BEYOND SHORTEST-PATH FLOW FIELDS: CERTIFIED CYCLE-EXCHANGE K-FLOW FIELDS FOR REUSABLE ROUTE ALTERNATIVES

Anonymous authors
Paper under double-blind review

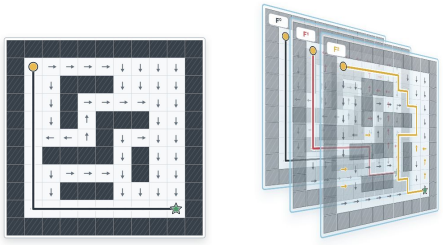
ABSTRACT

Flow fields give agents sharing a destination constant-time action lookup, but conventional construction confines every route to shortest-path descent. Even when another corridor exists, agents cannot use it if all shortest routes converge at a bottleneck. We study complete destination-reaching flow fields as reusable policy families and prove their one-to-one correspondence with spanning trees, making every simple source-to-destination route realizable. We introduce Certified Cycle-Exchange K-Flow Fields (CCE-KFF), which compiles these families by exchanging tree edges. Each exchange preserves reachability from every source and identifies exactly which induced routes change, while the exchanges connect the complete field space. We also characterize the minimum number of exchanges between fields. Length- and topology-aware selectors produce near-shortest routes or alternatives through distinct corridors. Each compiled field remains executable from every reachable source. Experiments on Moving AI and StarCraft maps show stronger topological separation than source-specific K-shortest planners at near-shortest length. On Moving AI with 20 sources, CCE-KFF is over $850\times$ faster than Yen while retaining constant-time action lookup.

1 INTRODUCTION

Pathfinding is a fundamental capability in autonomous robots, unmanned vehicles, transportation systems, and interactive games, where an agent must move safely and efficiently from its current location to a destination (Karur et al., 2021; Harabor & Grastien, 2011). Many game and simulation workloads involve large populations that pursue the same strategic target, such as a base, resource, exit, or rally point. Their starting locations may change continuously even when the destination and environment remain fixed. Repeating an independent search for every new source makes planning cost grow with the population, whereas movement control requires an action to be available immediately at every step.

Classical graph-search methods such as Dijkstra's algorithm (Dijkstra, 1959), A* (Hart et al., 1968), and Jump Point Search (Harabor & Grastien, 2011) efficiently compute the shortest routes between specified endpoints. Work on K-shortest paths by Yen (1971) and Eppstein (1998), together with alternative-routing methods (Abraham et al., 2010), provides multiple route choices but constructs them for a fixed source–destination pair. Flow-field (FF) methods take a different approach. As described by Emerson (2013) and extended by subsequent work (Yang et al., 2025), they perform destination-side preprocessing so that a complete direction table can guide agents from every reachable location through a constant-time next-action lookup. Conventional FF construction selects distance-descending actions, supplying only shortest-route policies. These lines of work therefore separate



(a) Standard FF (b) K-FF family

Figure 1: A standard flow field stores one reusable policy; a K-flow-field family stores multiple complete policies for the same destination.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reusable guidance from source-specific route diversity. Our goal is to provide both through the same lookup interface.

To close this gap, we introduce *Certified Cycle-Exchange K-Flow Fields* (CCE-KFF), which extends the conventional one-policy flow-field interface to a destination-indexed family of complete flow fields, as illustrated in Figure 1. The compiler begins with shortest-path fields and applies certified exchanges that replace one navigation connection while preserving reachability from every location; each exchange also identifies precisely which starting region receives a different route. Deterministic selectors can prioritize low destination-wide route stretch, greater topological separation, or broader source-region coverage. We prove that certified exchanges connect the complete deterministic flow-field space, characterize the exact number of exchanges between any two fields, and show that every simple route can be realized after finitely many exchanges. Our experiments compare CCE-KFF with source-specific planners and destination-side controls on Moving AI and StarCraft. At 20 sources, CCE-KFF is $850.65\times$ faster than Yen, $2.93\times$ faster than Postponed Node Classification (PNC), and $2.63\times$ faster than Plateaus, while its length-aware mode attains mean stretch 1.0003 on Moving AI and 1.0000 on StarCraft. Its topology-aware mode further expands directed-edge support and reduces normalized peak pressure by 25% relative to Standard FF.

Specifically, we make the following contributions:

(1) We formulate K-flow-field compilation as a destination-indexed multi-route pathfinding problem and introduce CCE-KFF, which compiles a requested family of complete and source-independent flow fields. The resulting policies can be reused by agents from arbitrary starting locations through constant-time next-action lookup, while deterministic selectors allow the compiler to prioritize route length, topological separation, or source-region coverage.

(2) We establish a one-to-one correspondence between complete deterministic flow fields and destination-rooted spanning trees. Building on this, we prove that certified exchanges preserve global destination reachability, identify which starting region receives changed routes, connect the complete flow-field space with an exact exchange distance, and realize every simple source-to-destination route after finitely many exchanges.

(3) We evaluate CCE-KFF on 33 Moving AI and 11 StarCraft maps against published multi-route planners and controlled field-family alternatives. CCE-KFF retains near-shortest route quality, produces strongly separated alternatives, and accelerates shared-destination planning by up to $850.65\times$ while reducing static route concentration.

2 Related Work

Flow-field navigation. Potential- and field-based navigation amortizes planning across many agents by representing goal-directed motion over an environment. Continuum potentials and navigation fields have been used for pedestrian and crowd motion (Hughes, 2002; Treuille et al., 2006; Patil et al., 2011). For grid-based games, Emerson (2013) described the standard cost-, integration-, and direction-field pipeline, while Yang et al. (2025) accelerated field construction through an integration-guided wavefront. These methods provide destination-wide reuse and constant-time action lookup but ordinarily encode one distance-descending policy. CCE-KFF preserves the same execution interface while compiling a family of complete fields that can induce both shortest and non-shortest routes.

K-shortest path enumeration. K-shortest-path methods rank multiple routes for a fixed source–destination pair. Representative algorithms include deviation-based loopless enumeration, general partitioning, algorithms specialized to undirected simple paths, output-sensitive unrestricted enumeration, and replacement-path acceleration (Yen, 1971; Lawler, 1972; Katoh et al., 1982; Eppstein, 1998; Hershberger et al., 2007). More recently, Zoobi et al. (2023) introduced Postponed Node Classification (PNC), a practical K-shortest simple-path algorithm with a favorable time–space trade-off. These methods provide strong source-specific length references, whereas CCE-KFF compiles once for a destination and reuses every resulting field from all reachable sources.

Diversity-aware alternative routing. Alternative-routing methods seek structurally separated routes rather than consecutive paths that differ only locally. Prior work includes disjoint paths,

penalty-based generation, locally optimal alternative-route graphs, and explicit length–overlap constraints (Suurballe, 1974; Akgün et al., 2000; Abraham et al., 2010; Bader et al., 2011). The edge-similarity algorithm ESX retrieves short paths under pairwise-overlap constraints (Chondrogiannis et al., 2020), while Hanaka et al. (2022) optimized aggregate diversity over shortest paths. Closest to our application setting, Li et al. (2023) developed the compressed via-path database (CVPD) for efficient alternative-path queries on game maps. These methods return route sets for fixed endpoints; CCE-KFF instead exposes a controllable length–diversity trade-off through complete destination-indexed policies reusable from every reachable source.

3 PRELIMINARIES

This section formulates shared-goal navigation, reviews conventional flow fields, and summarizes the spanning-tree exchanges used by our method.

3.1 PROBLEM DEFINITION

Let the traversable environment be a finite, connected, undirected unit-cost graph $G = (V, E)$, with $n = |V|$ and $m = |E|$, using standard graph-theoretic terminology (Bondy & Murty, 2008). For source s and destination g, an s-to-g route $P = (v_0, \dots, v_\ell)$ satisfies $v_0 = s$, $v_\ell = g$, and $\{v_i, v_{i+1}\} \in E$. It is simple if no vertex repeats, and its length is $L(P) = \ell$. The shortest-path distance is $d_G(s, g) = \min_{P: s \rightsquigarrow g} L(P)$.

A deterministic flow field for destination g is a complete next-action function

$$F : V \to V, \qquad F(g) = g, \qquad \{v, F(v)\} \in E \quad \text{for every } v \neq g.$$

Starting from $x^0 = s$, execution follows $x^{t+1} = F(x^t)$. A field is *destination-reaching* if every source reaches g in finitely many steps; then $L_F(s) = \min\{t \geq 0 : x^t = g\}$ and $P_F(s) = (x^0, x^1, \dots, x^{L_F(s)})$. Let $\mathcal{F}(G, g)$ denote all such fields. We study shared-goal workloads in which G and g are fixed while sources vary.

Problem 3.1 (K-flow-field compilation)**.** Given G, g, and a positive field budget K, compile a family $\mathcal{K}_g$ satisfying

$$\mathcal{K}_g \subseteq \mathcal{F}(G, g), \qquad |\mathcal{K}_g| = \min\{K, |\mathcal{F}(G, g)|\}. \tag{1}$$

The fields must be distinct, and $\mathcal{K}_g$ must contain a field $F^{(0)}$ satisfying $L_{F^{(0)}}(s) = d_G(s, g)$ for every $s \in V$. Each field is a complete next-action table reusable from every source with $O(1)$ lookup. The exact-K condition in Equation (1) counts distinct complete fields, which may still induce the same route for a particular source.

3.2 FLOW-FIELD NAVIGATION

Conventional flow fields use destination-side preprocessing to construct a reusable direction table (Emerson, 2013; Yang et al., 2025); related arrival-time potentials underlie continuum navigation methods (Sethian, 1996; Treuille et al., 2006). Under the unit-cost model, write $d(v) = d_G(v, g)$. Shortest-path distances computed by Dijkstra's algorithm (Dijkstra, 1959) define the field set

$$\mathcal{S}(G, g) = \{F \in \mathcal{F}(G, g) : d(F(v)) = d(v) - 1 \text{ for every } v \neq g\}.$$

Every lookup decreases d by one, so $L_F(s) = d_G(s, g)$ for every $s \in V$ and $F \in \mathcal{S}(G, g)$. Thus tie breaking may change the field but not route optimality. Conventional fields retain $O(1)$ lookup but restrict routes to the shortest-path directed acyclic graph (DAG); $\mathcal{F}(G, g)$ retains the same interface without this restriction.

3.3 SPANNING TREES AND EDGE EXCHANGES

For a spanning tree $T = (V, E_T)$ rooted at g, let $Q_T(u, v)$ be its unique u-to-v path. For any non-tree edge $e = \{a, b\} \in E \setminus E_T$, adding e creates the fundamental cycle $C_T(e) = Q_T(a, b) \cup \{e\}$. Replacing any tree edge $f \in C_T(e) \cap E_T$ by e yields another spanning tree, the standard edge-exchange operation, or basis exchange in the graphic matroid (Bondy & Murty, 2008; Oxley, 2011; Schrijver, 2003). Section 4 uses this operation to construct destination-reaching flow fields.

4 METHODOLOGY

In this section, we present CCE-KFF, our method for solving the K-flow-field compilation problem in Problem 3.1. We first establish a structural representation of complete flow fields, then introduce a certified operation for traversing the resulting policy space, and finally describe how CCE-KFF compiles a reusable family of destination-reaching fields.

4.1 FLOW FIELDS AS ROOTED SPANNING TREES

For a destination-reaching field $F \in \mathcal{F}(G, g)$, define its undirected support as $E_F = \big\{\{v, F(v)\} : v \in V \setminus \{g\}\big\}$. Discarding the directions of these support edges gives a compact structural representation of the field. The following result establishes that this representation is exact.

Theorem 4.1 (Flow-field–tree correspondence)**.** *There is a bijection between the deterministic destination-reaching flow fields in $\mathcal{F}(G, g)$ and the spanning trees of G. A field F maps to $T_F = (V, E_F)$, and a spanning tree maps to the unique field obtained by orienting it toward g. Consequently, by the Matrix–Tree Theorem (Kirchhoff, 1847; Chaiken, 1982), $|\mathcal{F}(G, g)| = \tau(G) = \det L_G^{(g)}$, where $\tau(G)$ is the number of spanning trees of G, and $L_G^{(g)}$ is the reduced Laplacian matrix obtained by deleting the row and column indexed by g.*

Proof sketch. Every vertex reaches g under F, so the support is connected. Its $n-1$ non-destination vertices contribute $n-1$ distinct edges: a shared undirected edge would require two vertices to point to each other, contradicting reachability. Hence the support is a spanning tree. Conversely, orienting any spanning tree toward g yields a unique destination-reaching field. □

By Theorem 4.1, we write $\mathrm{orient}_g(E_T)$ for the unique destination-reaching field obtained by orienting a spanning tree $T = (V, E_T)$ toward g. The size of the deterministic flow-field space is therefore independent of the chosen destination. The correspondence also characterizes the routes representable by a complete direction field. Let $P = (v_0 = s, v_1, \ldots, v_\ell = g)$ be any simple route. The edges of P form a forest and can be extended to a spanning tree T_P of the connected graph G, a standard graphic-matroid fact (Oxley, 2011; Schrijver, 2003). Orienting T_P toward g produces a field F_P satisfying

$$P_{F_P}(s) = P, \qquad L_{F_P}(s) = \ell. \tag{2}$$

Thus complete flow fields can realize any simple source-to-destination route, including routes outside the shortest-path DAG. CCE-KFF uses this tree representation to construct diverse fields with the same constant-time lookup interface.

4.2 CERTIFIED CYCLE EXCHANGE

The correspondence in Theorem 4.1 allows us to modify a complete flow field through a local transformation of its tree certificate. Let $F \in \mathcal{F}(G, g)$ have certificate $T_F = (V, E_F)$, and choose a non-tree edge $e \in E \setminus E_F$. As reviewed in Section 3.3, adding e creates the fundamental cycle $C_{T_F}(e)$. For any tree edge $f \in C_{T_F}(e) \setminus \{e\}$, define

$$E_{F'} = (E_F \cup \{e\}) \setminus \{f\}, \qquad F' = \mathrm{orient}_g(E_{F'}).$$

We call (e, f) a *certified cycle exchange*. The fundamental cycle certifies that removing f after inserting e preserves the spanning-tree structure. Removing f from T_F divides the tree into two components. Let $A(F; e, f)$ denote the component not containing g, and let $B(F; e, f) = V \setminus A(F; e, f)$ denote the component containing g. Because f lies on $C_{T_F}(e)$, the endpoints of e lie in different components; write $e = \{a, b\}$, where $a \in A(F; e, f)$ and $b \in B(F; e, f)$. Figure 2 illustrates this construction and the affected component $A(F; e, f)$.

Theorem 4.2 (Certified exchange and exact affected region)**.** *Every certified cycle exchange produces a field $F' \in \mathcal{F}(G, g)$. The induced route changes exactly for the sources in $A(F; e, f)$: if $s \notin A(F; e, f)$, then $P_{F'}(s) = P_F(s)$; otherwise, the new route crosses e, and its length is*

$$L_{F'}(s) = \begin{cases} L_F(s), & s \notin A(F; e, f), \\ \mathrm{dist}_{T_F[A]}(s, a) + 1 + L_F(b), & s \in A(F; e, f). \end{cases} \tag{3}$$

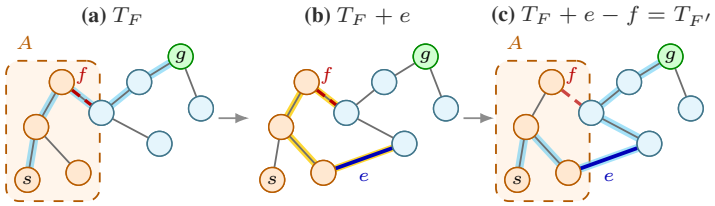


Figure 2: Certified cycle exchange. Replacing f with e on the fundamental cycle yields $T_{F'}$; exactly the routes from A change.

Proof sketch. Adding e creates exactly one cycle, and removing f from that cycle restores a connected graph with $n-1$ edges. The result is a spanning tree and therefore defines a destination-reaching field by Theorem 4.1. A source in $B(F;e,f)$ retains its original route because that route does not use f. A source in $A(F;e,f)$ must instead follow the preserved path to a, cross $e=\{a,b\}$, and then follow the unchanged route from b to g, which gives Equation (3). □

The certificate $(e, f, A(F;e,f), C_{T_F}(e))$ therefore specifies both the new field and its complete source-wise effect. The old and new routes of every affected source differ exactly on the fundamental cycle, so the aggregate edge-level route change is

$$I(F;e,f) = \sum_{s\in V} |E(P_F(s)) \triangle E(P_{F'}(s))|\,,$$
$$I(F;e,f) = |A(F;e,f)|\,|C_{T_F}(e)|. \tag{4}$$

Given the old length table, one traversal of the affected subtree rooted at a evaluates Equation (3) for every affected source in $O(|A(F;e,f)|)$ time. The same certificate can therefore evaluate the length, topology, and coverage effects of a candidate exchange without independently tracing a new route from every source. To characterize the global reach of this local operation, define the CCE policy graph $\mathscr{X}(G,g) = \big(\mathcal{F}(G,g), \mathcal{E}_{\mathrm{CCE}}\big)$, where two fields are adjacent exactly when one is obtained from the other by one certified cycle exchange. Under Theorem 4.1, this is the classical tree graph, equivalently the graphic-matroid basis graph (Holzmann & Harary, 1972; Maurer, 1973).

Theorem 4.3 (Connectivity and exact exchange distance)**.** *For any $F, F^\star \in \mathcal{F}(G,g)$, there exists a sequence of certified cycle exchanges transforming F into $F^\star$, and every intermediate field belongs to $\mathcal{F}(G,g)$. Their exact distance in the CCE policy graph is*

$$\operatorname{dist}_{\mathscr{X}}(F,F^\star) = |E_{F^\star} \setminus E_F| = \frac{1}{2}|E_F \triangle E_{F^\star}|. \tag{5}$$

Consequently,

$$\operatorname{diam}\mathscr{X}(G,g) \le \min\{n-1, m-n+1\}. \tag{6}$$

Proof sketch. If $F \neq F^\star$, choose $e \in E_{F^\star} \setminus E_F$. The fundamental cycle $C_{T_F}(e)$ contains an edge $f \in E_F \setminus E_{F^\star}$; otherwise, that entire cycle would be contained in the tree $T_{F^\star}$. Exchanging e for such an f adds one target edge without removing any edge already shared with $E_{F^\star}$. Repetition reaches $F^\star$ after exactly $|E_{F^\star} \setminus E_F|$ exchanges, while no shorter sequence can insert all missing target edges. The distance is bounded both by the $n-1$ edges in $E_{F^\star}$ and by the $m-n+1$ edges outside E_F, yielding Equation (6). □

Together with Equation (2), Theorem 4.3 shows that a field containing any simple source-to-destination route is reachable from an initial shortest-path field through finitely many certified exchanges. This connected exchange space supplies the candidate domain used by CCE-KFF.

4.3 K-Flow-Field Compilation

The depth-bounded compiler, including the $D=1$ case, is given in Algorithm 1 of Section A.6. CCE-KFF first constructs a deterministic shortest-path field $F^{(0)} \in \mathcal{S}(G,g)$. It then generates a scheduled stream of shortest-path skeletons through deterministic, field-indexed tie breaking. For each skeleton, the compiler selects one certified cycle exchange and materializes the resulting tree

as a complete next-action table. Duplicate fields do not consume the output budget. If the scheduled stream produces fewer than K distinct fields, a deterministic traversal of the connected CCE policy graph continues until the budget is reached or the field space is exhausted.

For each scheduled skeleton, a deterministic selector ranks a bounded candidate set of size $B = O(1)$ and chooses the exchange to materialize. For a candidate F' generated from F by (e, f), CCE-KFF supports

$$S_{\text{len}}(F') = \frac{1}{n-1} \sum_{s \in V \setminus \{g\}} \frac{L_{F'}(s)}{d_G(s, g)},$$

$$S_{\text{div}}(F'; F, e, f) = |A(F; e, f)|\,|C_{T_F}(e)|,$$

$$S_{\text{cov}}(F'; F, e, f) = |A(F; e, f)|.$$

The length-aware selector minimizes S_{len}, whereas the topology-aware and coverage-aware selectors maximize S_{div} and S_{cov}, respectively. The length certificate in Equation (3) evaluates the affected region, while Equation (4) gives the topology-aware score directly. Fixed lexicographic tie breaking makes the output stream deterministic, so different field budgets return nested prefixes of the same stream. The field–tree correspondence, exchange validity, and deterministic completion traversal ensure that the compiled family contains

$$|\mathcal{K}_g| = \min\{K, |\mathcal{F}(G, g)|\} \tag{7}$$

distinct fields in $\mathcal{F}(G, g)$, including the shortest-path seed $F^{(0)}$. Let $K_{\text{out}} = |\mathcal{K}_g| \leq K$. On four-neighbor grids, constructing the shortest field and indices takes $O(n \log n)$ time. With candidate cap B, the scheduled phase costs

$$T_{\text{scheduled}} = O(n \log n + BnK_{\text{out}}), \qquad S_{\text{scheduled}} = O(n \log n + nK_{\text{out}}).$$

For fixed B, these bounds reduce to $O(n \log n + nK_{\text{out}})$ time and space. When the requested budget is attained, $K_{\text{out}} = K$; storing the complete next-action tables uses $\Theta(nK)$ space.

After compilation, each field is destination-indexed rather than source-specific. An agent at vertex v assigned to field $F^{(i)}$ retrieves its next action as $F^{(i)}(v)$ in $O(1)$ time, allowing the family to serve arbitrary sources without another global planning query.

5 Experiments

We compare CCE-KFF with reusable flow-field families and source-specific multi-route planners on Moving AI and GPPC StarCraft. The main experiments evaluate route quality and shared-destination planning time, a qualitative example visualizes eight route families, and ablations isolate exchange selection and recursive depth.

5.1 Experimental Setup

Baselines. We compare published single-field and source-specific routing methods. **Standard FF** compiles one reusable destination-wide shortest-path policy (Emerson, 2013; Yang et al., 2025). Per source–destination pair, **Yen** enumerates loopless paths by length (Yen, 1971), **PNC** accelerates exact K-shortest-simple-path search (Zoobi et al., 2023), and **Plateaus** seeks spatially distinct alternatives through plateau regions (Li et al., 2023). The other three are internal controls introduced here. **Shortest-Only FF** changes only equal-cost parent ties, keeping every field in the shortest-path DAG and excluding non-shortest routes; CCE-KFF-Length can leave this DAG through a certified exchange. **Random-Tree FF** samples seeded depth-first spanning trees to expose diversity without optimizing length. **Repeated-Penalty FF** adapts iterative edge penalization used in alternative-route generation (Akgün et al., 2000; Li et al., 2023): it raises used-edge costs and rebuilds a weighted shortest field for every output, with $\alpha \in \{0.05, 0.5\}$. We report **CCE-KFF-Length** for near-shortest families and **CCE-KFF-Topology** for obstacle-separated alternatives.

Datasets. Moving AI contributes 33 four-connected maps and published scenarios (Sturtevant, 2012); the 2014 Grid-Based Path Planning Competition (GPPC) contributes 11 StarCraft maps (Sturtevant et al., 2015). We fix three destinations per map and 20 distance-stratified reachable sources per destination, giving 99 compilations and 1,980 source instances on Moving AI and

33 compilations and 660 source instances on StarCraft. A separate frozen Moving AI workload contains 330 destinations and 32,630 source instances for the shared-destination assignment diagnostic. The depth study uses the `Boston_0_256` destination in Figure 4 and 100 distance-stratified sources; Appendix B lists all manifests.

Metrics. For a compiled family $\mathcal{K}_g = \{F_i\}_{i=1}^{q}$, **Routes** is the number of distinct routes induced from a source. **Stretch** is $|P|/d_G(s,g)$; we report its mean, 95th percentile, and maximum, with lower values indicating shorter routes. **Edge-Jaccard distance** measures topological separation (Levandowsky & Winter, 1971):

$$D_J(s) = \frac{2}{q(q-1)} \sum_{1 \le i < j \le q} \left(1 - \frac{|E(P_{F_i}(s)) \cap E(P_{F_j}(s))|}{|E(P_{F_i}(s)) \cup E(P_{F_j}(s))|}\right),$$

where $E(P)$ is the undirected edge set of P, and $D_J(s) = 0$ for $q < 2$. Route count removes duplicates; field-family D_J retains multiplicity, while source-specific D_J uses cardinality-matched distinct routes. **DAG escape** is the fraction of induced routes containing an edge outside the shortest-path DAG. **Runtime** is process CPU time. For a shared destination, CCE-KFF and the field baselines are compiled once, while source-specific methods accumulate query time over M sources. **Speedup** is paired reference time divided by method time. We first average over sources and destinations within each map and then macro-average across maps. Static spreading reports **used directed edges** and **peak route-length-normalized edge load**.

Implementation. All methods use the same unit-cost four-neighbor graph in Python 3.12.4. CCE-KFF uses $D = 1$ and $K \in \{1, 4, 8, 16, 32\}$. Length selection minimizes mean normalized route length, topology selection ranks affected-source mass times fundamental-cycle size, and coverage selection ranks affected-source mass. Duplicate fields do not consume the budget. Repeated-Penalty FF uses edge weight $1 + \alpha u_e$, where u_e is prior field usage; Random-Tree FF uses seed 20260917; and Plateaus uses $\epsilon = 1.5$. PNC and Plateaus are reimplemented from their published algorithms in the same Python framework; Yen uses NetworkX 3.5 (Hagberg et al., 2008). Source-specific methods are cardinality-matched to CCE-KFF. PNC and Yen use per-source limits of 300 and 1,200 seconds, respectively. Timing uses $K = 32$ and shared source prefixes for $M \in \{1, \ldots, 20\}$.

5.2 Route Quality Evaluation

Shared-destination systems need short routes with distinct choices for traffic distribution, blockage recovery, and less predictable movement. The metrics form a hierarchy: Routes measures supply; mean, 95th-percentile, and maximum stretch measure travel cost and tail quality; D_J measures edge separation among usable routes; and DAG escape identifies structures unavailable to shortest paths. Used edges and peak normalized load then measure traffic spreading.

Shortest-Only FF, our tie-breaking control, produces many distinct routes and nonzero D_J, all within the shortest-path DAG. If that DAG funnels through one corridor or bottleneck, increasing K cannot create a non-shortest bypass. DAG escape therefore distinguishes corridor alternatives from local shortest-path variations. Certified exchange changes the tree support: length selection retains low-stretch exchanges, and topology selection favors large affected regions and fundamental cycles.

Table 1 and Figure 4 place CCE-KFF between the shortest-only and unconstrained-diversity extremes. Standard FF supplies one route, Shortest-Only FF remains inside the shortest-path structure, and Random-Tree FF attains diversity with impractically long routes. At matched cardinality, Yen and PNC exhibit substantially less separation than CCE-KFF-Length. CCE-KFF-Length remains near-shortest on Moving AI and preserves exact shortest length on StarCraft, where it also exceeds PNC and Plateaus in separation. Repeated-Penalty FF reaches a nearby quality point through repeated weighted field construction, providing the direct cost comparison in Section 5.3. CCE-KFF-Topology reaches a second obstacle-separated Boston corridor and produces broader edge support with lower peak concentration. The two selectors thus provide near-shortest separated families and deliberate corridor-level diversification within one reusable representation.

Table 1: Route quality at $K = 32$ and $M = 20$. Panel (a) evaluates reusable families on 33 Moving AI and 11 GPPC StarCraft maps; panel (b) uses the common 32-map Yen-complete cohort on Moving AI and all methods that completed on StarCraft. Entries are map macro-averages.

(a) Reusable field-family quality

	Moving AI (33 maps)						GPPC StarCraft (11 maps)					
Method	Routes↑	Mean↓	P95↓	Max↓	D_J ↑	Esc.↑	Routes↑	Mean↓	P95↓	Max↓	D_J ↑	Esc.↑
Standard FF*	1.00	1.0000	1.0000	1.0000	0.000	0.0000	1.00	1.0000	1.0000	1.0000	0.000	0.0000
Shortest-Only FF	26.81	1.0000	1.0000	1.0000	0.554	0.0000	31.97	1.0000	1.0000	1.0000	0.724	0.0000
Random-Tree FF	30.67	10.6885	20.1156	22.4686	0.891	0.9525	32.00	26.4195	51.4380	59.7692	0.974	1.0000
Penalty FF ($\alpha = .05$)	26.59	1.0010	1.0030	1.0045	0.603	0.0451	31.99	1.0046	1.0128	1.0161	0.836	0.2743
Penalty FF ($\alpha = .5$)	25.96	1.0075	1.0257	1.0427	0.614	0.1611	31.98	1.0200	1.0472	1.0563	0.838	0.5466
CCE-KFF-Length	26.83	1.0003	1.0009	1.0069	0.554	0.0011	31.97	1.0000	1.0000	1.0000	0.724	0.0000
CCE-KFF-Topology	28.27	1.6565	2.4422	2.7785	0.644	0.4318	31.97	1.9003	2.2457	2.3177	0.740	0.4455

(b) Source-specific path generators

	Moving AI			GPPC StarCraft		
Method	Routes↑	Stretch↓	D_J ↑	Routes↑	Stretch↓	D_J ↑
CCE-KFF-Length	26.684	1.001458	0.611	31.974	1.000000	0.724
Yen	26.684	1.001134	0.368		—	
PNC	26.684	1.001134	0.408	31.974	1.000000	0.203
Plateaus	26.680	1.092428	0.628	31.870	1.063563	0.496

*Standard FF is a single-route reference. Dashes mark unavailable experiments.

5.3 Computational Efficiency Evaluation

Shared-destination routing commonly serves game units, warehouse robots, or crowd agents in groups of different sizes. We therefore vary M from 1 to 20, covering isolated queries and common shared-goal workloads. The $M = 1$ endpoint measures compilation without reuse; larger values measure the benefit of serving a group from the same field family.

Repeated-Penalty FF provides a matched destination-side comparison. Yen, PNC, and Plateaus provide source-specific references as the number of shared-goal sources increases. Figure 3 separates compilation efficiency from destination-side amortization. At $M = 1$, before any field reuse is available, CCE-KFF is already $6.77\times$ faster than Yen; after only eight sources it also overtakes PNC and Plateaus. The advantage therefore emerges as soon as a small group of sources shares a destination, and reuse expands the gap to $850.65\times$ over Yen at $M = 20$. At comparable route quality and under the same destination-side interface, Repeated-Penalty FF is roughly $1.8\times$ slower because it recomputes K weighted shortest fields. The map-size comparison further shows that the advantage persists when Yen becomes unavailable and PNC becomes the reference. Together, these results isolate two gains: certified exchange avoids repeated field construction, and the compiled family amortizes that cost across every source sharing the destination.

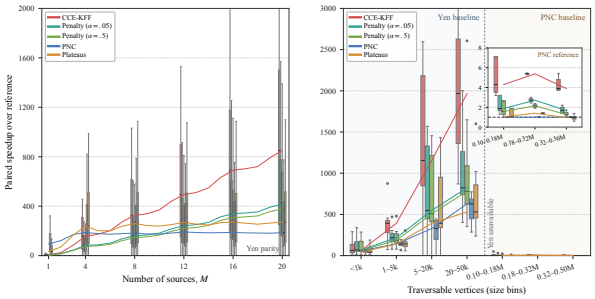


Figure 3: Paired $K = 32$ planning speedups by source count against Yen (left) and by map size against Yen or PNC (right); boxes show map-level distributions and curves show medians.

5.4 Ablation Studies

We isolate two compiler decisions: the selector chooses which certified exchanges are retained, and depth controls how many exchanges may be composed. We use the main benchmark manifests for the selector study and the Boston workload for depth.

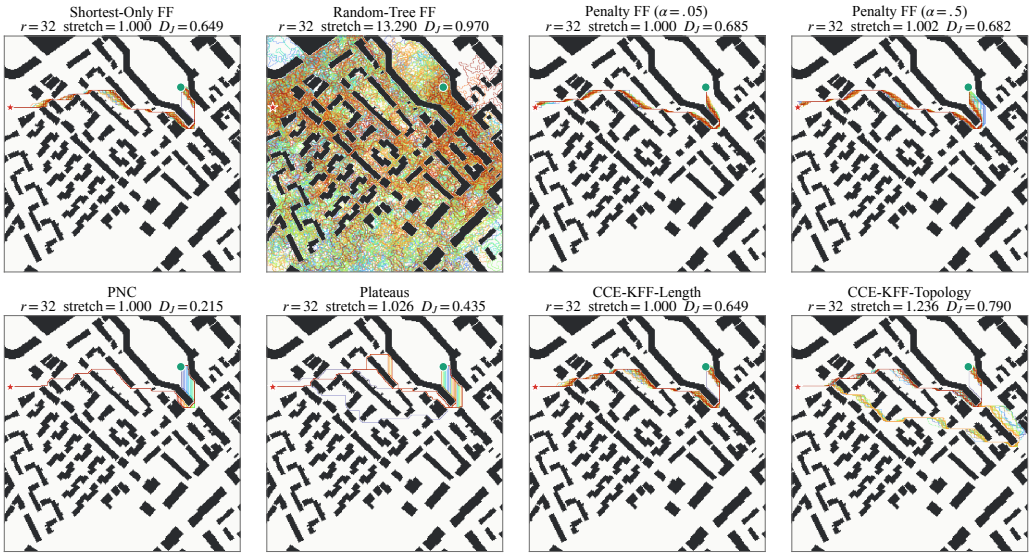


Figure 4: Eight $K = 32$ Boston route families with route count, mean stretch, and D_J.

Table 2: Selector and depth ablations at $K = 32$ on the main benchmarks and Boston.

		Moving AI				GPPC StarCraft			
Setting	Mode	Routes↑	Stretch↓	D_J ↑	Esc.↑	Routes↑	Stretch↓	D_J ↑	Esc.↑
(a) Effect of exchange-selection objective									
Length		26.831	1.000279	0.554	0.001	31.974	1.000000	0.724	0.000
Coverage		28.388	1.594793	0.640	0.521	31.974	1.724018	0.747	0.575
Topology		28.267	1.656487	0.644	0.432	31.974	1.900310	0.740	0.446
(b) Effect of exchange depth on Boston									
D		Length-aware				Topology-aware			
		Routes↑	Stretch↓	D_J ↑	Esc.↑	Routes↑	Stretch↓	D_J ↑	Esc.↑
0		1.000	1.000000	0.000	0.000	1.000	1.000000	0.000	0.000
1		31.990	1.000000	0.761	0.000	31.990	1.444282	0.823	0.248
2		30.990	1.000000	0.754	0.000	31.630	2.793676	0.880	0.730
3		30.990	1.000000	0.754	0.000	31.330	2.512628	0.877	0.881
4		30.990	1.000000	0.754	0.000	31.630	3.195874	0.914	0.916

Effect of exchange selection. Panel (a) compares three objectives at fixed $K = 32$ and $D = 1$. Length minimizes destination-wide mean route stretch and admits certified detours beyond the shortest-path DAG, giving nonzero Moving AI escape at mean stretch 1.000279. Coverage ranks affected-source mass and produces the largest route supply and escape rate. Topology additionally weights fundamental-cycle size, favoring obstacle-separated deviations and attaining the largest Moving AI D_J. The objectives therefore select near-shortest, broad-coverage, and topologically separated operating points, respectively.

Effect of exchange depth. Depth counts successive exchanges: $D = 0$ is the root, and each later stage expands the preceding K fields. With c non-tree edges, a stage examines $O(Kc)$ successors, making work linear in D for a fixed graph and budget; Section D gives the full bound. Depth expands reachable space, not output size. On Boston, length retains roughly 31 unit-stretch routes through $D = 4$, while topology raises D_J from 0.823 to 0.914 and DAG escape from 0.248 to 0.916; Figure 7 shows the newly reached obstacle-separated corridors.

6 Conclusion

CCE-KFF turns a single flow field into a reusable family of destination-reaching policies with $O(1)$ next-action lookup. We establish its tree correspondence, prove that certified cycle exchanges connect the family, and show that it realizes every simple source-to-destination route. On Moving AI and GPPC StarCraft, CCE-KFF produces near-shortest or topologically separated alternatives, reaches up to $850.65\times$ speedup, broadens edge support, and reduces route concentration.

AI USE STATEMENT

AI-based tools were used during the development and preparation of this work for literature retrieval, research brainstorming, assistance with code implementation and debugging, discussion and checking of mathematical arguments, drafting, language editing, and proofreading. AI-generated suggestions were treated as intermediate assistance rather than authoritative results. The authors independently evaluated and verified the mathematical claims, algorithmic design, cited literature, implementation, experimental results, and conclusions presented in this paper. All final research decisions and scientific claims were made and are the responsibility of the authors.

REFERENCES

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Alternative routes in road networks. In Paola Festa (ed.), *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pp. 23–34, Berlin, Heidelberg, 2010. Springer. doi: 10.1007/978-3-642-13193-6_3.

Vedat Akgün, Erhan Erkut, and Rajan Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121(2):232–246, 2000. doi: 10.1016/S0377-2217(99)00214-3.

Roland Bader, Jonathan Dees, Robert Geisberger, and Peter Sanders. Alternative route graphs in road networks. In *Theory and Practice of Algorithms in (Computer) Systems*, volume 6595 of *Lecture Notes in Computer Science*, pp. 21–32. Springer, 2011. doi: 10.1007/978-3-642-19754-3_5.

J. Adrian Bondy and U. S. R. Murty. *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, London, UK, 2008. doi: 10.1007/978-1-84628-970-5.

Seth Chaiken. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic and Discrete Methods*, 3(3):319–329, 1982. doi: 10.1137/0603033.

Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, Ulf Leser, and David B. Blumenthal. Finding K-shortest paths with limited overlap. *The VLDB Journal*, 29(5):1023–1047, 2020. doi: 10.1007/s00778-020-00604-x.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269–271, 1959. doi: 10.1007/BF01386390.

Elijah Emerson. Crowd pathfinding and steering using flow field tiles. In Steve Rabin (ed.), *Game AI Pro: Collected Wisdom of Game AI Professionals*, pp. 307–316. CRC Press, Boca Raton, FL, 2013. doi: 10.1201/b16725-29.

David Eppstein. Finding the K shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. doi: 10.1137/S0097539795290477.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11–15, Pasadena, CA, USA, 2008.

Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, See Woo Lee, and Yota Otachi. Computing diverse shortest paths efficiently: A theoretical and experimental study. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):3758–3766, 2022. doi: 10.1609/aaai.v36i4.20290.

Daniel Harabor and Alban Grastien. Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 1114–1119, 2011. doi: 10.1609/aaai.v25i1.7994.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

John Hershberger, Matthew Maxel, and Subhash Suri. Finding the K shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms*, 3(4):45, 2007. doi: 10.1145/1290672.1290682.

Carlos A. Holzmann and Frank Harary. On the tree graph of a matroid. *SIAM Journal on Applied Mathematics*, 22(2):187–193, 1972. doi: 10.1137/0122021.

Roger L. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002. doi: 10.1016/S0191-2615(01)00015-7.

Karthik Karur, Nitin Sharma, Chinmay Dharmatti, and Joshua E. Siegel. A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3):448–468, 2021. doi: 10.3390/vehicles3030027.

Naoki Katoh, Toshihide Ibaraki, and Hisashi Mine. An efficient algorithm for K shortest simple paths. *Networks*, 12(4):411–427, 1982. doi: 10.1002/net.3230120406.

Gustav Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik und Chemie*, 148(12): 497–508, 1847. doi: 10.1002/andp.18471481202.

Eugene L. Lawler. A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, 1972. doi: 10.1287/mnsc.18.7.401.

Michael Levandowsky and David Winter. Distance between sets. *Nature*, 234(5323):34–35, 1971. doi: 10.1038/234034a0.

Lingxiao Li, Muhammad Aamir Cheema, Mohammed Eunus Ali, Hua Lu, and Huan Li. Efficiently computing alternative paths in game maps. *World Wide Web*, 26(5):3321–3345, 2023. doi: 10.1007/s11280-023-01188-6.

Stephen B. Maurer. Matroid basis graphs. i. *Journal of Combinatorial Theory, Series B*, 14(3): 216–240, 1973. doi: 10.1016/0095-8956(73)90005-1.

James Oxley. *Matroid Theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, UK, 2 edition, 2011. doi: 10.1093/acprof:oso/9780198566946.001.0001.

Sachin Patil, Jur van den Berg, Sean Curtis, Ming C. Lin, and Dinesh Manocha. Directing crowd simulations using navigation fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):244–254, 2011. doi: 10.1109/TVCG.2010.33.

Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, Berlin, Germany, 2003. doi: 10.1007/978-3-662-07961-3.

James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. doi: 10.1073/pnas.93.4.1591.

Nathan R. Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144–148, 2012. doi: 10.1109/TCIAIG.2012.2197681.

Nathan R. Sturtevant, Jason M. Traish, James R. Tulip, Tansel Uras, Sven Koenig, Ben Strasser, Adi Botea, Daniel Harabor, and Steve Rabin. The grid-based path planning competition: 2014 entries and results. In *Proceedings of the Eighth Annual Symposium on Combinatorial Search*, pp. 241–250, 2015. doi: 10.1609/socs.v6i1.18364.

John W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974. doi: 10.1002/net.3230040204.

Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168, 2006. doi: 10.1145/1141911.1142008.

Jiongkun Yang, Xiai Chen, and Mingze Dong. Integration field-based breadth-first search for flow field pathfinding. *Scientific Reports*, 15(1):40580, 2025. doi: 10.1038/s41598-025-24412-x.

Jin Y. Yen. Finding the K shortest loopless paths in a network. *Management Science*, 17(11): 712–716, 1971. doi: 10.1287/mnsc.17.11.712.

Ali Al Zoobi, David Coudert, and Nicolas Nisse. Finding the k shortest simple paths: Time and space trade-offs. *ACM Journal of Experimental Algorithmics*, 28, 2023. doi: 10.1145/3626567.

A Proofs and Additional Theoretical Details

This section gives complete proofs for the principal theoretical results in Section 4 and records the guarantees used by the compiler.

A.1 Proof of the Flow-Field–Tree Correspondence

Proof of Theorem 4.1. Let $F \in \mathcal{F}(G, g)$. Every vertex has a finite successor chain ending at g, so the undirected support $T_F = (V, E_F)$ is connected. Each of the $n - 1$ non-destination vertices contributes one support edge. These edges are distinct: if two vertices contributed the same undirected edge, they would have to point to each other, creating a directed 2-cycle whose vertices could not reach g. Hence T_F is connected and has exactly $n - 1$ edges, and is therefore a spanning tree.

Conversely, let $T = (V, E_T)$ be a spanning tree of G. Because T contains a unique simple path from every $v \neq g$ to g, assigning $F_T(v)$ to the next vertex on this path defines a destination-reaching field. This orientation is unique. The two constructions are inverses: orienting T_F toward g recovers F, and discarding the directions of F_T recovers T. Thus the correspondence is bijective. The number of deterministic destination-reaching fields is consequently the number $\tau(G)$ of spanning trees of G, and the Matrix–Tree Theorem gives $\tau(G) = \det L_G^{(g)}$. Because every cofactor of the graph Laplacian has the same determinant, this number is independent of the chosen destination. □

A.2 Simple-Route Realizability

Let $P = (v_0 = s, v_1, \ldots, v_\ell = g)$ be any simple route in G. Its edges form a forest and can therefore be extended to a spanning tree T_P of the connected graph G. Orienting T_P toward g gives a destination-reaching field F_P by Theorem 4.1. The unique tree path from s to g is exactly P, so $P_{F_P}(s) = P$ and $L_{F_P}(s) = \ell$, proving Equation (2). Thus route realizability follows from the complete-field representation itself.

A.3 Proof of Certified Exchange and Its Exact Effect

Proof of Theorem 4.2. Let $T_F = (V, E_F)$ be the tree corresponding to F. Adding the non-tree edge e creates exactly one fundamental cycle $C_{T_F}(e)$. Removing any tree edge $f \in C_{T_F}(e) \setminus \{e\}$ breaks that cycle without disconnecting the graph. The resulting graph $T_{F'} = (V, (E_F \cup \{e\}) \setminus \{f\})$ is connected and has $n - 1$ edges, hence is a spanning tree. By Theorem 4.1, its orientation toward g is a field $F' \in \mathcal{F}(G, g)$.

Removing f from T_F produces the components $A = A(F; e, f)$ and $B = B(F; e, f)$, with $g \in B$. The endpoints of e lie in different components; write $e = \{a, b\}$ with $a \in A$ and $b \in B$. For $s \in B$, the unique tree path from s to g lies entirely in B, so it uses neither f nor e and remains unchanged. For $s \in A$, the unique new path first follows the unchanged path in $T_F[A]$ from s to a, crosses e, and then follows the unchanged path from b to g. Therefore the route changes exactly for the sources in A, and its length is

$$L_{F'}(s) = \begin{cases} L_F(s), & s \notin A, \\ \mathrm{dist}_{T_F[A]}(s, a) + 1 + L_F(b), & s \in A. \end{cases}$$

This is precisely Equation (3). □

For any $s \in A(F; e, f)$, the old and new s-to-g paths coincide outside the fundamental cycle. Inside that cycle, they take its two complementary arcs, so their edge symmetric difference is exactly $C_{T_F}(e)$. Every source in $A(F; e, f)$ therefore contributes $|C_{T_F}(e)|$, while every source outside the region contributes zero. Summing over all sources yields the impact identity in Equation (4).

A.4 Proof of Connectivity and Exact Exchange Distance

Proof of Theorem 4.3. Let $T_F = (V, E_F)$ and $T_{F^\star} = (V, E_{F^\star})$. If $E_F \neq E_{F^\star}$, choose $e \in E_{F^\star} \setminus E_F$. Adding e to T_F creates the fundamental cycle $C_{T_F}(e)$. This cycle contains an edge $f \in E_F \setminus E_{F^\star}$: otherwise every edge of the cycle would belong to $E_{F^\star}$, contradicting the acyclicity of $T_{F^\star}$. Exchanging f for e therefore produces a spanning tree whose intersection with $E_{F^\star}$ is larger

by exactly one. Repeating this construction reaches $T_{F^\star}$ after $|E_{F^\star} \setminus E_F|$ exchanges, and every intermediate tree corresponds to a valid destination-reaching field.

Conversely, one exchange inserts only one edge, so any sequence from F to $F^\star$ must contain at least $|E_{F^\star} \setminus E_F|$ exchanges. The constructed sequence is therefore shortest. Because both trees have $n-1$ edges, $|E_{F^\star} \setminus E_F| = |E_F \setminus E_{F^\star}| = \frac{1}{2}|E_F \triangle E_{F^\star}|$, proving Equation (5). Finally, the missing target edges are bounded both by $|E_{F^\star}| = n - 1$ and by the number $m - (n - 1) = m - n + 1$ of edges outside E_F, which gives Equation (6). □

A.5 Exact Budget, Nested Prefixes, and Scheduled Complexity

The compiler first emits distinct fields from its deterministic scheduled stream. If this stream produces fewer than K fields, the completion traversal explores unseen neighbors in the CCE policy graph while maintaining a global visited set. By Theorem 4.3, the policy graph is connected, so this traversal either reaches the requested budget or exhausts $\mathcal{F}(G, g)$. The resulting family therefore satisfies

$$|\mathcal{K}_g| = \min\{K, |\mathcal{F}(G, g)|\},$$

as stated in Equation (7).

All skeleton choices, exchange rankings, and tie-breaking rules are deterministic. The completion traversal also follows a fixed neighbor order. Consequently, the compiler defines a single ordered stream of distinct fields: for $K_1 < K_2$, the family returned with budget K_1 is a prefix of the family returned with budget K_2, provided that the graph, destination, and selector are unchanged.

On a four-neighbor grid, shortest-field and index preprocessing take $O(n \log n)$ time. The scheduled phase evaluates a constant-size candidate set for each field, and generating, evaluating, and materializing one field take $O(n)$ time. Producing $N \leq K$ scheduled fields therefore requires

$$T_{\text{scheduled}} = O(n \log n + nN), \qquad S_{\text{scheduled}} = O(n \log n + nN).$$

When the scheduled stream supplies the requested budget, these bounds become $O(n \log n + Kn)$. The $\Theta(nN)$ output term matches the space required to store N complete next-action tables.

In the Moving AI evaluation, the scheduled stream produced all $K = 32$ requested fields for 96 of the 99 destination instances. The remaining three instances belonged to the acyclic `maze-128-128-1` map, whose field space contains exactly one field; the completion traversal terminated without adding another field.

A.6 Depth-Bounded Compilation

Algorithm 1 makes the recursive compiler explicit. DepthOne is the scheduled compiler in Section 4.3; it returns each field together with the set of edges used by its exchange. For later layers, length selection ranks the resulting field by destination-wide mean stretch, while topology selection ranks cumulative topology mass. Retain keeps up to K candidates under the chosen selector, and field signatures are deduplicated across all roots and depths.

B Detailed Experimental Protocol

The recursive-depth study uses `Boston_0_256`, the destination at $(6, 77)$, and 100 frozen sources interleaved from four shortest-distance strata; this is the same destination used by Figure 4, whose marked source $(191, 55)$ is included. The main route-quality study uses 33 four-connected Moving AI maps, three fixed destinations per map, and 20 fixed reachable sources per destination, giving 99 destination-side compilations and 1,980 source instances. The GPPC 2014 StarCraft study uses all 11 benchmark maps under the same three-destination, 20-source protocol, giving another 33 compilations and 660 source instances. Destinations come from official scenarios; sources are interleaved from four shortest-distance strata so that every prefix covers both local and long-range queries. The cardinality-matched Yen comparison uses source prefixes $M \in \{1, 10, 20\}$. A separate frozen shared-destination workload uses ten destinations per Moving AI map and up to 100 reachable sources per destination, giving 330 compilations and 32,630 source instances; it supports the Shortest-Only and static-assignment analyses.

Algorithm 1 Depth-bounded CCE-KFF compilation

Require: Graph G, destination g, field budget K, depth $D \geq 0$, selector S
Ensure: At most K distinct destination-reaching fields within depth D

$F^{(0)} \leftarrow \textsc{ShortestField}(G, g)$
if $D = 0$ **then**
  return $\{F^{(0)}\}$
end if
$(\mathcal{L}, \mathcal{U}) \leftarrow \textsc{DepthOne}(G, g, K, S)$
if $D = 1$ **then**
  return $\mathcal{L}$
end if
$\mathcal{V} \leftarrow \{\mathrm{sig}(F) : F \in \mathcal{L}\}$
$\mathcal{Q} \leftarrow \{(F, \mathcal{U}(F)) : F \in \mathcal{L},\ |\mathcal{U}(F)| = 2\}$
for $d = 2, \ldots, D$ **do**
  $\mathcal{C} \leftarrow \emptyset$
  for all $(F, U) \in \mathcal{Q}$ **do**
    for all $(e, f) \in \textsc{CertifiedExchanges}(F)$ **do**
      if $e \in U$ or $f \in U$ **then**
        continue
      end if
      $F' \leftarrow \textsc{Exchange}(F, e, f)$, $U' \leftarrow U \cup \{e, f\}$
      if $\mathrm{sig}(F') \notin \mathcal{V}$ **then**
        add $(F', U', S(F'))$ to $\mathcal{C}$
      end if
    end for
  end for
  $\mathcal{Q}' \leftarrow \textsc{Retain}(\mathcal{C}, K, S)$
  if $\mathcal{Q}' = \emptyset$ **then**
    return $\{F : (F, U) \in \mathcal{Q}\}$
  end if
  $\mathcal{Q} \leftarrow \mathcal{Q}'$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathrm{sig}(F) : (F, U) \in \mathcal{Q}\}$
end for
return $\{F : (F, U) \in \mathcal{Q}\}$

The 33 maps are `Berlin_1_256`, `Boston_0_256`, `Paris_1_256`, `brc202d`, `den312d`, `den520d`, `empty-8-8`, `empty-16-16`, `empty-32-32`, `empty-48-48`, `ht_chantry`, `ht_mansion_n`, `lak303d`, `lt_gallowstemplar_n`, `maze-32-32-2`, `maze-32-32-4`, `maze-128-128-1`, `maze-128-128-2`, `maze-128-128-10`, `orz900d`, `ost003d`, `random-32-32-10`, `random-32-32-20`, `random-64-64-10`, `random-64-64-20`, `room-32-32-4`, `room-64-64-8`, `room-64-64-16`, `w_woundedcoast`, `warehouse-10-20-10-2-1`, `warehouse-10-20-10-2-2`, `warehouse-20-40-10-2-1`, and `warehouse-20-40-10-2-2`.

The 11 StarCraft maps are `AcrosstheCape`, `Aftershock`, `Archipelago`, `ArcticStation`, `Aurora`, `Backwoods`, `BigGameHunters`, `BlackLotus`, `BlastFurnace`, `BrokenSteppes`, and `Brushfire`.

Standard FF stores one distance-descending field. Shortest-Only FF varies only equal-cost shortest-parent choices and removes duplicate tables. Random-Tree FF samples reproducible depth-first spanning trees and measures the diversity and route length obtained without an explicit quality objective. Yen uses the loopless path enumerator in NetworkX 3.5 (Hagberg et al., 2008) and is called independently for each source. CCE-KFF is compiled once per destination and reused for all source prefixes. The principal near-shortest comparison uses the length-aware mode, whereas the route-spreading analyses use the topology-aware mode. Unless stated otherwise, CCE-KFF uses exchange depth $D = 1$.

For each source, we trace every compiled field to the destination, remove duplicate routes when computing route-family size, and retain field multiplicity when computing D_J. Stretch is measured against the graph shortest-path length. DAG escape is the fraction of induced routes containing an edge outside the shortest-path DAG. Metrics are first aggregated within their source and destination units and then across maps, so every map receives equal weight. Random seeds, destination manifests, source manifests, and assignment seeds are fixed and shared by the methods being compared.

CPU time is measured as primary-process CPU time on Python 3.12.4 under arm64 macOS. PNC and Yen use per-source limits of 300 and 1,200 seconds, respectively. Paired Yen summaries include only workloads completed by both methods, with the resulting map count reported in each table. CCE-KFF completed all 99 Moving AI destination-side compilations; 27 source-specific Yen queries reached the limit.

C Additional Route-Quality and Efficiency Results

Table 3 supplements the principal operating-point comparison with maximum stretch and DAG escape. Length-aware CCE-KFF remains close to shortest paths throughout the benchmark. Topology-aware CCE-KFF trades additional length for wider separation and much more frequent departure from the shortest-path DAG. Random-Tree FF confirms that validity and diversity alone do not control route quality.

Table 3: Operating points at $K = 32$ on 33 Moving AI maps.

Method	Routes	Mean stretch	P95	Max	D_J
CCE-KFF (Length)	26.83	1.0003	1.0009	1.0069	.554
CCE-KFF (Topology)	28.27	1.6565	2.4422	2.7785	.644
Random-Tree FF	30.67	10.6885	20.1156	22.4686	.891

The corresponding mean DAG-escape rates are 0.0011, 0.4318, and 0.9525 for length-aware CCE-KFF, topology-aware CCE-KFF, and Random-Tree FF, respectively. The length-aware mode therefore leaves the shortest-path DAG while retaining near-shortest route quality.

On the larger shared-destination workload at $K = 32$, Shortest-Only FF produces 26.89 distinct routes per source with $D_J = 0.551$, whereas topology-aware CCE-KFF produces 28.40 routes with $D_J = 0.642$. Shortest-Only FF has zero DAG escape by construction. CCE-KFF supplies at least one route outside the shortest-path DAG to 24,154 of 32,630 sources, or 74.02%. This comparison separates certified departures from equal-cost parent changes.

Table 4: StarCraft budget response. Each entry reports routes / mean stretch / D_J, macro-averaged over 11 maps.

K	CCE-Length	CCE-Topology	Penalty (.05)	Penalty (.5)	PNC	Plateaus
1	1.00 / 1.0000 / .000	1.00 / 1.0000 / .000	1.00 / 1.0000 / .000	1.00 / 1.0000 / .000	1.00 / 1.0000 / .000	1.00 / 1.0000 / .000
4	4.00 / 1.0000 / .763	4.00 / 1.6777 / .789	4.00 / 1.0019 / .831	4.00 / 1.0095 / .803	4.00 / 1.0000 / .146	4.00 / 1.0429 / .568
8	8.00 / 1.0000 / .741	8.00 / 1.8358 / .760	8.00 / 1.0030 / .836	8.00 / 1.0154 / .814	8.00 / 1.0000 / .155	8.00 / 1.0495 / .490
16	15.99 / 1.0000 / .729	15.99 / 1.8809 / .743	16.00 / 1.0045 / .841	16.00 / 1.0198 / .828	15.99 / 1.0000 / .174	15.98 / 1.0547 / .465
32	31.97 / 1.0000 / .724	31.97 / 1.9003 / .740	31.99 / 1.0046 / .836	31.98 / 1.0200 / .838	31.97 / 1.0000 / .203	31.87 / 1.0636 / .496

Table 5: StarCraft CPU seconds at $K = 32$ and $M = 20$. Destination-side methods report one compilation; source-specific methods report the cumulative 20-query time. Values average the three destinations on each map.

Map	CCE-KFF	Penalty (.05)	Penalty (.5)	PNC	Plateaus
AcrosstheCape	6.02	13.55	17.85	22.60	23.78
Aftershock	2.36	4.98	6.53	8.31	8.10
Archipelago	1.78	4.08	4.87	7.66	7.35
ArcticStation	5.95	14.63	18.90	32.30	23.50
Aurora	7.39	19.32	24.98	27.40	46.16
Backwoods	3.16	6.88	8.79	16.76	12.06
BigGameHunters	2.54	6.28	7.95	8.14	7.73
BlackLotus	4.90	8.92	11.24	26.78	18.97
BlastFurnace	5.74	13.90	18.78	27.58	26.12
BrokenSteppes	6.23	17.64	19.67	24.31	29.96
Brushfire	1.42	3.17	3.82	10.13	5.37
Mean	**4.32**	10.31	13.03	19.27	19.01

C.1 Detailed StarCraft Results

Table 4 gives the complete StarCraft budget response for the published baselines and internal controls. Length-aware CCE-KFF and PNC remain exactly shortest throughout, while CCE-KFF exposes substantially more separated routes: at $K = 32$, its $D_J = 0.724$ is $3.57\times$ PNC's 0.203. Repeated-Penalty FF attains larger D_J by recompiling 32 weighted shortest fields; Table 5 shows average CPU costs $2.39\times$ to $3.02\times$ those of CCE-KFF. Plateaus trades a 6.36% mean length increase for $D_J = 0.496$.

Figure 5 visualizes the workload with the largest increase in D_J from Shortest-Only FF to topology-aware CCE-KFF among StarCraft sources whose shortest distance lies between 100 and 900 and whose topology-aware mean stretch is below 2. Shortest-only methods remain concentrated in one corridor, Penalty FF and Plateaus create local detours, Random-Tree FF has much larger stretch, and certified exchange reaches a distinct obstacle-separated corridor while preserving a complete destination-side field family.

Table 6 reports every completed cardinality-matched operating point. The route count is identical for CCE-KFF and Yen within each paired workload by construction. CPU time includes one destination-side CCE-KFF compilation or the cumulative source-specific Yen or PNC searches for the stated source prefix.

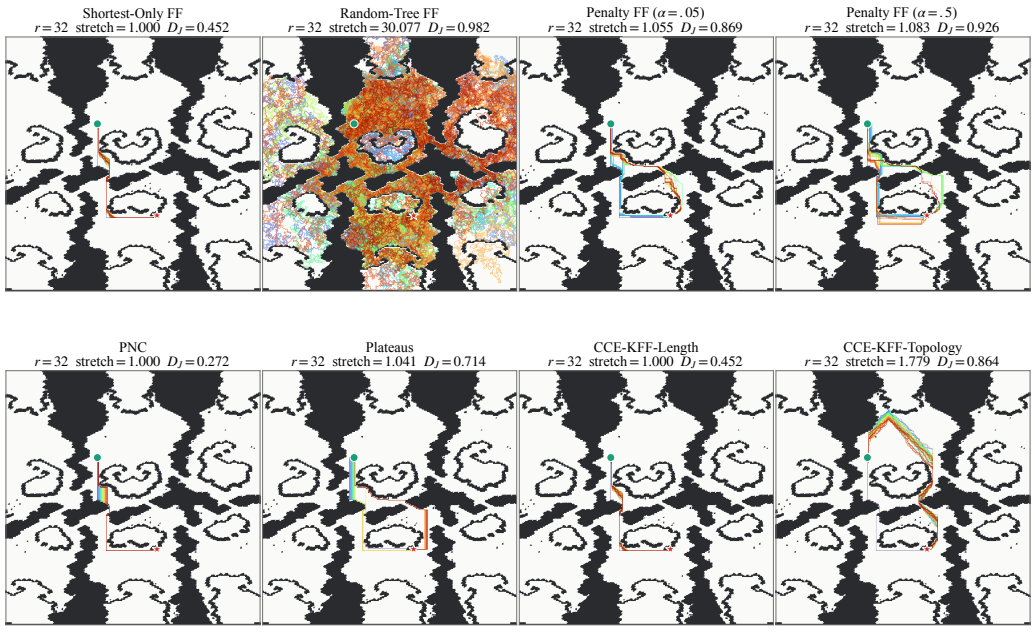


Figure 5: Eight $K = 32$ route families on `AcrosstheCape`. All panels share source, destination, crop, and color order. The title of each panel reports distinct routes, mean stretch, and edge-Jaccard distance.

Table 6: Full cardinality-matched comparison. Quality entries are CCE-KFF / Yen; PNC time is evaluated on the same completed map cohort in each row.

K	M	Maps	Routes	Mean stretch	D_J	CPU seconds	
						CCE / Yen	PNC
1	1	33	1.00	1.00000 / 1.00000	.000 / .000	.01 / .00	.009
1	10	32	1.00	1.00000 / 1.00000	.000 / .000	.01 / .03	.070
1	20	32	1.00	1.00000 / 1.00000	.000 / .000	.01 / .07	.136
4	1	33	3.46	1.00000 / 1.00000	.649 / .397	.06 / 1.14	.013
4	10	32	3.69	1.00015 / 1.00000	.656 / .303	.05 / 16.03	.102
4	20	32	3.69	1.00015 / 1.00000	.657 / .297	.05 / 36.75	.206
8	1	33	6.54	1.00000 / 1.00000	.629 / .394	.08 / 2.65	.016
8	10	32	7.14	1.00066 / 1.00017	.631 / .302	.06 / 37.42	.146
8	20	32	7.13	1.00040 / 1.00009	.630 / .296	.06 / 85.81	.304
16	1	33	12.19	1.00101 / 1.00101	.617 / .427	.11 / 5.66	.023
16	10	32	13.77	1.00079 / 1.00044	.617 / .335	.09 / 79.95	.237
16	20	32	13.80	1.00062 / 1.00034	.616 / .326	.09 / 183.30	.500
32	1	33	22.80	1.00101 / 1.00101	.613 / .457	.18 / 11.54	.036
32	10	32	26.53	1.00166 / 1.00128	.611 / .378	.14 / 163.24	.400
32	20	32	26.68	1.00146 / 1.00113	.611 / .368	.14 / 373.79	.858

All 99 CCE-KFF compilations completed. Each row uses the maps with complete Yen statistics for that operating point. CCE-KFF compilation is independent of M; the values at $M = 1$ and $M \in \{10, 20\}$ average over 33 and 32 maps, respectively.

Table 7 disaggregates the principal StarCraft operating point. Each entry pairs mean stretch with edge-Jaccard distance, exposing the same length–diversity trade-off on every map. Standard FF is omitted because it supplies only one route, and Yen is omitted because it completed no StarCraft map under the time limit.

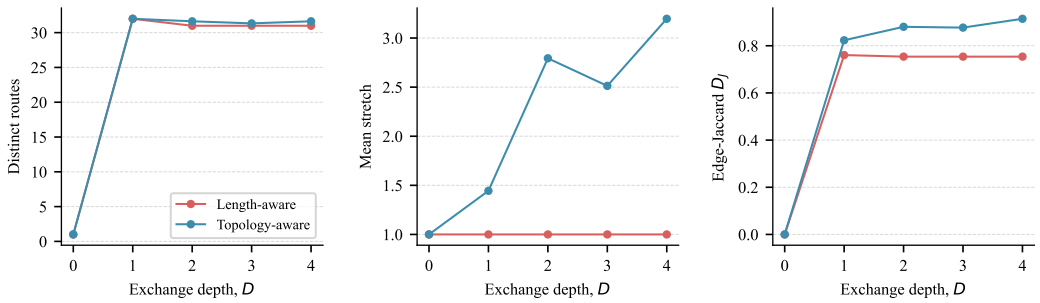


Figure 6: Recursive exchange depth on `Boston_0_256` at $K = 32$, averaged over 100 distance-stratified sources.

Table 7: Per-map StarCraft quality at $K = 32$ and $M = 20$. Each entry reports mean stretch / D_J, averaged over three destinations and 20 sources per destination.

Map	CCE-Length	CCE-Topology	Penalty (.05)	Penalty (.5)	PNC	Plateaus
AcrosstheCape	1.0000 / .693	2.0045 / .712	1.0070 / .836	1.0247 / .842	1.0000 / .237	1.0814 / .517
Aftershock	1.0000 / .744	1.0758 / .773	1.0017 / .866	1.0210 / .883	1.0000 / .319	1.0706 / .578
Archipelago	1.0000 / .784	3.0745 / .830	1.0058 / .853	1.0218 / .844	1.0000 / .219	1.0966 / .545
ArcticStation	1.0000 / .786	2.6157 / .808	1.0024 / .869	1.0117 / .874	1.0000 / .123	1.0430 / .332
Aurora	1.0000 / .729	1.9000 / .721	1.0028 / .821	1.0129 / .820	1.0000 / .111	1.0892 / .465
Backwoods	1.0000 / .769	1.1484 / .775	1.0018 / .842	1.0112 / .849	1.0000 / .158	1.0231 / .447
BigGameHunters	1.0000 / .720	1.1757 / .740	1.0016 / .826	1.0140 / .830	1.0000 / .318	1.0555 / .538
BlackLotus	1.0000 / .655	2.7327 / .659	1.0075 / .817	1.0268 / .829	1.0000 / .171	1.0588 / .533
BlastFurnace	1.0000 / .703	1.8538 / .710	1.0038 / .812	1.0257 / .808	1.0000 / .203	1.0690 / .511
BrokenSteppes	1.0000 / .659	2.0813 / .672	1.0109 / .807	1.0269 / .774	1.0000 / .180	1.0673 / .555
Brushfire	1.0000 / .722	1.2409 / .739	1.0055 / .852	1.0234 / .869	1.0000 / .199	1.0447 / .433

D Additional Ablation Results

Table 8 disaggregates the $K = 32$ operating points by map family. Length-aware CCE-KFF remains near-shortest in every family. Topology-aware CCE-KFF consistently increases route count and D_J, with the largest length trade-offs on warehouse, random, and empty maps.

Table 8: Map-family results at $K = 32$. Each entry reports routes / mean stretch / D_J.

Family	Length-aware CCE-KFF	Topology-aware CCE-KFF
Empty	18.633 / 1.002247 / .572	23.008 / 2.044265 / .686
Game/city	30.963 / 1.000002 / .605	31.347 / 1.380727 / .680
Maze	23.940 / 1.000008 / .353	24.380 / 1.033384 / .387
Random	24.650 / 1.000028 / .587	27.804 / 2.082425 / .737
Room	26.811 / 1.000010 / .512	28.606 / 1.400247 / .666
Warehouse	27.408 / 1.000003 / .622	28.583 / 2.710048 / .699

The exchange-depth experiment fixes $K = 32$ and applies each additional exchange to fields retained at the preceding depth. Length-aware selection ranks destination-wide mean stretch, whereas topology-aware selection ranks cumulative topology mass. If $c = m - n + 1$ is the number of non-tree edges, D stages evaluate $O(DKc)$ successors; with heavy–light cycle queries and field materialization, the implementation takes $O(DK(n + c \log n))$ time and stores $O(Kn)$ parent entries.

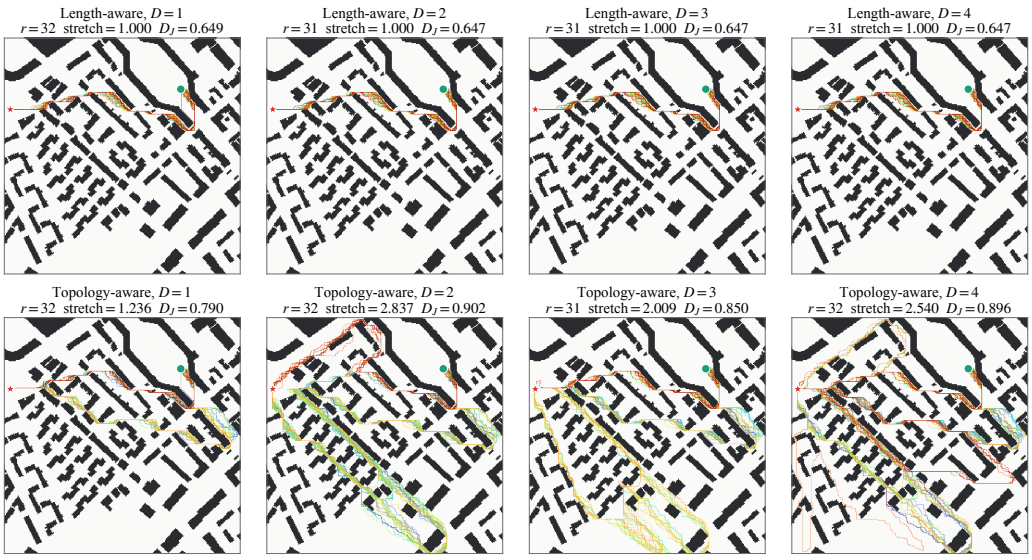


Figure 7: Routes from the marked Boston source as exchange depth increases. Length-aware selection remains on shortest corridors, while topology-aware selection progressively reaches additional obstacle-separated regions.

Table 9: Exchange-depth results at $K = 32$ on `Boston_0_256`, averaged over 100 distance-stratified sources. Candidates is the number of one-exchange successors evaluated at each stage.

Mode	D	Candidates	Fields	Routes	Stretch	D_J	Escape
Both	0	1	1	1.00	1.000000	.000	.000
Length	1	3,968	32	31.99	1.000000	.761	.000
Topology	1	3,968	32	31.99	1.444282	.823	.248
Length	2	1,328,939	32	30.99	1.000000	.754	.000
Topology	2	1,328,910	32	31.63	2.793676	.880	.730
Length	3	1,371,776	32	30.99	1.000000	.754	.000
Topology	3	1,371,729	32	31.33	2.512628	.877	.881
Length	4	1,371,744	32	30.99	1.000000	.754	.000
Topology	4	1,371,688	32	31.63	3.195874	.914	.916

The first stage evaluates 3,968 shortlisted candidates. Each later stage evaluates approximately 1.33–1.37 million successors in 7.8–8.8 CPU seconds per selector. Length-aware selection remains shortest, while topology-aware selection increases edge-Jaccard distance and DAG escape with depth. Every selected field was validated as a spanning tree, and every evaluated trace reached the destination.

Figure 7 visualizes the same trend: additional depth preserves the shortest-route family under length selection but expands the spatial reach of topology-aware families.

E Additional Static-Assignment Results

Table 10 gives the complete budget response for the frozen 330-destination static-assignment workload. The broader edge support and lower normalized peak emerge as the compiled family grows.

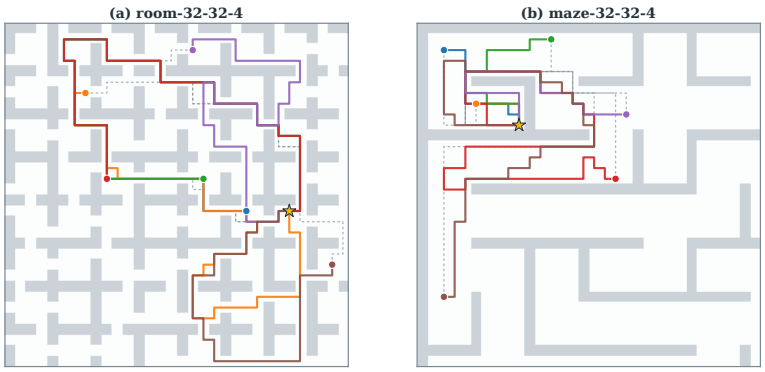


Figure 8: Representative assignments on `room-32-32-4` and `maze-32-32-4`. Routes from multiple sources use different parts of each shared-destination field family.

Table 10: Map-weighted static-assignment results. Gain and reduction compare topology-aware CCE-KFF with Shortest-Only FF.

K	SO edges	CCE edges	Gain	SO peak	CCE peak	Reduction
1	2,117.68	2,117.68	0.00%	.02232	.02232	0.00%
4	2,934.26	3,089.63	5.30%	.01827	.01792	1.92%
8	3,285.53	3,611.16	9.91%	.01811	.01702	6.01%
16	3,538.16	4,117.64	16.38%	.01811	.01696	6.35%
32	3,685.54	4,548.65	23.42%	.01819	.01674	7.96%

At $K = 1$, Standard FF, Shortest-Only FF, and CCE-KFF coincide. At $K = 32$, topology-aware CCE-KFF increases directed-edge support by 114.79% and lowers normalized peak edge pressure by 25.00% relative to Standard FF. Relative to Shortest-Only FF, the changes are 23.42% and 7.96%, respectively.

Table 11: Change from Shortest-Only FF by map family at $K = 32$. "Both" counts maps with broader support and a lower peak.

Family	Edge gain	Peak reduction	Both
Empty	70.4%	−3.1%	1/4
Game/city	18.9%	13.3%	13/13
Maze	3.8%	5.3%	4/5
Random	54.1%	18.6%	4/4
Room	34.6%	21.1%	3/3
Warehouse	37.6%	14.5%	3/4

The peak reduction is positive in five of the six map families. Empty maps are the exception: their highly symmetric geometry allows CCE-KFF to spread routes over many more edges without consistently lowering the single most loaded edge. Both improvements hold simultaneously on 28 of the 33 maps.

F Reproducibility and Validation

All reported field families use deterministic ordering and fixed tie breaking. The random-tree control uses fixed pseudorandom seeds, and all compared multi-field methods receive the same destination, source, budget, and assignment manifests. Budget curves are obtained from nested prefixes of one maximum-budget compilation rather than from separately resampled runs. The route tracer rejects repeated vertices, missing successor entries, and traces that fail to reach the destination.

Every emitted CCE-KFF object was checked for $n-1$ distinct support edges, connectivity, acyclicity, and destination reachability from every free cell. The compiler completed all 132 main-evaluation

destinations across Moving AI and StarCraft. The acyclic `maze-128-128-1` graph contains only one spanning tree, so every requested budget returns one field. The recursive-depth study applies the same validity checks to every selected field and associated trace.

The archived records retain per-map and per-destination field counts, route counts, stretch summaries, pairwise edge-Jaccard values, DAG-escape indicators, CPU times, timeout flags, and static-assignment statistics. These records support reconstruction of the reported aggregation and incomplete-workload exclusions.