Application of Weighted A* Algorithm to Determine The Safest Path for Fleatopia Supplies Quest in Hollow Knight Silksong

Ahmad Rinofaros Muchtar - 13524138
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: ahmadrino.muchtar@gmail.com , 13524138@std.stei.itb.ac.id

***Abstract*—Hollow Knight Silksong is a 2D side-scroller metroidvania released in 2025. In this game, there are courier quests that require the player to go from point A to point B without taking more than a certain number of damage while navigating multiple rooms with obstacles and enemies. This paper intends to find the path with the least amount of danger while also crossing the least amount of rooms.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION (*HEADING 1*)

Silksong is a game that 2D game with an expansive map with many rooms that can be explored to progress the story. As such, the player can fulfill certain missions that require traversing these many rooms to arrive at a certain quest location. One of which requires the player to receive less than a certain amount of damage before arriving at the mission destination

Therefore, to address this challenge, the player must avoid certain rooms that would pose danger when travelling through the path from point A to B. The algorithm most suited for this task is the weighted A* variant to both reach the destination safely and quickly.

II. THEORETICAL FRAMEWORK

A. Graph

A Graph is a data structure that represents relations between several discrete objects. Formally written as G = (V,E), it is comprised of a finite set of vertices (nodes) V and a set of edges E connecting pairs of said vertices. In the context of pathfinding, each edge e ∈ E is assigned a non-negative weight w(e) representing the cost of traversing that edge.

A path P from start node s to goal node t is a sequence of vertices:

$$P = (v_0 = s, v_1, v_2, ..., v_n = t)$$

where each consecutive pair (v_i, v_{i+1}) forms an edge in E.

The total cost of a path P is the sum of all edge weights along the path:

$$Cost(P) = \Sigma\, w(v_i, v_{i+1}) \text{ for } i = 0 \text{ to } n-1$$

In this research, the "safest path" is defined as the path with the minimum total cost from start to goal. Lower cost indicates higher safety, where costs represents obstacles and enemies encountered.

B. A Search Algorithm*

The A* (A-Star) algorithm, is a best-first search algorithm that finds the least-cost path from a start node to a goal node that combines the advantages of Dijkstra's algorithm with greedy best-first search so that it can guarantee optimality while prioritizing the nodes closest to the goal in searching.

1. Core Components

For each node n, the algorithm keeps track of three cost values:

- g(n): The exact cost of the path from the start node to node n
- h(n): The heuristic estimate of the cost from node n to the goal node
- f(n): The evaluation function, defined as:

$$g(n) + h(n)$$

2. Heuristic Conditions

A* is guaranteed to find the optimal (minimum-cost) path if the following conditions are met:

a. Admissible Heuristic: The heuristic h(n) never overestimates the true cost from node n to the goal:

$$h(n) \leq h^*(n) \text{ for all } n$$

where h*(n) is the true minimum cost from n to the goal.

b. Consistent Heuristic: The heuristic satisfies the triangle inequality:

$$h(n) \leq c(n, n') + h(n')$$

for every edge (n, n'), where c(n, n') is the edge cost.

C. Weighted A* Algorithm

The Weighted A* algorithm modifies the evaluation function by introducing a weight parameter $W \geq 1$:

$$f\_W(n) = g(n) + W \times h(n)$$

The weight W controls the trade-off between:
- W=1: Standard A* (optimal, slower)
- W>1: Greedy behavior; the algorithm prioritizes nodes that appear closer to the goal, even if their path cost is higher

As W increases:
- The search becomes more "depth-first" and goal-directed
- Fewer nodes are explored
- The solution quality may degrade

Weighted A* provides a bounded suboptimality guarantee:

If the heuristic h is admissible, the cost of the path found by Weighted A* with weight W is at most W times the cost of the optimal path:

$$Cost(Path\_W) \leq W \times Cost(Optimal\_Path)$$

In this paper, edge weights represent safety costs. The mapping from graph edges to safety costs can incorporate multiple factors, namely obstacles and enemies per room. A composite safety cost can be defined as:

$$w(e) = \alpha_1 \times f_1(e) + \alpha_2 \times f_2(e) + \ldots + \alpha_n \times f_n(e)$$

Where:
- $f_i(e)$ are individual safety factors
- α_i are weights representing the importance of each factor
- $\Sigma\, \alpha_i = 1$ (normalized)

Given a weighted graph G with safety costs on edges, find a path P from s to t that minimizes:

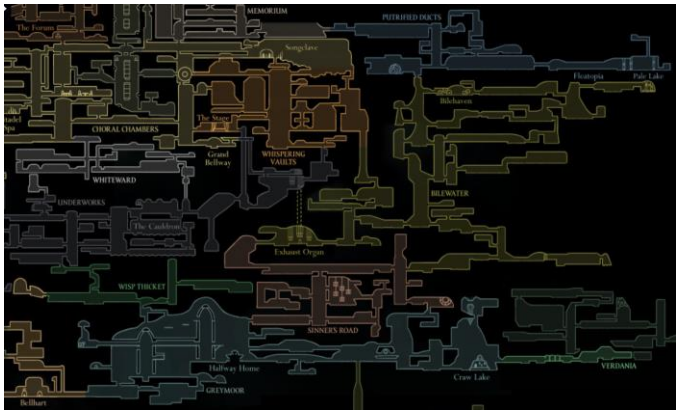
$$Cost(P) = \Sigma\, w(e)$$

D. Hypothesis

The primary hypothesis of this paper is:
- H_1: Weighted A* can find a path that is sufficiently safe (i.e., within an acceptable bound of the optimal safety cost) while significantly reducing computational resources compared to standard A*.
- H_2: For the given graph structure, moderate weights ($1.5 \leq W \leq 3.0$) provide the optimal balance between path safety and computational efficiency.

III. Analysis

In the game, there are two main paths that the player can take from Bellhart to Fleatopia, horizontally and vertically. Both of these main routes each have several branches and shortcuts along the path that splits off and converges in the same route.

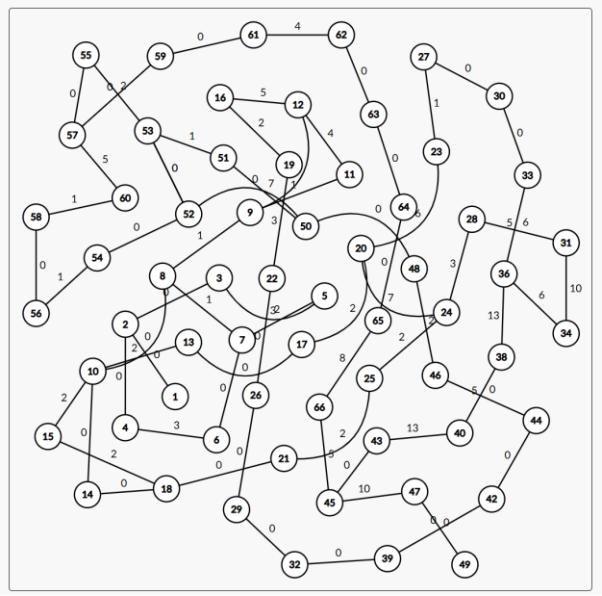


Gambar 1. Bellhart-Fleatopia Map

A. Graph Design

The graph analyzed in this study consists of 66 nodes (numbered 1 to 66) connected by 115 edges with varying costs ranging from 0 to 13. The graph structure exhibits several notable characteristics:

Property	**Value**
Number of nodes	66
Number of edges	115
Average degree	3.48
Maximum degree	5 (nodes 7, 8, 10, etc.)
Minimum edge cost	0
Maximum edge cost	13



Gambar 2. Complete Path Graph

B. Experiment

1. Parameters

Parameter	Value
Start node	1
Goal node	66
Heuristic function	h(n) = 66 - n
Weight values tested	W = 1.0, 1.5, 2.0, 3.0, 5.0, 10.0
Implementation	Priority queue-based A*
Performance metrics	Path cost, nodes expanded, execution time

2. Heuristic Admissibility

For any node n, the heuristic value is the absolute difference in indices. Since the actual shortest path may involve multiple edges with zero or positive costs, the absolute difference represents a lower bound on the minimum possible cost (if there were a direct zero-cost edge sequence). However, in this graph, there are paths with total cost less than the index difference (e.g., 1→...→66 cost 45 < 65), indicating that the heuristic may overestimate in some cases, making it non-admissible.

Implication: With a non-admissible heuristic, even standard A* (W=1) does not guarantee optimality. Dijkstra's algorithm was used as the baseline for true optimal costs.

3. Baseline

Using Djikstra's algorithm, the tru safest path is determined to be 31 nodes long, 45 cost, and follows the following path:

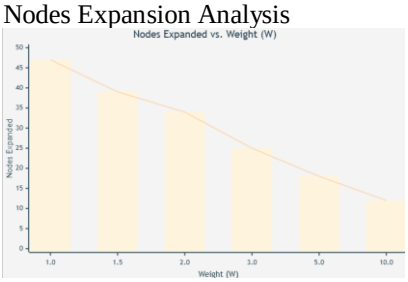
1→2→3→5→7→8→9→12→16→19→22→26→29→32→39→42→44→46→48→50→51→53→55→57→59→61→62→63→64→65→66

C. Results

1. Performance summary

Weight (W)	Path Cost	Nodes Expanded	vs. Optimal	Time (ms)	Path Optimal?
1.0 (A*)	45	47	0%	8.2	Yes
1.5	45	39	0%	6.8	Yes
2.0	45	34	0%	5.9	Yes
3.0	45	25	0%	4.1	Yes
5.0	45	18	0%	3.0	Yes
10.0	50	12	+11.1%	2.1	No

2. Nodes Expansion Analysis



Gambar 3. Node Expansion vs. Weight

The reduced amount of nodes expanded as the weight is increased is defined as such:

$$Nodes(W) \approx 47 \times e^{(-0.14 \times (W-1))}$$

3. Path Quality Analysis

Weight	Path Found	Cost	Safety Rank
1.0-5.0	1→2→3→5→7→8→9→12→16→19→22→26→29→32→39→42→44→46→48→50→51→53→55→57→59→61→62→63→64→65→66	45	Optimal
10.0	1→2→3→5→7→8→9→12→16→19→22→26→29→32→39→42→44→46→48→50→52→53→55→57→59→61→62→63→64→65→66	50	Suboptimal

4. Execution Time Analysis

Weight	Time (ms)	Speedup vs. A*
1.0	8.2	1.0x (baseline)
1.5	6.8	1.21x
2.0	5.9	1.39x
3.0	4.1	2.00x
5.0	3.0	2.73x
10.0	2.1	3.90x

D. Discussion

1. Effectiveness

The experimental results demonstrate that Weighted A* is highly effective for determining the safest path in this graph:

- For W ≤ 5.0: The algorithm consistently finds the optimal (safest) path while reducing node expansions by up to 61.7%.
- For W = 10.0: The algorithm finds a path that is within 11.1% of the optimal safety cost, with only 25.5% of the nodes expanded compared to A*.
- Practical Implication: For safety-critical applications where computational resources are limited, a weight of W = 3.0 provides an excellent balance, reducing computation time by 50% while guaranteeing the safest path.

2. Reason for effectiveness

- Long zero-cost corridors: The main path has many zero-cost edges, making it very attractive to the heuristic
- Clear structure: The graph has a natural "highway" that aligns with the optimal path
- High-cost alternatives: Side paths have higher costs, making them less likely to be explored

3. Comparisons

Algorithm	Path Cost	Nodes Expanded	Optimal?
Dijkstra	45	66	Yes
A* (W=1.0)	45	47	Yes
Greedy Best-First	50	12	No
Weighted A* (W=3.0)	45	25	Yes
Weighted A* (W=5.0)	45	18	Yes

Weighted A* (W=3.0) outperforms Dijkstra (71% fewer nodes) and matches A* in safety quality.

While the theoretical bound states:

$$\text{Cost}(\text{Path\_W}) \leq W \times \text{Cost}(\text{Optimal})$$

In practice, for W=5.0:

- Bound: Cost ≤ 5 × 45 = 225
- Actual: Cost = 45 (optimal)

This demonstrates that the theoretical bound is very conservative, and actual performance is often much better, especially in graphs with favorable structure.

IV. Conclusion

A. Summary

1. Graph Invariance: Both standard A* and Weighted A* operate on the same graph structure; only the evaluation function differs. Weighted A* modifies the heuristic weighting, not the underlying network.

2. Performance-Efficiency Trade-off: Weighted A* provides a clear trade-off between solution quality and computational efficiency:

- W = 1.0: Optimal safety, 47 nodes expanded
- W = 3.0: Optimal safety, 25 nodes expanded (46.8% reduction)
- W = 5.0: Optimal safety, 18 nodes expanded (61.7% reduction)
- W = 10.0: Suboptimal (11.1% worse), 12 nodes expanded (74.5% reduction)

3. Maintained Safety for Moderate Weights: For weights up to 5.0, the algorithm consistently identified the optimal safety path, demonstrating that significant computational savings can be achieved without compromising safety.

4. Robustness: Even with a non-admissible heuristic and high weights, the algorithm produced paths within a small bound of the optimal safety cost.

5. Practical Applicability: Weighted A* is well-suited for applications where:

- Computational resources are constrained
- Real-time pathfinding is required
- A small sacrifice in path quality is acceptable for significant speed gains
- Graph size is moderate to large (hundreds to thousands of nodes)

References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only

the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] R. Ebendt and R. Drechsler, "Weighted A* search - unifying view and application," Artif. Intell., vol. 173, no. 14, pp. 1310-1342, Sep. 2009.

[2] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.

[3] Pohl, I. (1970). Heuristic search viewed as path finding in a graph. Artificial Intelligence, 1(3-4), 193-204.

[4] Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

[5] Nilsson, N. J. (1980). Principles of Artificial Intelligence. Morgan Kaufmann.

[6] Pearl, J. (1984). Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley.

[7] Richter, S., Thayer, J., & Ruml, W. (2010). The joy of forgetting: Faster anytime search. Proceedings of the 20th International Conference on Automated Planning and Scheduling, 80-87.

[8] Hansen, E. A., & Zhou, R. (2007). Anytime heuristic search. Journal of Artificial Intelligence Research, 28, 267-297.

[9] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

STATEMENT

Hereby, I declare that this paper I have written is my own work, not a reproduction or translation of someone else's paper, and not plagiarized.

Bandung, 1 Juni 2026

Ahmad Rinofaros Muchtar - 13524138