MCP SERVER | NO CODE | CLOUD HOSTED

Shortest Path Engine MCP for AI Agents

Calculating Optimal Routes and Analyzing Weighted Graph Networks

The Shortest Path Engine calculates optimal routes in any weighted graph. It provides specialized algorithms—Dijkstra, A*, and Bellman-Ford—to find the most efficient path and total distance, even when dealing with negative edge weights or complex spatial heuristics.

A+ Quality Score 100/100

dijkstra | astar | bellman-ford | graph-algorithms | shortest-path

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/shortest-path-engine

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Shortest Path Engine MCP

3 tools available
Cloud-hosted on Vinkius

This MCP handles computational tasks for highly connected networks. Whether you're mapping logistics routes, analyzing financial dependencies, or modeling communication flow, this connector calculates the absolute shortest path between any two points in your graph. It doesn't just guess; it runs specialized algorithms designed to handle specific types of network complexity.

Need a quick route calculation for a standard map? Use one method. But if your connections have negative costs or you need to detect cyclical issues, another algorithm is required. This engine manages that complexity automatically, giving you precise paths and total distances every time. By connecting this MCP through the Vinkius catalog, your AI client gains access to robust graph analysis without needing specialized computational libraries.

Core Capabilities

01 — Find optimal paths using Dijkstra's algorithm

Calculates the shortest route for graphs where all connection weights are zero or positive.

02 — Perform optimized search with A* heuristics

Determines the fastest path in spatial networks by factoring in directional estimates, like Manhattan or Euclidean distances.

03 — Scan graph for negative cycles and paths

Analyzes graphs containing negative edge weights to find the true shortest path or detect infinite cost loops.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/shortest-path-engine** — connect your AI agent in three steps.

01 You define your network by providing a set of nodes (points) and weighted edges (connections).

02 Your AI client specifies the required search parameters, such as whether the graph has negative weights or if spatial heuristics should be used.

03 The MCP executes the appropriate algorithm—Dijkstra, A*, or Bellman-Ford—and returns the precise sequence of nodes that form the shortest path and its total calculated distance.

The bottom line is you send a network map and a start/end point; you get back the guaranteed most efficient route based on your graph's specific rules.

Built For

This MCP is essential for Data Scientists, Logistics Planners, and Network Architects. If your job involves determining connectivity or optimizing movement across complex systems—like supply chains or electrical grids—you need this tool.

Logistics Planner

Determines the most fuel-efficient routes for delivery trucks, accounting for variable road costs and traffic weights.

Network Analyst

Checks connectivity between servers or routers, identifying potential bottlenecks or paths that might suffer from negative load impact.

Data Scientist

Tests complex mathematical models by finding the true shortest distance in a simulated graph containing varied weight types.

What Changes When You Connect

01 Eliminate guesswork when planning routes. Use `dijkstra` to find the most efficient path in networks where all costs (like distance or fuel) are positive.

02 `astar` provides optimized searches for geographical problems, factoring in heuristics like Manhattan distances to narrow down possible paths quickly.

03 Safely analyze graphs with tricky negative weights. The `bellman_ford` tool detects infinite cost reduction loops and calculates accurate paths where other methods fail.

04 Stop manually comparing algorithms. This single MCP gives you three industry-standard pathfinding tools, letting your agent select the right one instantly.

05 Get more than just a route; you receive total distances and full reachability maps for any weighted directed or undirected graph.

Real-World Applications

Optimizing city-wide delivery routes

A logistics manager needs to find the fastest sequence of stops in a dense urban area. By using `astar` with a Manhattan heuristic, the agent calculates the optimal path between hundreds of nodes far faster than manual routing software.

Modeling electrical grid failure points

A utility engineer must find the most resilient connection path across an aging grid where certain lines have negative operational costs. Using `bellman_ford` identifies the true shortest route, ignoring dangerous cost cycles.

Simulating financial dependency chains

A quantitative analyst models how a decline in one asset affects others. Since some dependencies might represent 'cost reduction' (negative weights), they use `bellman_ford` to ensure the pathfinding is mathematically sound.

Finding optimal data flow paths

A system architect needs to map out data transfer routes between microservices. Since all connections have a positive latency cost, using `dijkstra` guarantees the minimal data transit time.

Patterns to Avoid

Using Dijkstra’s for negative weights

✗ AVOID

Attempting to find the shortest route in an electrical grid model where removing a connection actually reduces total cost. Dijkstra will return a misleadingly short path.

✓ INSTEAD

You must use `bellman_ford`. This tool is designed specifically to handle negative edge weights and detect if any infinite-loop cycles exist before calculating the true minimum distance.

Overlooking spatial heuristics

✗ AVOID

Calculating a route across a city grid but ignoring that straight lines are impossible, leading to an overly optimistic path estimate.

✓ INSTEAD

Use `astar`. By incorporating heuristics like Euclidean or Manhattan distances, you constrain the search space and find a much more realistic optimal path.

The Right Fit

You need this MCP if your core problem involves finding the minimum cost path in a network. If every connection weight is positive (like time or distance), use `dijkstra`. When you are dealing with physical space and directional estimates, switch to `astar`. However, if your graph includes any negative costs—meaning moving along an edge *reduces* the total accumulated value—you must use `bellman_ford` first. Never rely on Dijkstra's or A* when negative weights are possible; you risk missing a critical cycle detection or calculating the wrong path entirely.

Shortest Path Engine MCP for Graph Network Analysis

Manually mapping out complex connections is brutal. You're constantly toggling between algorithms, checking if negative weights exist, and then running different calculations just to see which path is 'optimal.' It’s a tedious cycle of selecting the right tool for the graph type.

With this MCP, you send your network once. Your agent handles the complexity. Whether it's calculating paths in spatial coordinates or identifying tricky negative cost cycles, you get one single, accurate result describing the precise path and total distance.

Shortest Path Engine MCP for Computational Logistics Routing

The old way required breaking down a massive logistics challenge into dozens of smaller, isolated routing problems. You’d run one calculation for the positive-cost segments, then another check just to see if negative costs (like rebates or subsidies) invalidated your whole plan.

Now, you feed the entire system—all positive costs and all potential cost reductions—into this MCP. It calculates the final, single optimal route in one step, giving immediate, reliable answers for global optimization.

Shortest Path Engine: 3 Algorithms for Weighted Graph Analysis

Use these specialized functions to calculate minimum distances, find optimized paths, and detect negative cycles across any complex network map.

#	TOOL	DESCRIPTION
01	`astar`	Finds the shortest path using A* search, ideal for graphs with spatial heuristics like Manhattan or Euclidean distances.
02	`bellman_ford`	Scans a graph using the Bellman-Ford algorithm, which is necessary when negative edge weights are present or if you need to detect negative cycles.
03	`dijkstra`	Finds the shortest path using Dijkstra's algorithm; it must only be used for graphs with non-negative connection weights.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Find the best path from HQ to Warehouse C using Dijkstra with these connections: [{'start': 'HQ', 'end': 'A', 'cost': 5}, {'start': 'A', 'end': 'B', 'cost': 3}, {'start': 'HQ', 'end': 'C', 'cost': 10}]

The optimal path from HQ to C is [HQ, A, B, C] with a total cost of **8**.

- **Direct Path:** 10
- **Optimal Path (via A and B):** 5 + 3 = 8. This confirms the most efficient route.

U What is the optimal path from Start to End in this grid using A* search with Manhattan distance?

Optimal Path Found: [Start, Mid, East, End]

NODE	DISTANCE ESTIMATE
Start	0
Mid	2
East	4
End	6

The total distance is **8**. The A* heuristic guided the search to bypass unnecessary nodes.

U Check for any negative cycles in a graph starting from node S with these edges: [{'start': 'S', 'end': 'A', 'cost': 1}, {'start': 'A', 'end': 'B', 'cost': -2}, {'start': 'B', 'end': 'S', 'cost': 1}]

The Bellman-Ford analysis completed.

✅ **Negative Cycle Detected.** The path S → A → B → S has a total weight of 0, but if weights were lower (e.g., -3), it would signal an infinite cost reduction loop.

In this specific case, the minimum cost is confirmed as 1 via the direct path.

Frequently Asked Questions

01 **How do I use the Shortest Path Engine MCP for complex logistics planning?**

You feed the engine all your nodes and weighted connections, including potential cost reductions. It handles the complexity automatically, giving you a single, mathematically proven optimal route that accounts for every variable in your network.

02 **Is Shortest Path Engine better than standard mapping tools?**

Yes, because it's algorithmic, not just geographical. Standard maps struggle with negative costs or highly specialized graph structures; this MCP calculates the true minimum path based on mathematical rules.

03 **When should I use A* versus Dijkstra in Shortest Path Engine?**

Use `astar` when your network is geographically defined and you have a good estimate of how far away the destination is (a heuristic). Use `dijkstra` if you just need basic pathfinding where all costs are positive.

04 **What happens if my graph has negative edge weights?**

You must use the Bellman-Ford algorithm. It's the only tool in this MCP that correctly processes connections with negative weights and, crucially, alerts you if an impossible infinite loop exists.

05 **Does Shortest Path Engine require me to know complex graph theory?**

No. You just tell your agent what your network looks like (nodes/edges), and the MCP handles selecting and running the correct, specialized mathematical algorithm for you.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"shortest-path-engine": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Shortest Path Engine is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Shortest Path Engine. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	July 2026
MCP Server	Shortest Path Engine MCP
Server ID	`019f2d2e-dc3c-7339-b32a-d1ee3e80eb7d`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/shortest-path-engine.