+		WA*DH		WA*	
	Cost (node)	Time (s)	Cost (node)	Time (s)	Cost (node)	Time (s)
3	404.84	1.08	404.84	1.54	440.46	1.08
2.8	404.84	0.88	404.84	1.33	439.63	1.07
2.6	404.84	0.88	404.84	1.41	437.97	1.15
2.4	404.84	0.88	404.84	1.57	436.32	1.28
2.2	404.84	0.88	404.84	1.71	431.35	1.42
2.0	404.84	0.91	408.15	1.90	425.55	1.60
1.8	404.84	0.94	408.15	2.43	423.06	1.94
1.6	404.84	0.99	404.84	2.96	415.61	2.46
1.4	404.84	0.90	404.84	4.13	409.81	4.14
1.2	404.84	0.89	404.84	8.52	404.84	7.95
1.0	404.84	1.24	404.84	14.65	404.84	14.65

TABLE 3 Quantitative comparison of the performances of algorithms on simulation 2.

€	WA*DH+		WA*DH		WA*	
	Cost (node)	Time (s)	Cost (node)	Time (s)	Cost (node)	Time (s)
3	345.14	0.85	345.14	1.27	394.13	1.25
2.8	345.14	0.68	345.14	1.18	390.62	1.15
2.6	345.14	0.67	345.14	1.16	385.44	1.15
2.4	345.14	0.68	345.14	1.18	379.1	1.17
2.2	345.14	0.69	345.14	1.22	375.79	1.21
2.0	345.14	0.74	345.14	1.23	370.82	1.22
1.8	345.14	0.72	365.85	1.23	365.85	1.23
1.6	345.14	0.72	361.71	1.27	361.71	1.27
1.4	345.14	0.67	352.59	1.29	352.59	1.29
1.2	345.14	0.69	346.79	1.83	346.79	1.83
1.0	345.14	0.85	345.14	4.43	345.14	4.43

TABLE 4 Quantitative comparison of the performances of algorithms on simulation 3.

€	WA*DH+		WA*DH		WA*	
	Cost (node)	Time (s)	Cost (node)	Time (s)	Cost (node)	Time (s)
3	264.79	0.03	278.85	0.08	296.43	0.05
2.8	264.79	0.03	277.68	0.08	297.01	0.05
2.6	264.79	0.03	274.17	0.06	297.01	0.05
2.4	264.79	0.03	273.00	0.06	295.84	0.06
2.2	264.79	0.04	270.65	0.08	292.91	0.07
2.0	264.79	0.03	268.31	0.08	285.3	0.07
1.8	264.79	0.03	264.79	0.08	277.68	0.07
1.6	264.79	0.03	264.79	0.12	271.24	0.08
1.4	264.79	0.03	264.79	0.16	265.97	0.10
1.2	264.79	0.03	264.79	0.29	264.79	0.23
1.0	264.79	0.40	264.79	3.56	264.79	3.56

TABLE 5 Quantitative comparison of the performances of algorithms on simulation 4.

€	WA*DH+		WA*DH		WA*	
	Cost (node)	Time (s)	Cost (node)	Time (s)	Cost (node)	Time (s)
3	350.33	0.40	348.58	1.12	437.97	1.11
2.8	350.33	0.39	348.58	1.23	436.32	1.23
2.6	350.33	0.40	347.75	1.37	431.34	1.36
2.4	349.50	0.38	382.23	1.59	426.37	1.57
2.2	348.68	0.39	382.23	1.73	418.09	1.71
2.0	346.09	0.39	393.24	1.99	409.00	1.98
1.8	345.26	0.39	346.09	2.41	403.18	2.40
1.6	342.78	0.39	337.81	3.03	337.81	3.02
1.4	338.63	0.41	336.15	2.72	336.15	2.69
1.2	336.15	0.41	336.15	3.27	336.15	3.27
1.0	336.15	0.49	336.15	9.13	336.15	9.13

elapsed time. In all simulation cases, elapsed times of WA*DH increase exponentially near $\epsilon = 1$. However, the elapsed time of WA*DH + does not significantly increase with decreasing inflation factor, and elapsed times of WA*DH + are much lower than WA*DH. In fact, considering that using the GBFS algorithm has the same meaning as using the infinite inflation factor at WA*, it is a natural result because the higher the inflation factor is, the faster the result can be returned.

Discussion

This study aims to evade the threat of the critical values on WA*DH. We found that the critical value comes from the fault detection of suboptimal nodes from the initial solution with relatively high optimality. We hypothesized that high suboptimality could find suboptimal nodes clearly, so we suggested our algorithm, WA*DH +, which uses the GBFS algorithm for the initial solution. From simulations, it can be proven that WA*DH + can avoid the threat of the critical value successfully by detecting suboptimal nodes more clearly than WA*DH. In terms of the elapsed time, we also confirmed that the elapsed time does not change significantly with varying inflation factors; however, the elapsed time of WA*DH increases exponentially when the inflation factor is near 1.

Although WA*DH + shows powerful performances in terms of the path cost and the elapsed time, WA*DH + still cannot guarantee admissibility because WA*DH + refines the initial solution based on the GBFS algorithm that has the infinitely bounded suboptimality. It will remain a limitation of algorithms using the concept of WA*DH. Also, WA*DH+ cannot refine the initial solution if there are circular obstacles in the path planning scene. However, we expect this will be removed in future works using new filtering methods of DH or additional procedures to the result of WA*DH+.

Data availability statement

The original contributions presented in the study are included in the article. Further inquiries can be directed to the corresponding author.

Author contributions

DL proposed the idea and simulated the algorithm and wrote the paper except the simulation part. JJ analyzed the result of the computer simulation and wrote the computer simulation part of the paper.

Funding

This research was a part of project “No. 20190497,” funded by the Korea Coast Guard, Korea.

Conflict of interest

DL and JJ were employed by Vessel aerospace Co.Ltd.

Publisher’s note

References

Aine, S., Swaminathan, S., Narayanan, V., Hwang, V., and Likhachev, M. (2016). Multi-heuristic a. *Int. J. Robotics Res.* 35 (1-3), 224–243. doi:10.1177/0278364915594029

Boulares, M., and Barnawi, A. (2021). A novel UAV path planning algorithm to search for floating objects on the ocean surface based on object's trajectory prediction by regression. *Robotics Aut. Syst.* 135, 103673. doi:10.1016/j.robot.2020.103673

Dai, X., Long, S., Zhang, Z., and Gong, D. (2019). Mobile robot path planning based on ant colony algorithm with A* heuristic method. *Front. Neurorobot.* 13, 15. doi:10.3389/fnbot.2019.00015

De Momi, E., Kranendonk, L., Valenti, M., Enayati, N., and Ferrigno, G. (2016). A neural network-based approach for trajectory planning in robot–human handover tasks. *Front. Robot. AI* 3, 34. doi:10.3389/frobt.2016.00034

Duan, C., Hu, Q., Zhang, Y., and Wu, H. (2020). Constrained single-axis path planning of underactuated spacecraft. *Aerosp. Sci. Technol.* 107, 106345. doi:10.1016/j.ast.2020.106345

Fu, B., Chen, L., Zhou, Y., Zheng, D., Wei, Z., Dai, J., et al. (2018). An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Robotics Aut. Syst.* 106, 26–37. doi:10.1016/j.robot.2018.04.007

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cyber.* 4 (2), 100–107. doi:10.1109/tssc.1968.300136

Hou, M., Cho, S., Zhou, H., Edwards, C. R., and Zhang, F. (2021). Bounded cost path planning for underwater vehicles assisted by a time-invariant partitioned flow Field model. *Front. Robot. AI* 8, 575267. doi:10.3389/frobt.2021.575267

Jing, X., and Yang, X. (2018).Application and improvement of heuristic function in A* algorithm2018 37th Chinese Control Conference (CCC), 25-27 July 2018, Wuhan, China.

Li, B., Gong, J., Jiang, Y., Nasry, H., and Xiong, G. (2012).ARA*+: Improved path planning algorithm based on ARAIEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 04-07 December 2012, Macau, China.

Likhachev, M. (2005). Anytime dynamic A*: An anytime, Replanning Algorithm.. *ICAPS.* 5, 262–271.

Likhachev, M., Gordon, G. J., and Thrun, S. (2003). ARA*: Anytime A* with provable bounds on sub-optimality. *Adv. neural Inf. Process. Syst.* 16, 767–774.

Lim, D., Park, J., Han, D., Jang, H., Park, W., Lee, D., et al. (2020). UAV path planning with derivative of the heuristic angle. *Int. J. Aeronaut. Space Sci.* 22, 140–150. doi:10.1007/s42405-020-00323-1

Liu, Z., Yue, M., Guo, L., and Zhang, Y. (2020). Trajectory planning and robust tracking control for a class of active articulated tractor-trailer vehicle with on-axle structure. *Eur. J. Control* 54, 87–98. doi:10.1016/j.ejcon.2019.12.003

Mac, T. T., Copot, C., Tran, D. T., and De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics Aut. Syst.* 86, 13–28. doi:10.1016/j.robot.2016.08.001

Pearl, J., and Kim, J. H. (1982). Studies in semi-admissible heuristics. *IEEE Trans. Pattern Anal. Mach. Intell.* 4, 392–399. doi:10.1109/tpami.1982.4767270

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artif. Intell.* 1 (3-4), 193–204. doi:10.1016/0004-3702(70)90007-x

Pohl, I. (1973). The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In Proceedings of the 3rd international joint conference on Artificial intelligence, Stanford USA, August 20–23, 1973, 12–17.

Thayer, J., and Ruml, W. (2009). Using distance estimates in heuristic search. In Proceedings of the International Conference on Automated Planning and Scheduling, Thessaloniki, September 19–23, 2009 (Vol. 19, 1).

Thayer, J. T., and Ruml, W. (2008). Faster than weighted A*: An optimistic approach to bounded suboptimal search. In ICAPS, New Delhi, India, June, 2008, 355–362.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J., and Xia, Y. (2016). Survey of robot 3D path planning algorithms. *J. Control Sci. Eng.*, 1–22. doi:10.1155/2016/7426913

Yang, L., Qi, J., Xiao, J., and Yong, X. (2014). A literature review of UAV 3D path planningProceeding of the 11th World Congress on Intelligent Control and Automation, 29 June 2014 - 04 July 2014, Shenyang

Yiu, Y. F., Du, J., and Mahapatra, R. (2018).Evolutionary heuristic a* search: Heuristic function optimization via genetic algorithmIEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)26-28 September 2018, Laguna Hills, CA, USA.

Zhou, R., and Hansen, E. A. (2002) Multiple sequence alignment using A*. In Proc. of the National Conference on Artificial Intelligence (AAAI), Alberta, Canada, July 28–August 1, 2002.