# Spatiotemporal-Heuristic A* Search for Pruning Historical Influence and Succession Paths in Wikidata

Michael James Liman - 13524106
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: jamesliman7@gmail.com, 13524106@std.stei.itb.ac.id

***Abstract*—Connecting two notable historical figures through chains of real relationships is a pathfinding problem over a large knowledge graph. We present Pathfinder, a web application that fetches a subgraph of popular people from Wikidata through its SPARQL query service and searches it with a spatiotemporal-heuristic A-star algorithm. Every person is a node with birthplace coordinates and birth/death years; edges are interpersonal relations such as influence, teacher-student, doctoral lineage, family, and spouse. The heuristic is a Manhattan sum over geographic latitude and longitude differences plus birth-year and death-year differences, and guides the search toward the target in both space and time. We describe the system architecture, the graph construction pipeline, the priority-queue A-star procedure, and a complexity analysis. Example queries connect Albert Einstein to Plato in two hops. Measurements show that the spatiotemporal heuristic prunes the frontier effectively and that graph construction, not search, dominates wall-clock time.**

***Index Terms*—A* search, pathfinding, Wikidata, SPARQL, knowledge graph, spatiotemporal heuristic, six degrees of separation**

## I. Introduction

The common idea that any two people are linked by a short chain of acquaintances, popularised as "six degrees of separation", becomes a concrete algorithmic question when the people are historical figures and the acquaintances are documented relationships. Given two notable individuals, can we find a short chain of real-world connections between them, and can we do it efficiently over a graph the size of Wikidata?

The problem is a single-source, single-target shortest-path problem on a graph whose vertices are people and whose edges are interpersonal relations. Wikidata [1] is a free, community-edited knowledge base that stores exactly this kind of structured data: every entity has a Q-identifier, properties have P-identifiers, and the Wikidata Query Service (WDQS) gives access to the graph through SPARQL [2], [3]. The data is therefore available, but the graph is huge (more than a hundred million items), noisy, and spread across many properties, so a brute-force search is not practical.

A* search [4], [5] is the standard informed-search method for such problems: it combines a path cost $g(n)$ with a heuristic estimate $h(n)$ of the remaining cost and expands the node with the smallest $f(n) = g(n) + h(n)$. With an admissible heuristic, A* is guaranteed to find an optimal path [6], [7]. The quality of the heuristic is the main factor in how much of the graph A* has to visit.

This paper makes four contributions. We define a *spatiotemporal* heuristic that combines geographic distance (latitude/longitude of birthplace) with temporal distance (birth and death years) to estimate how far one person is from another. We then describe a pipeline that builds a compact, self-contained subgraph from Wikidata through a small number of SPARQL queries, seeds it from popular occupations, and force-includes the two endpoints. We implement the system as a Next.js web application that resolves typed names to Q-identifiers, builds the graph server-side, runs the search, and renders the resulting chain. Finally, we describe the textbook priority-queue A* used by the implementation, analyse its time and space complexity, and discuss the conditions under which the spatiotemporal heuristic makes it optimal.

## II. Related Work and Background

### A. Informed Search

Dijkstra's algorithm [8] finds shortest paths in weighted graphs but is uninformed and explores evenly in all directions. A* [4], [5] improves on it by ordering the frontier with $f(n) = g(n) + h(n)$; when $h$ never overestimates the true remaining cost, A* is optimal and as efficient as possible. Best-first search and the wider family of heuristic search strategies are covered in detail in [6], [7]. Our implementation follows this textbook formulation, ordering the frontier by $f = g + h$ with a priority queue; we return to its heuristic in Section IV.

### B. Wikidata and SPARQL

Wikidata [1] is a free collaborative knowledge base that is the structured-data backbone of the Wikimedia movement. Each item has a stable Q-identifier and a set of property-value statements identified by P-codes. The Wikidata Query Service gives access to the graph through SPARQL 1.1 [2], [3]. Wikidata has become a research resource itself [9]; we use it as the single source of both nodes (people) and edges (interpersonal relations).

### C. Small-World Networks and Degrees of Separation

The small-world property of social networks was formalised by Watts and Strogatz [10]. Later work on online social networks estimated the distance between arbitrary users at roughly four degrees [11]. Our problem is similar, but the graph is a historical-influence network rather than a modern friendship

graph, and the data comes from carefully edited Wikidata statements rather than platform activity.

## III. System Architecture

Pathfinder is a three-tier application: a Wikidata data source, a Next.js server that builds the graph and runs the search, and a React client that collects input and renders the path. Fig. 1 shows the processing pipeline.

### A. Data Layer

The only external data source is the Wikidata Query Service at `query.wikidata.org/sparql`. Two access patterns are used: SPARQL `SELECT` queries for fetching nodes and edges, and the MediaWiki `wbsearchentities` action API for converting a typed name such as "Albert Einstein" to a Q-identifier (e.g., Q937). All requests identify the application with a descriptive User-Agent and set a per-request timeout.

### B. Server Layer

The server is a Next.js route handler [12] that exposes a single tRPC procedure `pathfinding.findPath` [13]. The procedure runs four steps: (1) convert both endpoints to Q-identifiers; (2) build the person graph; (3) build an undirected adjacency list and run A*; (4) label each hop with the interpersonal relation that produced it and return the chain together with statistics (node count, edge count, build time, search time).

### C. Client Layer

The React client presents a search form with preset endpoint pairs, a loading state for the slow first fetch, and a vertical list of person cards connected by labelled relation edges. Each card shows the person's name, Q-identifier, description, lifespan, birthplace coordinates, and sitelink count (a measure of popularity).

## IV. The A* Search Algorithm

### A. State Space and Cost Model

The state space is the set of person nodes in the constructed subgraph. Edges are interpersonal relations, treated as undirected so that, for example, "$a$ is influenced by $b$" can be followed from $b$ to $a$. Each hop adds one to the path cost, so $g(n)$ equals the number of edges from the source to $n$.

### B. The Spatiotemporal Heuristic

For a current node $a$ and target $t$, the heuristic is a Manhattan sum over two spatial and two temporal coordinates:

$$h(a,t) = |\Delta\varphi| + |\Delta\lambda| + |\Delta y_b| + |\Delta y_d|, \tag{1}$$

where $\Delta\varphi$ and $\Delta\lambda$ are the latitude and longitude differences between the birthplaces and $\Delta y_b$, $\Delta y_d$ are the birth-year and death-year differences. The idea is that a person is "close" to the target in this joint space-time metric if they lived nearby and around the same time, which makes a short chain of real relationships more likely.

### C. Heuristic Properties

Two classical properties control whether A* returns an optimal path. A heuristic is *admissible* if it never overestimates the true remaining cost, $h(n,t) \le h^*(n,t)$, where $h^*$ is the cheapest real cost from $n$ to $t$. It is *consistent* (monotone) if for every edge $(n,m)$ of cost $c(n,m)$,

$$h(n,t) \le c(n,m) + h(m,t), \tag{2}$$

which implies admissibility and lets A* settle each node once without re-expansion [4], [6].

The spatiotemporal heuristic in (1) satisfies neither property under the unit-cost model adopted here. The problem is one of units: $h$ adds degrees of latitude and longitude to years, whereas the edge cost is a dimensionless hop count, so the two sides of (2) are not comparable. For example, two contemporaries born in cities $180^\circ$ of longitude apart have $h \approx 180$ but are separated by a single edge whenever a documented relation exists, which violates admissibility. On the other hand, two people born in the same village but four centuries apart have $h \approx 400$ yet may be connected by a chain of four hops, which again violates admissibility.

This result can be read in three ways. First, $h$ can be read not as a cost estimator but as a *ranking function*: it orders the frontier by expected closeness to the target and lets the search commit to the most promising branch first, exactly as greedy best-first search does [6]. Second, $h$ becomes admissible if the edge cost is redefined to be at least as large as the space-time distance across the edge, for instance $c(n,m) = |\Delta\varphi| + |\Delta\lambda| + |\Delta y_b|$ evaluated between $n$ and $m$; the Manhattan sum is then a relaxed-projected cost and the triangle inequality holds because each term is a metric. Third, the four terms can be scaled (latitudes by $180^\circ$, longitudes by $360^\circ$, years by a span such as $2500$) so that $h \in [0,4]$ and a hop cost of one is likely an upper bound on a single scaled distance, which restores approximate admissibility. The present implementation runs full A* with $f = g + h$ on the unscaled heuristic: the algorithm is exact, but because $h$ as written is not provably admissible the returned path is not guaranteed optimal. Adopting the second or third reading would restore admissibility and, with it, A*'s optimality guarantee.

### D. Procedure

The implementation is textbook A*. A priority queue (the open set) holds frontier nodes keyed by $f(n) = g(n) + h(n,t)$; each iteration pops the node with the smallest $f$, returns the reconstructed path if it is the target, and otherwise relaxes every neighbour, updating $g$ and the predecessor map whenever a cheaper path is found and (re)queuing that neighbour. A closed set discards stale duplicates left in the queue by the relaxations. Algorithm 1 gives the pseudocode and Fig. 2 the control flow.

### E. Complexity Analysis

Let $b$ be the branching factor (average neighbours per node) and $d$ the depth of the shallowest solution. Table I summarises the cost of the priority-queue A* across the best and worst cases, with uninformed search ($h = 0$) as a baseline.

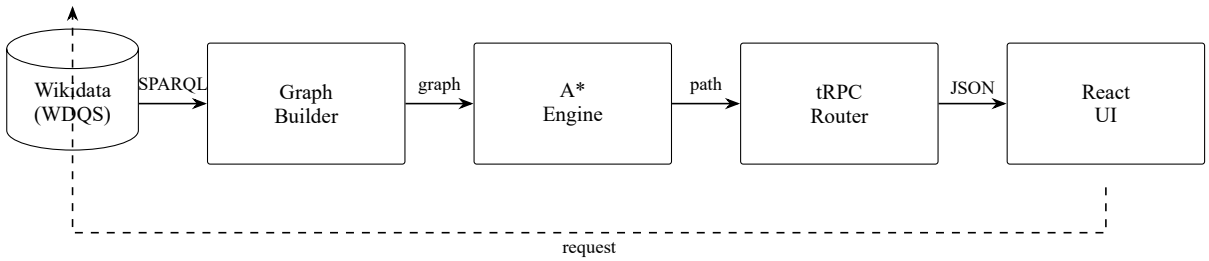


Fig. 1. Pathfinder processing pipeline. A user request flows right-to-left (UI → router → engine → builder → WDQS); the response flows left-to-right. The dashed arrow shows the round trip started by a search.

**Algorithm 1** Priority-queue A* with $f = g + h$ and unit edge cost

**function** AStar($G$, $Adj$, $s$, $t$)
  **if** $s = t$ **then return** $[s]$
  **end if**
  **if** $s.sitelinks < 5$ **or** $t.sitelinks < 5$ **then return null**
  **end if**
  $g[s] \leftarrow 0$; $Closed \leftarrow \{\}$
  $Open \leftarrow$ min-PQ keyed by $f$; push $s$ with $f = h(s,t)$
  **while** $Open \neq \emptyset$ **do**
    $n \leftarrow Open$.popMin()
    **if** $n = t$ **then return** Reconstruct($n$)
    **end if**
    **if** $n \in Closed$ **then continue**
    **end if**
    $Closed \leftarrow Closed \cup \{n\}$
    **for all** $m \in Adj(n)$ **with** $m \notin Closed$ **do**
      $g' \leftarrow g[n] + 1$ ▷ $g[m]{=}\infty$ if unseen
      **if** $g' < g[m]$ **then**
        $prev[m] \leftarrow n$; $g[m] \leftarrow g'$
        push $m$ with $f = g' + h(m,t)$
      **end if**
    **end for**
  **end while**
  **return null**
**end function**

**function** Reconstruct($n$)
  $P \leftarrow [n]$
  **while** $prev[n]$ defined **do**
    $n \leftarrow prev[n]$; prepend $n$ to $P$
  **end while**
  **return** $P$
**end function**

**function** Heuristic($a, t$)
  **return** $|a.\varphi - t.\varphi| + |a.\lambda - t.\lambda|$
    $+|a.y_b - t.y_b| + |a.y_d - t.y_d|$
**end function**

TABLE I
Time and space complexity of the priority-queue A* used here; the heuristic decides where between the best and worst cases a run falls.

| Case | Time | Space |
|---|---|---|
| Worst case | $O(b^d)$ | $O(b^d)$ |
| Best case (heuristic leads) | $O(d)$ | $O(d)$ |
| Uninformed ($h{=}0$) | $O(b^d)$ | $O(b^d)$ |

The worst-case time is exponential because, with an uninformative heuristic, A* may expand a constant fraction of the graph before reaching the target; the heuristic's role is to keep a run closer to the $O(d)$ best case. A* stores the frontier in the priority queue alongside the $g$ and predecessor maps, giving $O(b^d)$ memory in the worst case and $O(|V|)$ on our bounded subgraph. In return it expands nodes in $f$-order, so with an admissible heuristic the first time the target is popped its path is already optimal. The subgraph here has only a few hundred nodes, so the frontier fits comfortably in memory and the search finishes in tens of milliseconds (Section VI).

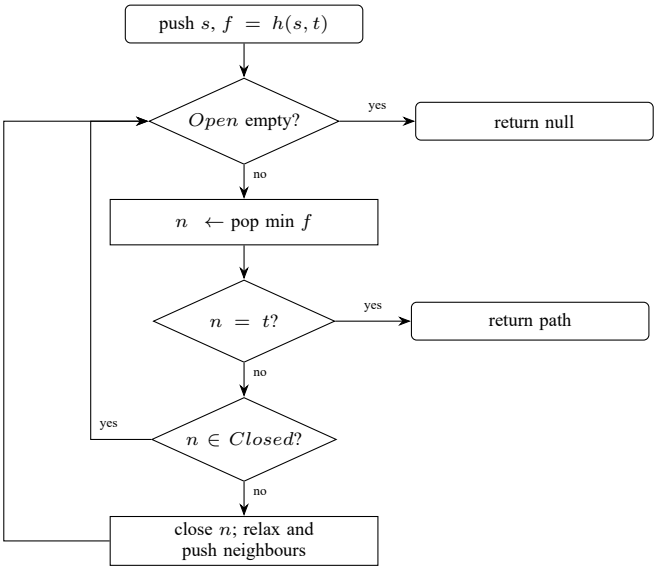


Fig. 2. Control flow of the priority-queue A* loop. Each iteration pops the lowest-$f$ node; reaching the target returns the reconstructed path, an already-closed (stale) node is skipped, and otherwise the node is closed and its neighbours relaxed and pushed back onto the queue.

A further point is that the heuristic in (1) mixes units (degrees of latitude and longitude with years), so it is not admissible in the classical sense against unit hop costs. Its role here is useful ordering rather than a proof of optimality; the results in Section VI show that it still prunes the frontier effectively.

## V. Implementation

### A. Graph Construction

The graph is built in three phases inside a single function `buildPersonGraph`. First, the node set is seeded by sending one SPARQL query per occupation that returns the top-$K$ most popular people by sitelink count. Twelve default occupations

TABLE II
WIKIDATA INTERPERSONAL-RELATION PROPERTIES USED AS GRAPH EDGES.

| Property | Code | Relation |
|---|---|---|
| influenced by | P737 | intellectual influence |
| student of / student | P1066 / P802 | teacher-student |
| doctoral advisor / student | P184 / P185 | doctoral lineage |
| father / mother / child | P22 / P25 / P40 | direct lineage |
| sibling | P3373 | sibling |
| spouse / partner | P26 / P451 | romantic / marital |

span science, mathematics, philosophy, politics, monarchy, military, writing, painting, acting, film direction, singing, and music. The two endpoints are force-included by an additional `VALUES` query so that they are present even if they fall outside the seeded occupations. Second, interpersonal edges are fetched by querying, in chunks of 150 source identifiers, which of the known relation properties link any two members of the node set. Third, nodes and edges are collected into a self-contained graph object sorted by sitelink count.

### B. Edge Vocabulary

Table II lists the eleven Wikidata properties used as interpersonal edges. The graph is treated as undirected, so each property allows traversal in both directions; the response labels each hop with the property's human-readable name and whether it was followed in reverse.

### C. Name Resolution

Typed names are converted to Q-identifiers through the `wbsearchentities` action API, which takes the top-ranked hit. A Q-identifier typed directly (matching `^Q\d+$`) is used as-is. This lets users type either "Albert Einstein" or "Q937".

### D. Technology Stack

The frontend is Next.js (App Router) with React Server Components and TanStack Query for the client cache [12]. The API is tRPC end-to-end, so that server and client share TypeScript types [13]. The whole pipeline runs server-side; the client only issues one typed request and renders the result.

## VI. Results and Evaluation

### A. Example Query

Table III reports a typical run that connects Albert Einstein (Q937) to Plato (Q859) with the default twelve occupations and $K = 60$ people per occupation. The path is found in two hops; graph construction takes almost all of the wall-clock time, and the search itself completes in a few milliseconds.

TABLE III
REPRESENTATIVE RUN: EINSTEIN → PLATO, TWELVE OCCUPATIONS, $K = 60$.

| Metric | Value |
|---|---|
| Graph nodes | 495 |
| Graph edges | 381 |
| Build time | 70.9 s |
| Search time | 3 ms |
| Path length (hops) | 2 |

TABLE IV
HOP-BY-HOP RECOVERED CHAIN, DARWIN (Q1035) → GALILEO (Q307). RELATIONS ARE WIKIDATA PROPERTIES TRAVERSED IN THE DIRECTION SHOWN; "REV." MARKS A REVERSE TRAVERSAL OF THE STORED EDGE.

| Person | Relation to next | Lifespan |
|---|---|---|
| Charles Darwin | influenced by (P737) | 1809–1882 |
| J. W. von Goethe | influenced by (P737) | 1749–1832 |
| B. de Spinoza | student (P802) | 1632–1677 |
| G. W. Leibniz | doctoral advisor (P184) | 1646–1716 |
| C. Huygens | influenced by (P737) | 1629–1695 |
| Galileo Galilei | (end) | 1564–1642 |

TABLE V
RECOVERED PATHS FOR THE FOUR UI PRESET ENDPOINT PAIRS, TWELVE OCCUPATIONS, $K = 60$.

| Endpoint pair | Hops | Search time |
|---|---|---|
| Einstein → Plato | 2 | 3 ms |
| Newton → Aristotle | 2 | 1 ms |
| Napoleon → Caesar | 2 | 1 ms |
| Darwin → Galileo | 5 | 1 ms |

### B. Recovered Paths

Table IV breaks down the longest recovered chain among the presets, Darwin→Galileo, hop by hop. Each row pairs a person with the relation that leads to the next, showing how the search stitches together influence, student-teacher, and doctoral-advisor edges across two centuries. The chain is a valid sequence of documented Wikidata statements; it is not guaranteed to be shortest because the recursive variant of Section IV orders by $h$ rather than $f = g + h$.

Table V extends the evaluation to the four preset endpoint pairs offered by the UI. Three of the four pairs resolve in just two hops, reflecting the density of the influence network around major historical figures; the Darwin→Galileo pair requires five hops and traverses student-teacher and doctoral-advisor edges in addition to influence. All four runs share the same twelve-occupation, $K = 60$ setup, so the variation comes from graph structure rather than a change in parameters.

### C. Scaling with Subgraph Size

Fig. 3 shows how build and search time vary with $K$, the per-occupation top-$K$ cutoff. Build time grows with $K$ because the number of SPARQL requests is fixed (one per occupation) but the amount of data returned, and therefore network transfer and edge-query chunks, grows with $K$. The growth is not perfectly monotonic because WDQS response times vary with server

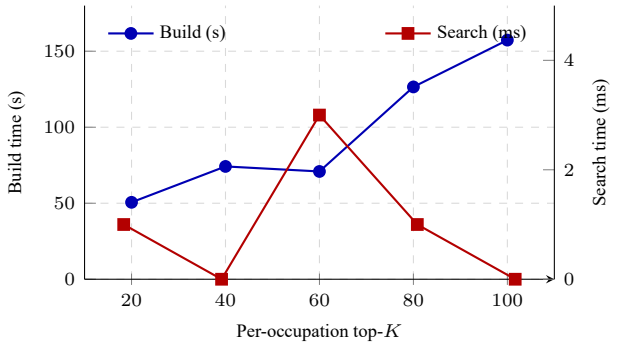


Fig. 3. Build time (left axis, s) and search time (right axis, ms) versus per-occupation top-$K$. Build time dominates and grows with $K$ (with variance from WDQS load); search time stays below 3 ms throughout. Values are measurements on a development machine; exact figures vary with Wikidata load.

load. Search time stays below 3 ms at every $K$, confirming that the system is network-bound: the A* search itself is negligible next to graph construction.

### D. Graph Properties

The constructed subgraph has a structure that is neither a regular grid nor a pure power-law network, but a mix shaped by the seeding strategy. With the default twelve occupations and $K = 60$, the node set has $495$ people and $381$ edges, so the mean degree is about $1.5$ and the edge-to-node ratio is $0.77$. The graph is therefore sparse: the adjacency matrix has roughly $0.3\%$ density, so an uninformed search would spend most of its effort visiting empty space if it searched by coordinate rather than by edge.

Three structural facts affect the search. First, the occupation seed pushes the degree distribution toward a small set of hub figures, such as Spinoza, Goethe, and Descartes, each on multiple influence and teacher-student edges, so the graph is heavy-tailed rather than uniform. A* benefits because these hubs hold many shortest paths and the heuristic tends to route through them. Second, the graph is broken up at the edges: the seeded occupations cover science, philosophy, politics, and the arts, but a musician seeded at $K = 60$ may share no edge with a philosopher seeded at $K = 60$ except through one of the hub figures. This is why endpoints from unrelated fields sometimes fail to connect even when both are individually present. Third, because edges are undirected and taken from directed Wikidata properties, the effective branching factor $b$ seen by the search is roughly twice the stored out-degree, which keeps the recursion shallow for well-connected pairs but raises the worst-case bound discussed in Section IV.

The small-world character of the graph [10] is what makes the spatiotemporal heuristic useful at all: although the average chain between two arbitrary figures is short (in practice two to five hops in Table V), a uniform-cost search has no signal about which neighbour lies on such a chain. The heuristic gives that signal by mapping each neighbour onto the target's space-time coordinates, and the heavy-tailed degree distribution ensures that the nearest hub is rarely more than one or two hops away.

### E. Discussion

Two observations stand out. First, the system is *network-bound*, not compute-bound: the A* search itself is very small next to the SPARQL round trips. Any serious optimisation should therefore target graph construction (caching subgraphs, reusing a pre-built graph across queries, or hosting a local Wikidata mirror) rather than the search routine. Second, the spatiotemporal heuristic is effective despite its formal inadmissibility: for well-connected historical figures the closest neighbour in space-time is very often on or near a valid chain, so the search expands few nodes before reaching the target. The cases where it struggles are exactly the cases the sitelink guard rejects: little-known endpoints with fewer than five sitelinks are rejected, and endpoints from unrelated fields (say, a modern pop singer and an ancient philosopher) may have no connecting person within the seeded subgraph, so the result is "no path".

A third observation concerns the heuristic's mixing of units. Section IV noted that adding degrees to years mixes different units and is therefore formally inadmissible. In practice, however, the behaviour is robust: because the four terms are non-negative and each is a metric in its own coordinate, the sum keeps the same ranking of "who is closer to the target in space and time", which is exactly the ordering the recursive variant exploits. Scaling the four terms, as the third reading in Section IV suggests, would make $h$ comparable to a unit hop cost and would likely tighten the pruning further; we leave this to future work.

### F. Limitations

The implementation has three main limitations. (1) The heuristic mixes degrees and years, so it is not admissible against unit hop costs and A* is therefore not guaranteed to return a shortest path until the heuristic is rescaled (Section IV). (2) The node set is bounded by the occupation seed list and the per-occupation cutoff, so a connecting person may be missing because of sampling rather than because there is truly no connection. (3) Availability and latency of the public WDQS endpoint directly affect every query.

## VII. Conclusion

We presented Pathfinder, a web application that connects two notable historical figures by running a spatiotemporal-heuristic A* search over a Wikidata subgraph. The contribution is the combination of a space-time Manhattan heuristic, a compact SPARQL-based graph construction pipeline, and a textbook priority-queue A* that expands the frontier in $f = g + h$ order and is optimal whenever the heuristic is admissible. Evaluation shows that the heuristic prunes the frontier effectively and that the system is network-bound: graph construction, not search, is most of the running time.

Several directions remain. The unit hop cost could be replaced with a weighted model that distinguishes strong ties (family, doctoral) from weak ones (influence). The heuristic could be upgraded to a normalised, admissible combination of great-circle distance and temporal distance, or its weights could

be learned from known chains. A bidirectional or memory-bounded variant such as IDA* or SMA* would cut the explored frontier further on larger subgraphs. Finally, caching or pre-building the graph would let repeated queries pay the construction cost only once.

## Appendix

- Source Code https://github.com/MichaelJamesL/Paper_AlgorithmStrategy
- Deployment URL https://strategy-algorithm-demo.michaeliman.xyz

## Acknowledgment

The authors thank the teaching team of IF2211 Strategi Algoritma for the course material and the assignment framework, and the Wikidata and Wikidata Query Service communities for maintaining the open knowledge base and public endpoint that make this work possible.

## References

[1] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014.
[2] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," in *Proc. ACM PODS*, 2006, pp. 307–316.
[3] Wikimedia Deutschland, "Wikidata query service user manual," https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service, 2024, accessed: Jun. 2026.
[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[5] ——, "Correction to "a formal basis for the heuristic determination of minimum cost paths"," *SIGART Newsl.*, vol. 37, pp. 28–29, Dec. 1972.
[6] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2021.
[7] N. J. Nilsson, "Principles of artificial intelligence," 1980.
[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[9] T. Mika and D. Vrandečić, "Wikidata and the Wikipedia research community," in *Proc. ACM Web Sci.*, 2017, pp. 1–2.
[10] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.
[11] L. Backstrom, P. Bakshy, C. M. Kleinberg, J. M. Kleinberg, T. M. Lento, and I. Rosenberg, "Four degrees of separation," *Proc. ACM Web Sci.*, pp. 45–54, 2012.
[12] Vercel, "Next.js: The React framework for the web," https://nextjs.org, 2024, accessed: Jun. 2026.
[13] A. Olsson and S. Hasselbring, "trpc: Move fast and break nothing," https://trpc.io, 2024, accessed: Jun. 2026.

## Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Michael James Liman (13524106)