SUFFICIENT TO ACT: WHAT A REPRESENTATION MUST REVEAL WHEN ITS CONSUMER OPTIMISES

Anonymous authors
Paper under double-blind review

ABSTRACT

Representation learning usually defines sufficiency relative to prediction, but many systems consume representations by solving an optimisation problem and returning an action. We ask what information a representation must preserve when its consumer optimises. For a fixed downstream objective, query sufficiency and action sufficiency can separate sharply: there are input families for which answering all relevant queries requires $\Omega(n)$ bits while recovering the optimal action requires only $O(1)$. For constraint-generated minimum-cost hitting-set decisions, we then characterise universal-cost action sufficiency exactly: two inputs induce the same complete family of optimal actions under all strictly positive costs if and only if their hypergraphs have the same antichain of inclusion-minimal generators. Its union gives the exact action support, identifying candidates that can be removed with zero decision loss. We also show that pseudonymisation does not remove structure-determined nuisances and give exact conditions for quotienting the remaining support while preserving the complete Argmin set. In a real-code study motivated by post-quantum cryptography (PQC) migration, the default model yields exact hypergraphs for 30 repositories; the action support retains a median 4.7% of graph vertices and the admissible quotient 3.5%, while pseudonymised full topology remains re-identifiable at about $19\times$ chance. On controlled synthetic instances, a graph-aware selector recovers the exact support on 37.0% of matched-budget cases but preserves the exact Argmin on 89.8% of sampled instance-cost pairs, showing that empirical objective preservation is weaker than universal action sufficiency.

1 INTRODUCTION

An organisation may want external advice about a private codebase without disclosing the source itself. It may instead release a derived artefact, such as a dependency graph with identifiers removed, to a third-party service or model that returns a remediation plan. This raises a representation question: *what information must the artefact preserve for the downstream decision, and what information can be omitted without changing that decision?*

Post-quantum cryptography (PQC) migration makes this representation question operationally immediate. Moving deployed systems away from quantum-vulnerable public-key cryptography is an organisation-wide transition involving cryptographic discovery, dependency analysis, prioritisation, and coordinated replacement (Joseph et al., 2022; Hasan et al., 2024). NIST has likewise outlined a transition from quantum-vulnerable algorithms to post-quantum standards and emphasised crypto-agility as a systems concern spanning software, protocols, hardware, and infrastructure (Moody et al., 2024; Barker et al., 2025). In such settings, an external decision service may need enough structural information to recommend interventions without needing the full private artefact from which those interventions were derived.

Classical notions of sufficiency are defined relative to inference: a representation preserves what is needed to predict a target, while minimality removes information irrelevant to that target (Tishby et al., 2000; Achille & Soatto, 2018); information-bottleneck and graph variants formalise this trade-off (Wu et al., 2020; Yu et al., 2021). Decision sufficiency for optimisation has also been studied, including objective-side sufficient information for linear optimisation (Bennouna et al., 2025), learned decision-sufficient representations (Ye et al., 2026), and action-sufficient state representations for control (Huang et al., 2022). We therefore do not claim the general notion of decision or action

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.

sufficiency as new. Our setting is complementary: a private structured input generates the feasible family, while the downstream objective remains external and may arrive only after release. The relevant notion is therefore *constraint-side* action sufficiency.

This distinction also separates our setting from predict-then-optimise and decision-focused learning, which learn parameters consumed by an optimiser (Elmachtoub & Grigas, 2022; Mandi et al., 2024; Wilder et al., 2019; Liu et al., 2024), and from query-preserving graph compression, which preserves answers rather than downstream actions (Fan et al., 2012). Mimicking networks provide a related precedent for exact task-aware compression (Krauthgamer & Rika, 2013). Redundancy of non-minimal hyperedges is classical (Eiter & Gottlob, 1995); our contribution is the resulting action-equivalence characterisation under all strictly positive costs, not minimal-transversal enumeration (Fredman & Khachiyan, 1996). Graph release can also leak topology and properties (Zhang et al., 2022; Nguyen-Cong et al., 2026), whereas our exact reductions identify distinctions that can be removed with zero decision loss; this is not by itself a privacy guarantee. Appendix A.3 gives the extended related-work discussion.

Contributions.

1. **Separation.** For a fixed strictly positive downstream cost vector, we give an input family for which answering every query requires $\Omega(n)$ bits while recovering the corresponding optimal action requires only $O(1)$ (Section 3.1).
2. **Exact characterisation.** For constraint-generated minimum-cost hitting-set decisions, two inputs induce the same complete family of optimal actions under all strictly positive costs if and only if their hypergraphs have the same antichain of inclusion-minimal constraint generators (Section 3.2).
3. **Exact disclosure reduction.** We identify the exact action support, show that pseudonymisation is structurally inert for topology-determined nuisances, and give structural and cost conditions under which a node quotient preserves the complete Argmin set (Section 3.3).
4. **Real-code validation.** In a PQC-motivated study of 34 analysable open-source codebases, the default MEDIATED model yields exact generator hypergraphs for 30 repositories. Their exact action support contains a median 4.7% of graph vertices, while the admissible graph-form quotient retains 3.5% under block-constant costs; pseudonymised full topology remains identifiable at about $19\times$ chance on the re-identification subset (Section 4).
5. **Learning as a surrogate.** On controlled synthetic instances, a GNN recovers the exact action support on 37.0% of matched-budget test instances, compared with 24.1% for an MLP and 0.3% for random selection, while preserving the exact Argmin on 89.8% of sampled instance-cost pairs (Section 4.2).

Scope and empirical instantiation. The theory applies whenever a private structured input induces a finite family of nonempty intervention constraints over a legal action set and the downstream consumer selects a minimum-cost set that intersects every induced constraint. This pattern arises in several security-remediation settings. PQC migration requires identifying and prioritising interventions across cryptographic dependencies (Joseph et al., 2022; Hasan et al., 2024); attack-graph analysis considers countermeasure selection over attack paths (Stan et al., 2021; Jmal et al., 2024); information-flow repair seeks program changes that eliminate violating flows (Tiwari et al., 2024); and access-control repair modifies policies while preserving required permissions (Eiers et al., 2023). These works motivate the application domains rather than imply that every instance has our form: the characterisation applies when each exposure or violation induces alternative legal intervention sites and the downstream objective reduces to the minimum-cost hitting-set decision of Section 2. We use PQC migration as the real-code instantiation because it combines a private structured artefact, path-generated intervention constraints, and a natural outsourcing setting. The empirical results concern our static import-graph and path-blocking abstraction; the hypergraph characterisation itself applies to the broader problem class above.

2 System Model and Problem Formulation

Private and public rooms. A *private room* holds a structured input G^* and releases $Z = T(G^*)$ before the downstream objective is known. A *public room*final receives Z together with public

decision parameters, in particular intervention costs, and returns a remediation plan for private execution. A nuisance variable S denotes information that the private room prefers not to disclose; whenever information leakage is considered, (G^*, S) is understood under the stated joint distribution.

Constraint generation. We model the private input as a typed attributed digraph $G^* = (V, E, \tau, \sigma, \mathbf{a}, X)$, where $\tau : E \to \mathcal{R}$ gives edge types, $\sigma : V \to \Sigma$ semantic labels, $\mathbf{a}$ attributes, and $X \subseteq V$ entry points. Let $\Sigma_{\text{target}} \subseteq \Sigma$ denote the target labels and $L \subseteq \mathcal{R}^*$ the admissible edge-type language. A path $p = v_0 \xrightarrow{r_1} \cdots \xrightarrow{r_k} v_k$ is an L-exposure if $v_0 \in X$, $\sigma(v_k) \in \Sigma_{\text{target}}$, and $r_1 \cdots r_k \in L$.

Only nodes in $V_{\text{act}} \subseteq V$ are legal intervention sites. Each remediable exposure path p induces a nonempty *generator* $e(p) \subseteq V_{\text{act}}$ containing the alternative sites whose individual intervention remediates that exposure. The induced generator hypergraph is $\mathcal{H}(G^*) = \{e(p) : p \text{ is a remediable } L\text{-exposure}\}$. In the real-code path-blocking instantiation, $e(p) = V(p) \cap V_{\text{act}}$; details are deferred to Section 4.1.

Downstream decision. For brevity write $\mathcal{H} = \mathcal{H}(G^*)$. A plan $x \subseteq V_{\text{act}}$ is feasible when it hits every generator, i.e.,

$$\mathcal{F}(\mathcal{H}) = \{x \subseteq V_{\text{act}} : x \cap e \neq \emptyset \text{ for every } e \in \mathcal{H}\}.$$

Given strictly positive intervention costs $c \in \mathbb{R}_{>0}^{V_{\text{act}}}$, the target action is the complete set of minimum-cost feasible plans,

$$A(\mathcal{H}, c) = \arg\min_{x \in \mathcal{F}(\mathcal{H})} \sum_{v \in x} c_v. \tag{1}$$

Because the release precedes the cost vector, it must preserve this decision for admissible costs supplied only after release.

Problem class. We call any task in which a private structured input induces a finite family of nonempty intervention constraints over V_{act}, followed by the minimum-cost hitting-set decision in (1), a *constraint-generated intervention problem*. Exposure paths are one mechanism for generating these constraints; attack traces, data-flow violations, and policy violations provide other instantiations when their legal interventions admit the same form (Appendix A.2). Our main characterisation acts on $\mathcal{H}$ once this domain-specific constraint family has been constructed.

This formulation separates three semantic layers: $G^* \to \mathcal{H}(G^*) \to A(\mathcal{H}, c)$. The first map is domain-specific, whereas our main characterisation acts on the induced constraint family. This distinction is what allows different application domains to use different extraction mechanisms while sharing the same action-sufficiency result.

Figure 1 summarises the role of the main objects before we develop the formal results. The graph-side representation G^* is first mapped to the decision constraints $\mathcal{H}$. The subsequent reductions operate on the decision problem: redundant generators are removed, their exact action support is identified, and admissible symmetries may then be quotiented. This separates transformations of the private graph from transformations that preserve the optimiser's action.

3 Action Sufficiency and Exact Reductions

We now make the decision-side reductions in Figure 1 precise, beginning with the distinction between information required to answer queries and information required to preserve an optimal action.

The central question is how much of the private constraint structure must be retained for the downstream optimiser. We first show that, under a fixed objective, preserving an optimal action can require asymptotically less information than answering queries about the private input. This motivates releases that discard distinctions irrelevant to the decision itself. Because the release must be chosen before the downstream costs are known, we then ask the stronger question: what must be preserved for every strictly positive cost vector?

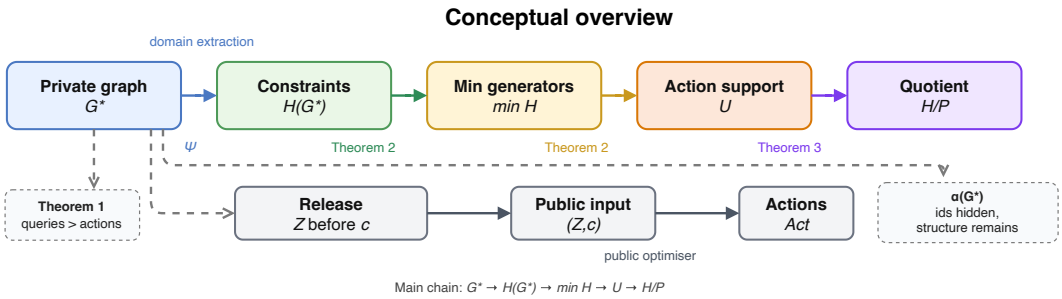

Figure 1: Overview of the release pipeline. The private graph G^* is mapped to the decision constraints $\mathcal{H}$; Theorem 2 characterises the exact universal-cost reduction and its support U, and Theorem 3 gives an exact quotient under admissibility. The release Z is fixed before c, after which the public optimiser receives (Z, c) and returns A. Pseudonymisation $\alpha(G^*)$ is graph-side and need not remove structural information.

3.1 Queries Can Require More Information Than Actions

For a query family $\mathcal{Q}$, a representation $Z = \phi(G^*)$ is *$\mathcal{Q}$-sufficient* if every answer in $\mathcal{Q}$ can be recovered from Z. For a fixed strictly positive cost vector c, it is *fixed-objective action-sufficient* if $A(\mathcal{H}(G^*), c)$ can be recovered from Z.

Theorem 1 (Fixed-Objective Separation). *There is an input family and a query family for which, under a fixed strictly positive cost vector, every $\mathcal{Q}$-sufficient representation requires $\Omega(n)$ bits, while an action-sufficient representation requires only $O(1)$ bits.*

The proof is given in Appendix C.1. A witness is indexed by $b = (b_1, \ldots, b_n) \in \{0,1\}^n$: it contains source nodes $x_1, \ldots, x_n$, a hub h, and a target r, with $x_i \to h$ present iff $b_i = 1$ and $h \to r$ always present. The n reachability queries recover b exactly. Each exposure has generator $e_i = \{x_i, h, r\}$; with $c_h = \frac{1}{2}$ and $c_{x_i} = c_r = 1$, the unique optimum is $\emptyset$ when $b = 0$ and $\{h\}$ otherwise. Hence the queries distinguish 2^n inputs, while the action distinguishes only two cases.

Theorem 1 establishes an asymptotic separation, so action sufficiency is not merely a constant-factor compression of query sufficiency. The witness, however, fixes the downstream objective in advance. In the two-room setting the release precedes the cost vector, so a representation must remain sufficient after an arbitrary admissible positive cost vector is supplied. This motivates the universal-cost notion below.

3.2 Universal-Cost Action Sufficiency

Once the objective is unknown at release time, the relevant object is the entire action map $c \mapsto A(H, c)$, not the optimum for one selected cost vector. The question is therefore which part of H determines this map exactly. A representation $Z = \phi(G^*)$ is *universal-cost action-sufficient* if $A(\mathcal{H}(G^*), c)$ can be recovered from (Z, c) for every $c \in \mathbb{R}_{>0}^{V_{\text{act}}}$. The first simplification is standard redundancy: if $e' \subseteq e$, then any plan hitting e' also hits e. Writing $\min \mathcal{H}$ for the inclusion-minimal generators,

$$\mathcal{F}(\mathcal{H}) = \mathcal{F}(\min \mathcal{H}). \tag{2}$$

For hypergraphs on the same actionable ground set, define *action equivalence* by

$$\mathcal{H}_1 \equiv_{\text{act}} \mathcal{H}_2 \quad \Longleftrightarrow \quad A(\mathcal{H}_1, c) = A(\mathcal{H}_2, c) \text{ for every } c \in \mathbb{R}_{>0}^{V_{\text{act}}}. \tag{3}$$

Equations (2) and (3) separate constraint redundancy from universal-cost action equivalence; the result below characterises the latter exactly. Let $U = \bigcup_{e \in \min \mathcal{H}} e$ be the *action support*. For a retained candidate set $R \subseteq V_{\text{act}}$, write $\mathcal{F}_R(\mathcal{H}) = \mathcal{F}(\mathcal{H}) \cap 2^R$ and let $A_R(\mathcal{H}, c)$ denote the corresponding minimum-cost plans.

Theorem 2 (Minimal-Generator Characterisation). *Let $\mathcal{H}$ be a finite hypergraph on V_{act}, and assume all intervention costs are strictly positive.*

- *Removing non-minimal generators does not change the feasible plans:* $\mathcal{F}(\mathcal{H}) = \mathcal{F}(\min \mathcal{H})$.
- *Two hypergraphs on the same actionable ground set are action-equivalent if and only if their inclusion-minimal generators are identical:*

$$\mathcal{H}_1 \equiv_{\mathrm{act}} \mathcal{H}_2 \quad \Longleftrightarrow \quad \min \mathcal{H}_1 = \min \mathcal{H}_2.$$

- *Every optimal plan is contained in* U*. Moreover, for any* $R \subseteq V_{\mathrm{act}}$*,*

$$A_R(\mathcal{H}, c) = A(\mathcal{H}, c) \text{ for every } c \in \mathbb{R}^{V_{\mathrm{act}}}_{>0} \quad \Longleftrightarrow \quad U \subseteq R.$$

The proof is given in Appendix C.2. Theorem 2 is the central structural result of the paper. For universal positive costs, the full private graph and the full generated constraint family contain distinctions that the downstream action does not require: the complete action behaviour is determined by the inclusion-minimal generator antichain $\min \mathcal{H}$. This gives the canonical decision release $Z_{\min} := \min \mathcal{H}$.

More strongly, the equivalence in Theorem 2 makes $\min \mathcal{H}$ the least deterministic sufficient statistic under functional refinement for the base universal-cost problem on a fixed actionable ground set. Let $Z = \phi(G^*)$ be any deterministic universal-cost action-sufficient release. If two inputs produce the same Z, sufficiency requires their complete action maps to agree; Theorem 2 then requires their minimal-generator antichains to be identical. Hence $\min \mathcal{H}$ is recoverable from every such Z. In this sense, every exact deterministic release must reveal the minimal-generator antichain, while $Z_{\min}$ reveals no additional decision-side distinction.

Its union $U = \bigcup_{e \in \min \mathcal{H}} e$ has a second exact interpretation: among pure candidate-set restrictions, retaining all of U is necessary and sufficient to preserve the complete Argmin set for every strictly positive cost vector. We write π for this restriction from V_{act} to U after $\mathcal{H}$ has been generated. It is a decision-side reduction, not permission to delete $V \setminus U$ from G^* before the constraints are constructed.

The universal quantifier is deliberate. If admissible costs are restricted to a smaller class, distinct minimal-generator antichains may become action-equivalent and a coarser sufficient release may exist. Theorem 2 therefore characterises the information required when the release cannot be tailored to a future objective; Appendix F.1 discusses the restricted-cost case.

Once $\mathcal{H}$ is available, computing $\min \mathcal{H}$ is straightforward; in graph instantiations the practical bottleneck may instead be constructing the constraint family itself.

3.3 Exact Release Transformations

Theorem 2 answers which constraints and candidates can be discarded without changing universal-cost decisions. A different question is whether candidates that must remain need to retain their individual identities. We first show why changing identifiers alone does not remove structure-determined information, then give conditions under which candidates inside U can be merged exactly. These operations must be distinguished from graph-side changes to G^* before constraint generation, which need not preserve the induced decision problem.

3.3.1 Structural Leakage Under Pseudonymisation

Let α change node identifiers or other attributes in $\mathbf{a}$ while preserving (V, E, τ, σ, X). If a nuisance S_{struct} is determined by the retained structure, then pseudonymisation cannot remove it:

$$I(S_{\mathrm{struct}}; \alpha(G^*)) = I(S_{\mathrm{struct}}; G^*) = H(S_{\mathrm{struct}}). \tag{4}$$

The same conclusion applies to isomorphism-invariant structural properties when vertices are also renamed. Section 4 measures the remaining structural leakage in the PQC-motivated study, and Appendix B.2 gives the information ordering for deterministically refined service outputs.

3.3.2 Exact Quotients of the Action Support

For $v \in U$, define its minimal-generator signature $S(v) = \{e \in \min \mathcal{H} : v \in e\}$. A partition P of U together with a cost class $\mathcal{C}_P \subseteq \mathbb{R}^{V_{\mathrm{act}}}_{>0}$ is *admissible* when (i) sites in the same block have

identical signatures and (ii) every $c \in \mathcal{C}_P$ is constant on each block. The first condition makes every block either contained in or disjoint from each minimal generator; the second makes its members indistinguishable to the objective.

For $e \in \min \mathcal{H}$, let $\bar{e} = \{B \in P : B \subseteq e\}$ and $\mathcal{H}/P = \{\bar{e} : e \in \min \mathcal{H}\}$. For $c \in \mathcal{C}_P$, let $\bar{c}_B$ be the common cost of block B. For a quotient plan $Y \subseteq P$, define

$$\mathrm{Lift}_P(Y) = \{x \subseteq U : |x \cap B| = 1 \text{ for } B \in Y,\ x \cap B = \emptyset \text{ for } B \notin Y\}.$$

Theorem 3 (Exact Admissible Quotient)**.** *For an admissible partition $(P, \mathcal{C}_P)$ and every $c \in \mathcal{C}_P$,*

$$A(\mathcal{H}, c) = \bigcup_{Y \in A(\mathcal{H}/P, \bar{c})} \mathrm{Lift}_P(Y).$$

The proof is given in Appendix C.3.

Thus, solving the quotient problem and lifting its optimal block plans recovers the complete set of optimal plans of the original problem. The quotient provides a different reduction from support restriction: U removes candidates globally, whereas P removes distinctions among retained candidates only when both the constraint structure and the admissible costs make them interchangeable.

Both admissibility conditions are necessary. Structural compatibility alone does not preserve choices when block members have different costs, while equal costs alone do not justify merging sites that occur in different minimal generators. Under the stated conditions, the decision-side reduction is $\mathcal{H} \to \min \mathcal{H} \to U \to \mathcal{H}/P$. We denote the quotient operation by γ_P.

The theorem establishes exactness for this construction, not global coarseness among all possible releases; Appendix C.4 gives the scope of that claim and Appendix B.4 treats the separate typed graph-side contraction.

4 Empirical Evaluation

The theory gives exact reductions, but leaves two empirical questions. First, are the exact action support and admissible quotient materially smaller than the structured inputs encountered in practice? Second, when constructing the exact constraint family is expensive, can a learned selector approximate the action support well enough to preserve downstream decisions? We study the first question on real codebases and the second on controlled synthetic graphs, where exact support and complete decision ground truth can be computed.

4.1 Evaluation on Real Codebases

We instantiate the framework on static in-repository import graphs using an explicit path-blocking intervention model. PQC migration is a useful measured case because exposure constraints arise from software dependencies while the underlying codebase and its architecture may be commercially sensitive.

Corpus. The corpus contains 52 open-source repositories across Go, Java, Python, and JavaScript. Nodes are source files, except that Go uses packages, and edges represent static imports. Targets are source units associated with Shor-vulnerable schemes and entry points are detected using language-specific rules. Of the 52 repositories, 34 are analysable: 10 have no detected target and 8 have no detected exposure path. We evaluate three actionability models: STRICT makes only target nodes actionable, MEDIATED additionally allows their direct importers, and OPEN allows every node on an exposure path. Generators are enumerated exactly after reachability-preserving contraction; repositories exceeding the enumeration budget are excluded rather than approximated. This removes none under STRICT, four under MEDIATED, and seven under OPEN. Full construction and filtering details are given in Appendix D.1.

Exact support and quotient. For topology measurements we use the induced support graph $G_U = G^*[U]$ and its admissible block quotient $G_{U/P}$. These graph-form artefacts are optional; the canonical decision release remains $Z_{\min} = \min \mathcal{H}$ by Theorem 2. The median retained fractions are reported in Table 1.

Table 1: Median retained vertex fractions and exact-hypergraph coverage under the three actionability models.

Artefact	STRICT	MEDIATED	OPEN
Support G_U	2.0%	4.7%	6.4%
Admissible quotient $G_{U/P}$	**2.0%**	**3.5%**	**5.3%**
Repositories	34/34	30/34	27/34

The dominant exact reduction is therefore the restriction to U. Under the default MEDIATED model, G_U retains a median 4.7% of the original vertices and the quotient retains 3.5%. The median quotient ratio $|P|/|U|$ is 1.000, the quotient is the identity on 21 of 30 repositories, and the largest admissible block has four nodes.

A cheaper surrogate that groups nodes by (reachable schemes, reaching entries) appears more aggressive but violates the structural condition of Section 3.3.2. Under MEDIATED, its median quotient ratio is 0.500 rather than 1.000, and its largest block has 130 rather than four nodes. It still retains 4.4% of the original graph, compared with 3.5% for the exact construction, because it keeps exposure-subgraph nodes outside the exact support. This illustrates that apparent compression need not preserve the decision. As a consistency check, 1,380 cost-vector tests on 69 corpus-derived hypergraphs produced no mismatch in the quotient identity of Theorem 3.

Structural leakage. Equation (4) implies that pseudonymisation cannot remove a private property determined by retained topology. We test practical leakage by matching each released graph to a structural fingerprint from a perturbed reference snapshot. The fingerprint excludes explicit graph-size features. Under MEDIATED, the 24-repository subset with $|U| \geq 3$ has random-chance accuracy 4.2%; the results are reported in Table 2.

Table 2: Closed-world repository re-identification accuracy under structural perturbation level q for the 24-repository MEDIATED subset with $|U| \geq 3$. Random chance is 4.2%.

Release	$q = 0.05$	$q = 0.15$	$q = 0.30$
Pseudonymised full topology $\alpha(G^*)$	79.2%	**80.6%**	56.9%
Support topology G_U	20.8%	20.8%	16.7%
Block quotient $G_{U/P}$	18.1%	20.8%	**18.1%**

At $q = 0.15$, pseudonymisation alone leaves the repository identifiable at about $19\times$ chance. Restricting to G_U reduces accuracy from 80.6% to 20.8%, while admissible quotienting gives essentially no further reduction, consistent with the near-identity quotient above. Additional corpus statistics, actionability results, and re-identification variants are reported in Appendix D.

4.2 Learning Action-Sufficient Representations

Learning is not required when $\min \mathcal{H}$ can be constructed exactly, because $U = \bigcup_{e \in \min \mathcal{H}} e$ is then available directly. We instead treat a learned selector as a computational surrogate for settings in which exact constraint construction is costly. Given a graph, the selector predicts a retained set R without explicitly reconstructing $\min \mathcal{H}$; exact U is used only as training and evaluation ground truth.

Setup. We use controlled typed layered DAGs because exact ground truth is required for every instance. Graphs contain between 14 and 30 nodes, six edge types, entry and target nodes, and an actionable set. The exposure language excludes paths containing a `test` edge and requires at least one `flows` edge. After exact enumeration, retained instances satisfy $4 \leq |U| \leq 12$. We use 1200 training, 400 validation, and 400 disjoint test instances and report learned results over three training seeds. RANDOM samples under the same node budget, MLP-SUP scores nodes from local features, and GNN-SUP additionally uses typed message passing. Both learned models predict $y_v = \mathbf{1}[v \in U]$. Features and ablations are detailed in Appendix E.

Capacity and support recovery. For a fixed budget k, some instances are impossible because $|U| > k$. We therefore report the capacity ceiling $C(k) = \Pr(|U| \le k)$, overall support success $S(k) = \Pr(U \subseteq R_k)$, and conditional ranking success $Q(k) = \Pr(U \subseteq R_k \mid |U| \le k)$. We evaluate $k \in \{4, 5, 6, 7, 8, 10, 12\}$.

Figure 2 separates capacity from ranking error. At $k = 4$, $C(k) = 22.2\%$ and GNN-SUP achieves $S(k) = 9.7 \pm 1.1\%$ and $Q(k) = 43.8 \pm 4.9\%$. At $k = 8$, its overall success is $62.3 \pm 0.8\%$, compared with $58.4 \pm 0.6\%$ for MLP-SUP and $7.9 \pm 1.0\%$ for Random; the corresponding conditional values are $77.2 \pm 1.0\%$, $72.3 \pm 0.8\%$, and $9.8 \pm 1.2\%$. At $k = 12$, capacity is no longer limiting and GNN-SUP reaches $94.7 \pm 0.4\%$.

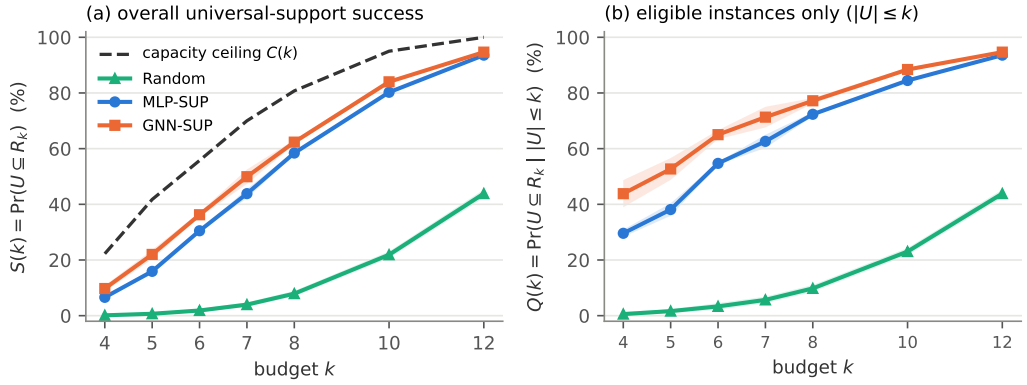

Figure 2: Fixed-budget recovery of the exact action support on 400 held-out synthetic instances. Panel (a) shows overall success $S(k)$ and the capacity ceiling $C(k)$; panel (b) shows conditional ranking success $Q(k)$ over instances satisfying $|U| \le k$. The eligible set grows with k and contains all 400 test instances at $k = 12$. Learned results are means over three seeds and Random over 30 draws.

Matched-budget decision preservation. To remove capacity as a confound, we set $k_i = |U_i|$, so $U_i \subseteq R_i$ is equivalent to exact recovery $R_i = U_i$. The resulting support and decision metrics are reported in Table 3.

Table 3: Matched-budget results with $k_i = |U_i|$. Decision metrics use 32 sampled positive cost vectors per instance. Values are percentages.

Method	Exact U	Recall	Pos. regret	Exact Argmin
Random	0.3 ± 0.3	47.0 ± 0.6	18.5 ± 1.4	24.1 ± 1.8
MLP-SUP	24.1 ± 1.7	81.7 ± 0.6	13.9 ± 0.8	85.2 ± 0.4
GNN-SUP	$\mathbf{37.0 \pm 2.4}$	$\mathbf{86.5 \pm 0.8}$	$\mathbf{9.4 \pm 0.7}$	$\mathbf{89.8 \pm 0.5}$
Oracle-U	100.0	100.0	0.0	100.0

The matched-budget comparison isolates ranking quality: GNN-SUP recovers the exact support on $37.0 \pm 2.4\%$ of test instances, compared with $24.1 \pm 1.7\%$ for MLP-SUP and $0.3 \pm 0.3\%$ for Random. Exact support recovery is stronger than success for a sampled objective. Although the GNN recovers U exactly on only 37.0% of instances, it preserves the complete Argmin set on $89.8 \pm 0.5\%$ of sampled instance-cost pairs. These sampled-cost outcomes are diagnostic rather than universal: only $U \subseteq R$ guarantees preservation for every strictly positive cost vector by Theorem 2.

Thus the learning experiment supports a narrower conclusion than the exact theory. Graph structure improves prediction of the decision-relevant support, but empirical preservation on sampled objectives can substantially overstate universal sufficiency. The synthetic instances are intentionally small enough for exact ground truth, and we make no runtime, speedup, impossibility, or repository-scale transfer claim. Full training, feature, ablation, regret, and robustness results are given in Appendix E.

5 LIMITATIONS

The real-code study uses static in-repository import graphs and a lightweight cryptographic-target detector, so it can miss runtime dependencies and does not establish that a detected vulnerable operation is executed. Its path-blocking model also treats interventions as atomic sites and does not model coordinated edits or complete PQC migration. Exposure paths stop at the first crypto-labelled target. Exact generator enumeration is budgeted; under the default MEDIATED model, four repositories exceed the budget and are excluded from exact-hypergraph analyses. The reported statistics therefore characterise this structural abstraction and the subset for which $\min \mathcal{H}$ is constructed exactly. The re-identification experiment is closed-world and concerns optional graph-form releases rather than the canonical statistic $\min \mathcal{H}$. In addition, the original repository heads were not archived by commit hash, so reproducing the corpus from current heads need not yield identical graphs.

The learning study is intentionally controlled and small: graphs contain 14 to 30 nodes and the retained instances satisfy $4 \leq |U| \leq 12$. We evaluate one node-local MLP and one typed message-passing GNN with three training seeds and fixed hyperparameters; the feature set also includes the structural prior V_π, whose effect is reported by ablation. Decision metrics use 32 sampled positive cost vectors per test instance, so sampled exact-Argmin rates do not establish universal sufficiency; only $U \subseteq R$ guarantees preservation for every strictly positive cost vector. We do not compare exact-extraction and learned-inference runtime, and therefore make no speedup, impossibility, architecture-benchmark, or repository-scale transfer claim. Appendices E and F give the full scope and protocol details.

6 CONCLUSION

For constraint-generated minimum-cost hitting-set decisions, universal action equivalence is determined exactly by the antichain of inclusion-minimal generators. This yields the exact action support U and, under additional structural and cost conditions, an exact quotient of that support. In the PQC-motivated real-code study, the default MEDIATED model produces exact generator hypergraphs for 30 repositories; U retains a median 4.7% of graph vertices and the admissible graph-form quotient retains 3.5%. Pseudonymised full topology remains re-identifiable at about $19\times$ chance on the evaluated subset, showing that identifier removal alone does not remove structural disclosure.

The synthetic study shows a complementary computational picture. At matched budget, the GNN recovers the exact support on 37.0% of held-out instances but preserves the complete Argmin set on 89.8% of sampled instance-cost pairs. This gap separates empirical preservation for sampled objectives from the universal guarantee $U \subseteq R$. The same perspective extends beyond the measured PQC setting whenever a private structured system generates remediation constraints and an external consumer optimises over the resulting intervention choices. Overall, the results support designing releases around the downstream action rather than the input representation alone. Future work includes certified approximate releases, richer cost and intervention models, scalable surrogates with explicit guarantees, and validation on more faithful program analyses.

AI USE STATEMENT

Large language models were used as research assistants for literature triage, prose editing, counterexample generation, proof checking, candidate proof steps, and code for verification experiments. Claude and ChatGPT were used iteratively during theoretical development and manuscript revision. The authors independently checked the mathematical statements, experimental claims, code, and final text and take full responsibility for the submission.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19, 2018.

Ravindra K. Ahuja and James B. Orlin. Inverse optimization. *Operations Research*, 49(5), 2001.

Elaine Barker, Lily Chen, David Cooper, Dustin Moody, Andrew Regenscheid, Murugiah Souppaya, William Newhouse, Russ Housley, Sean Turner, William Barker, and Karen Kent. Considerations

for achieving crypto agility: Strategies and practices. NIST Cybersecurity White Paper NIST CSWP 39-upd1, National Institute of Standards and Technology, 2025. Includes updates as of June 29, 2026.

Omar Bennouna, Amine Bennouna, Saurabh Amin, and Asuman Ozdaglar. What data enables optimal decisions? an exact characterization for linear optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

David Blackwell. Equivalent comparisons of experiments. *The Annals of Mathematical Statistics*, 24 (2), 1953.

CycloneDX. CycloneDX cryptography bill of materials (CBOM). https://cyclonedx.org/capabilities/cbom/, 2026.

William Eiers, Ganesh Sankaran, and Tevfik Bultan. Quantitative policy repair for access control on the cloud. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 564–575, 2023.

Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 1995.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 2022. arXiv:1710.08005.

Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. Query preserving graph compression. In *ACM SIGMOD International Conference on Management of Data*, pp. 157–168, 2012.

Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 1996.

Khondokar Fida Hasan, Leonie Simpson, Mir Ali Rezazadeh Baee, Chadni Islam, Ziaur Rahman, Warren Armstrong, Praveen Gauravaram, and Matthew McKague. A framework for migrating to post-quantum cryptography: Security dependency analysis and case studies. *IEEE Access*, 12: 23427–23450, 2024. doi: 10.1109/ACCESS.2024.3360412.

Andrés F. Herrera, Mehraj Gadimaliyev, Ying Meng, and Licheng Wang. A graph-based framework for cryptographic risk modeling in the post-quantum transition: TLS 1.3 and TLCP analysis. *Information and Software Technology*, 195:108099, 2026. doi: 10.1016/j.infsof.2026.108099.

Biwei Huang, Chaochao Lu, Liu Leqi, Jose Miguel Hernandez-Lobato, Clark Glymour, Bernhard Schölkopf, and Kun Zhang. Action-sufficient state representation learning for control with structural constraints. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9260–9279, 2022.

Jinu Hyeon, Woobin Park, Hongjoon Ahn, and Taesup Moon. Action-sufficient goal representations. *arXiv preprint arXiv:2601.22496*, 2026.

Patrick Indri, Tamara Drucks, and Thomas Gärtner. On the trade-off between expressivity and privacy in graph representation learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Houssem Jmal, Firas Ben Hmida, Nardine Basta, Muhammad Ikram, Mohamed Ali Kaafar, and Andy Walker. SPGNN-API: A transferable graph neural network for attack paths identification and autonomous mitigation. *IEEE Transactions on Information Forensics and Security*, 19:1601–1613, 2024. doi: 10.1109/TIFS.2023.3338965.

David Joseph, Rafael Misoczki, Marc Manzano, Joe Tricot, Fernando Dominguez Pinuaga, Olivier Lacombe, Stefan Leichenauer, Jack Hidary, Phil Venables, and Royal Hansen. Transitioning organizations to post-quantum cryptography. *Nature*, 605(7909):237–243, 2022.

Robert Krauthgamer and Inbal Rika. Mimicking networks and succinct representations of terminal cuts. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1789–1799, 2013. doi: 10.1137/1.9781611973105.128.

Yang Liu, Chuan Zhou, Peng Zhang, Shirui Pan, Zhao Li, and Hongyang Chen. Decision-focused graph neural networks for combinatorial optimization. *arXiv preprint arXiv:2406.03647*, 2024.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024. doi: 10.1613/JAIR.1.15320.

Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. NIST Internal Report NIST IR 8547 ipd, National Institute of Standards and Technology, November 2024. Initial Public Draft.

Sidney A. Morris. Decision-sufficient feature sets, decision cores, and decision complexes: A topological approach to classification. *Axioms*, 15(7):549, 2026. doi: 10.3390/axioms15070549.

Thinh Nguyen-Cong, Truong-Son Hy, and Thang N. Dinh. Spectral embeddings leak graph topology: Theory, benchmark, and adaptive reconstruction. *arXiv preprint arXiv:2604.21094*, 2026.

Victor Spitzer and Francois Sanson. Managing solution stability in decision-focused learning with cost regularization. *arXiv preprint arXiv:2601.21883*, 2026.

Orly Stan, Ron Bitton, Michal Ezrets, Moran Dadon, Masaki Inokuchi, Yoshinobu Ohta, Tomohiko Yagyu, Yuval Elovici, and Asaf Shabtai. Heuristic approach for countermeasure selection using attack graphs. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pp. 1–16. IEEE, 2021.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Abhishek Tiwari, Jyoti Prakash, Zhen Dong, and Carlo A Furia. Automated repair of information flow security in android implicit inter-app communication. In *International Symposium on Formal Methods*, pp. 285–303. Springer, 2024.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI Conference on Artificial Intelligence*, 2019.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yuhan Ye, Saurabh Amin, and Asuman Özdağlar. Learning decision-sufficient representations for linear optimization. In *Proceedings of the Thirty-Ninth Conference on Learning Theory*, volume 336 of *Proceedings of Machine Learning Research*, pp. 6924–6975, 2026.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations (ICLR)*, 2021.

Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *USENIX Security Symposium*, 2022. arXiv:2110.02631.

A Reference, Scope, and Related Work

A.1 Core Concepts and Notation

This appendix is a reference map, not a substitute for local definitions: any concept used by a theorem is defined in the main text before that theorem. The paper deliberately separates three semantic layers,

$$\underbrace{G^*}_{\text{graph layer}} \longrightarrow \underbrace{\mathcal{H}(G^*) \longrightarrow \min \mathcal{H}}_{\text{constraint layer}} \longrightarrow \underbrace{A(\mathcal{H}, c)}_{\text{decision layer}}, \qquad U = \bigcup_{e \in \min \mathcal{H}} e \subseteq V_{\text{act}}.$$

The graph generates constraints; the inclusion-minimal constraints determine the universal-cost action class; and U is the support on which an optimal action can live. In particular, restricting the

optimisation ground set to U is not the same operation as deleting $V \setminus U$ from the private graph before the constraints are derived. Table 4 summarises the core concepts and their semantic roles.

Table 4: Core concepts. Terms that enter formal results are defined at first use in the main text; this table fixes their intended meaning and scope in one place.

Term	Meaning in this paper	Layer / scope
Private room	Party that holds G^*, computes the release before seeing the downstream cost vector, and later executes the returned action.	system
Public room	External consumer that receives the release, supplies public decision parameters such as costs, and solves the downstream optimisation.	system
Constraint-generated intervention problem	A task in which a private structured input induces a family of nonempty subsets of V_{act}, and the downstream action is a minimum-cost set intersecting every induced subset. The generator construction may come from paths, traces, rules, or another domain-specific mechanism.	problem class
Query sufficiency	A representation is $\mathcal{Q}$-sufficient when every answer in the specified query family $\mathcal{Q}$ is recoverable from it.	representation
Action sufficiency	A representation is sufficient for a specified action map when the complete returned action is recoverable from it.	representation
Fixed-objective action sufficiency	Action sufficiency for one fixed downstream objective, such as the fixed cost vector in Theorem 1.	decision
Universal-cost action sufficiency	Sufficiency for the entire family $\{A(\mathcal{H}, c) : c > 0\}$ when the positive cost vector is unknown at release time.	decision
Exposure path	An L-accepted path from an entry point in X to a target-labelled node.	graph
Actionable generator	$e(p)$, the actionable sites on exposure path p whose individual intervention is sufficient to remediate that exposure. Under the real-code path-blocking abstraction, $e(p) = V(p) \cap V_{\mathrm{act}}$.	constraint
Minimal generator	An inclusion-minimal member of $\mathcal{H}$; this is *not* a minimal hitting set / transversal.	constraint
Minimal-generator antichain	$\min \mathcal{H}$, the family of all inclusion-minimal generators. By Theorem 2, it labels universal-cost action-equivalence classes exactly.	constraint
Action support	$U = \bigcup_{e \in \min \mathcal{H}} e$, the actionable sites that occur in at least one minimal generator. For a pure node-set restriction, retaining all of U is necessary and sufficient for universal-cost exactness.	decision
Generator coverage	A released set R covers the minimal generators when $R \cap e \neq \varnothing$ for every $e \in \min \mathcal{H}$. This is the condition for at least one safe plan to remain available within R.	evaluation
Action equivalence	$\mathcal{H}_1 \equiv_{\mathrm{act}} \mathcal{H}_2$ when their complete Argmin sets agree for every strictly positive cost vector.	decision
Exact action preservation	Equality of the *complete set-valued* Argmin before and after release, not merely equality of optimal value or recovery of one optimum.	decision
Argmin mismatch	Any inequality between the released and original complete optimal-plan sets at an evaluated cost vector. It is an exactness diagnostic, not by itself a safety diagnosis.	evaluation
Safety failure	The released set omits every available intervention from at least one minimal generator, so no plan using only released sites can hit all original constraints.	evaluation
Cost regret	For a safe restriction, $\Delta_R(c) = \mathrm{OPT}_R(c) - \mathrm{OPT}(c) \geq 0$. Positive regret means the best plan available after release is more expensive than the original optimum.	evaluation
Zero-regret Argmin mismatch	The restricted and original optimal values agree, but their complete optimiser sets differ, for example because one tied optimum was removed.	evaluation
Admissible quotient	A quotient satisfying both generator-signature compatibility and the stated block-constant cost condition of Section 3.3.2.	constraint / decision

Term	Meaning in this paper	Layer / scope
Canonical decision release	$Z_{\min} = \min \mathcal{H}$ for the base universal-cost hitting-set problem; no dependency edges are required by (1).	release
Graph-form operational release	An optional topology-bearing artefact such as G_U or $G_{U/P}$, retained for structural analysis or richer services beyond the base hitting-set input.	release
Structural leakage	Information about nuisance variables determined by graph structure, rather than by redacted identifier or attribute content.	privacy

Table 5 collects the notation and operators together with the objects on which they act.

Table 5: Symbols and operators, with their semantic domains. The domain column is important: graph operators, constraint objects, and decision-set restrictions are not interchangeable.

Symbol	Type / domain	Meaning and object acted on
G^*	typed attributed graph	Private input $(V, E, \tau, \sigma, \mathbf{a}, X)$.
V_{act}	subset of V	Legal intervention sites; ground set of the hitting-set decision.
X	subset of V	Entry points from which exposure paths may start.
L	regular language over edge types	Specifies which typed paths count as exposures.
$\mathcal{H}(G^*)$	hypergraph on V_{act}	Family of actionable generators induced by remediable exposure paths.
$\min \mathcal{H}$	antichain	Inclusion-minimal members of $\mathcal{H}$; canonical universal-cost decision statistic.
U	subset of V_{act}	Support $\bigcup_{e \in \min \mathcal{H}} e$.
$\mathcal{F}(\mathcal{H})$	family of subsets of V_{act}	Plans that hit every generator.
$A(\mathcal{H}, c)$	set of plans	Complete minimum-cost Argmin under strictly positive costs c.
S	nuisance variable	Proprietary information the private room prefers not to disclose.
$Z_{\min}$	hypergraph representation	Canonical base release $\min \mathcal{H}$.
π	decision-set restriction	Restricts the optimisation ground set from V_{act} to U after $\mathcal{H}$ is computed; it is not graph deletion.
P	partition of U	Candidate blocks for an admissible quotient.
γ_P	quotient on support / constraints	Replaces generator-compatible sites by blocks, under the admissible cost schema.
α	graph operator	Redacts attributes $\mathbf{a}$ while leaving graph structure unchanged.
λ, ρ	graph operators	Coarsen semantic labels or edge-type distinctions.
χ	enumeration-graph operator	Safely contracts eligible non-actionable typed chains before exact path enumeration.
δ	released-graph operator	Adds structural noise; not part of the exact decision guarantee.
G_U	induced graph $G^*[U]$	Optional graph-form support topology used in the structural-leakage experiment.
$G_{U/P}$	graph quotient	Optional topology after applying the admissible block partition to G_U.

A.2 Application Class and Representative Instantiations

Structural condition for applicability. The central characterisation in Theorem 2 depends on the induced constraint family, not on how that family was extracted. Formally, let a private structured input g be mapped by some domain-specific procedure Ψ to a finite hypergraph

$$\Psi(g) = \mathcal{H}_g \subseteq 2^{V_{\text{act}}} \setminus \{\emptyset\}.$$

If the public consumer's action is

$$A(\mathcal{H}_g, c) = \operatorname*{arg\,min}_{x \subseteq V_{\text{act}}} \left\{ \sum_{v \in x} c_v : x \cap e \neq \emptyset \text{ for every } e \in \mathcal{H}_g \right\}, \qquad c > 0,$$

then the input is a constraint-generated intervention problem in the sense used here. The map Ψ may be produced by regular-language paths, attack traces, data-flow witnesses, policy violations, or another domain-specific construction. Once $\mathcal{H}_g$ is fixed, Theorem 2 applies without reference to the internal mechanism of Ψ: its second part characterises action equivalence and its third part gives the exact support restriction.

This condition is deliberately narrower than claiming that every planning or remediation problem is a hitting-set problem. The transfer is exact only when each downstream constraint can be represented by the nonempty set of legal interventions that independently satisfy or block that constraint, and when the downstream objective is additive over selected interventions as in (1). Coupled interventions, sequence-dependent costs, capacity constraints, or scheduling constraints require an extended decision model.

Table 6: Representative instantiations of the same constraint-generated intervention pattern. The rows are structural mappings, not additional empirical evaluations. Domain-specific validity depends on whether the stated constraints and interventions faithfully represent the application.

Domain	Structured input and constraint-generating object	Legal interventions	Downstream decision
PQC migration	Software dependency / exposure graph; a path from an entry point to a vulnerable cryptographic use generates a constraint.	Migration, wrapping, or replacement sites.	Minimum-cost migration set hitting every exposure generator.
Attack-path remediation	Enterprise attack graph or reachability model; an attack path or exploit chain reaching a protected asset generates a constraint.	Patch, privilege, filtering, or segmentation sites represented as candidate controls.	Minimum-cost control set blocking every represented attack path.
Sensitive data-flow remediation	Information-flow, provenance, or dependency graph; a disallowed source-to-sink flow or violating trace generates a constraint.	Sanitisation, filtering, declassification, or access-control sites.	Minimum-cost intervention set intersecting every violating flow.
Policy / compliance remediation	Workflow, provenance, or configuration dependency structure; a trace or witness violating a stated policy generates a constraint.	Enforcement or configuration-change sites.	Minimum-cost change set intersecting every represented violation.

What the theory means across these domains. For any valid instantiation above, the general question is not whether the released artefact preserves the private structure itself. It is which distinctions in that structure can change the downstream intervention. Under universal strictly positive costs, the answer for the base hitting-set decision is the inclusion-minimal generator antichain $\min \mathcal{H}_g$: two private inputs are action-equivalent exactly when they induce the same antichain. The support

$$U_g = \bigcup_{e \in \min \mathcal{H}_g} e$$

then identifies the only intervention sites that can appear in an optimum. Thus the same principle applies across domains: release what determines the action-relevant constraints, rather than the full structure that generated them.

Which results transfer directly. The scope is not identical for every result in the paper. The minimal-generator characterisation and exact support restriction in Theorem 2, and the admissible quotient result in Theorem 3, are stated at the hypergraph / decision level and transfer whenever the induced decision has the form above. The pseudonymisation result requires a structured release with a nuisance determined by the retained structure. The typed-chain contraction theorem additionally requires the regular-path and DFA assumptions of Theorem 4. The empirical compression, re-identification, and learned-selector numbers are specific to the PQC and synthetic instances used in this paper and are not asserted for the other rows of Table 6.

Why PQC migration is the measured case. PQC migration gives a concrete instance in which the structured input is commercially sensitive, the constraint-generating paths are operationally

interpretable, candidate interventions have heterogeneous costs, and outsourcing decision support is realistic. This makes it useful for testing how much structure can be removed while preserving an exact downstream decision. We use the other domains only to delimit the broader applicability of the formal result; evaluating them requires domain-specific extraction, threat models, and cost models and is left separate from the empirical claims of this submission.

A.3 Extended Related Work

This appendix expands Section 1. Nothing here is required to follow the results.

Sufficiency and minimality in representation learning. Sufficiency in machine learning descends from the statistical notion. The information bottleneck (Tishby et al., 2000) recast it as a trade-off between compression and retained information about a target, and Achille & Soatto (2018) connected minimal sufficiency to invariance and disentanglement in deep representations. Wu et al. (2020) extended the principle to graphs and Yu et al. (2021) to subgraph recognition. Every one of these defines sufficiency relative to a target that is *predicted*. Our target is the solution of a combinatorial program; Section 3.1 shows the notions separate by $\Omega(n)$ bits against $O(1)$, so a representation minimal in the bottleneck sense can be unboundedly larger than one sufficient for the action. Section 4.2 instead asks whether a learned selector can approximate the exact action support when constructing it explicitly is costly.

Two concurrent works use close terminology in unrelated settings. Hyeon et al. (2026) study action-sufficient goal representations in offline goal-conditioned reinforcement learning, showing that value sufficiency does not imply action sufficiency. This gives a structurally similar separation in a different domain and provides independent evidence that the distinction is consequential. Morris (2026) defines decision-sufficient feature sets for classification via a topological construction. Neither concerns release to a third party, constrained combinatorial decisions, or privacy.

Decision-sufficiency for optimisation. Bennouna et al. (2025) give an exact characterisation of which data determines the optimal decision of a linear program when the cost vector is uncertain, with the span of differences between achievable optima as the minimal sufficient statistic; Ye et al. (2026) study learning such representations, and the term "decision-sufficient representation" is theirs. Huang et al. (2022) use "action-sufficient" for state representations in control. We therefore do not claim the notion as new. The distinction is in which side of the program the private input governs: in all of that work the feasible set is fixed and known, and sufficiency concerns objective-side information. Here the feasible set is generated by the private input and the objective is supplied by the other party, so sufficiency is constraint-side. The mathematical idea is complementary to the objective-side setting; our additional focus is that the constraint-generating structured input is private and explicitly released to another party.

Learning for decisions. Predict-then-optimise (Elmachtoub & Grigas, 2022) and decision-focused learning (Mandi et al., 2024; Wilder et al., 2019; Liu et al., 2024) study pipelines in which a model predicts parameters consumed by an optimiser and trains against decision loss rather than prediction loss; recent work examines solution stability under cost perturbation (Spitzer & Sanson, 2026). Inverse optimisation (Ahuja & Orlin, 2001) studies the dual question, and its geometry is the normal fan: the set of cost vectors sharing an optimal solution is a normal cone. Two differences matter. First, where the input enters: when it enters the objective, action-equivalence is co-location in a normal cone, classical polyhedral geometry for which we claim no novelty; when it enters the constraints, the equivalence classes are a different object. Second, the question asked: predict-then-optimise asks how prediction error propagates to decision quality, whereas we ask what the representation may therefore *omit*, and ask which distinctions may be omitted with zero decision loss, then measure the privacy benefit for specific nuisances.

Query-preserving compression. Fan et al. (2012) compress a graph so that a class of queries returns identical answers, via bisimulation-style equivalences, and a large graph-summarisation literature follows. Prior work preserves answers; we preserve actions. The consequence is that a query-preserving compression is not minimal for a client who only ever acts on the graph, and the separation theorem shows that the gap can be unbounded in representation size.

Hypergraph transversals. Our constraint generators form a hypergraph and the action is a minimum-cost transversal. That non-minimal hyperedges are redundant in any transversal is elementary and long known (Eiter & Gottlob, 1995); we claim no novelty for the redundancy observation. What we contribute is its reading as a minimal sufficient statistic for a family of actions, in particular the necessity direction that quantifies over all strictly positive cost vectors, and the release consequences that follow. We are careful not to conflate two objects: the inclusion-minimal *generator* antichain $\min \mathcal{H}$ is not the family of minimal *transversals*, and the quasi-polynomial dualization results of Fredman & Khachiyan (1996) concern the latter. Pruning inclusion-redundant generators from an enumerated $\mathcal{H}$ is polynomial; the computational difficulty in our setting is enumerating L-exposure paths, of which there may be exponentially many.

Privacy and leakage in graph release. Zhang et al. (2022) demonstrate property inference, subgraph inference and graph reconstruction attacks against GNNs; Nguyen-Cong et al. (2026) show spectral embeddings leak topology and give adaptive reconstruction. These results motivate the structural observation in Section 3.3.1: pseudonymisation acts only on the attribute map and leaves the structure unchanged (or isomorphic under vertex renaming), so it cannot hide a nuisance determined by that structure. The closest work is Indri et al. (2026), who study the trade-off between expressivity and privacy under edge-level differential privacy using homomorphism-density embeddings, and show expressivity can be preserved in expectation under calibrated noise. Our exact trimming addresses a different question: which distinctions are unnecessary for the action before any additional privacy mechanism is applied.

Cryptographic inventory and migration graphs. CycloneDX CBOM (CycloneDX) standardises cryptographic inventory. Graph-based PQC migration and risk analysis already use dependency structure to identify and prioritise migration targets (Hasan et al., 2024; Herrera et al., 2026). We use this domain as an instantiation rather than a contribution; our question is what portion of such a private dependency representation must be released to preserve the downstream optimisation action.

B Additional Theoretical Material

B.1 Separation: Additional Discussion

The main text uses the standard notion of sufficiency relative to a specified consumer: a representation is query-sufficient when all answers in the chosen query family can be recovered, and action-sufficient when the specified downstream action can be recovered. For the separation in Theorem 1, the downstream objective is one fixed strictly positive cost vector.

For $b \in \{0,1\}^n$, the witness uses vertices $\{x_1, \ldots, x_n, h, r\}$, includes $h \to r$ always, and includes $x_i \to h$ exactly when $b_i = 1$. Query q_i asks whether x_i reaches r, so the query-answer vector equals b and distinguishes all 2^n inputs. Under the fixed costs $c_h = 1/2$ and $c_{x_i} = c_r = 1$, the optimal action is $\emptyset$ when $b = 0$ and $\{h\}$ otherwise. The full proof is in Appendix C.1.

The gap is therefore $\Omega(n)$ versus $O(1)$ *bits*: linear and unbounded in description length, or equivalently exponential in the number of distinguishable input classes. The same witness does not give an $O(1)$ representation under universal future costs; in that setting its minimal-generator antichain identifies b, as explained by Theorem 2.

B.2 Service Refinement and Information Leakage

The service hierarchy is illustrative rather than part of the base hitting-set theorem. Fix a decision domain $\mathfrak{G}$, an aligned public-parameter space Θ, and action codomains $\mathcal{Y}_i$, with $A_i : \mathfrak{G} \times \Theta \to \mathcal{Y}_i$. The illustrative service levels are given in Table 7.

The comparison requires a refinement condition: there must be deterministic maps $f_{21} : \mathcal{Y}_2 \to \mathcal{Y}_1$ and $f_{32} : \mathcal{Y}_3 \to \mathcal{Y}_2$ such that, for aligned public parameters,

$$A_1(G;\theta) = f_{21}(A_2(G;\theta)), \qquad A_2(G;\theta) = f_{32}(A_3(G;\theta)).$$

Under this condition, forgetting detail gives the information ordering derived below. The $D_2 \to D_1$ map is immediate when service metadata are aligned: forgetting sites leaves the required replacement

Table 7: Illustrative service levels used in the refinement analysis.

Level	Question answered	Action A_i
D_1	*what* is needed	multiset of required replacement schemes
D_2	*what* and *where*	annotated plan: (site, replacement) pairs
D_3	*what*, *where*, *when*, *how*	ordered plan with modes and compatibility windows

schemes. We do not establish the corresponding D_3 optimisation model, so statements involving D_3 remain conditional on this refinement.

For each service level, define

$$G \sim_i G' \iff A_i(G;\theta) = A_i(G';\theta) \qquad \forall \theta \in \Theta,$$

under the aligned decision domains and public parameters defined above. The equivalence class $Z_i^\star(G) = [G]_{\sim_i}$ is sufficient by construction. It is also the least deterministic sufficient representation under functional refinement: if another sufficient representation T maps G and G' to the same value, their actions must agree for every θ, so $G \sim_i G'$. Thus $Z_i^\star$ is constant on every fibre of T and factors through T.

When the refinement maps above exist, agreement at a richer level implies agreement at every coarser level. Hence

$$\sim_3 \text{ refines } \sim_2 \text{ refines } \sim_1,$$

and there are deterministic maps $Z_3^\star \to Z_2^\star \to Z_1^\star$. For any joint distribution on the private input and nuisance variable, data processing gives

$$I(S; Z_1^\star) \le I(S; Z_2^\star) \le I(S; Z_3^\star).$$

There is also a simple disclosure floor. For any fixed public parameter θ, the returned action is a deterministic function of every sufficient release Z_i, so

$$I(S; Z_i) \ge I(S; A_i(G;\theta)).$$

This is a lower bound, not an equality between release leakage and action leakage. The functional ordering is the deterministic analogue of Blackwell garbling (Blackwell, 1953).

B.3 Release Operators and Their Domains

The release pipeline uses several operators, but they do not all act on the same object. Table 8 summarises their operations and semantic domains. This distinction is important when interpreting what an exact reduction allows us to remove.

Table 8: Release operators, the information they remove or coarsen, and their semantic domains.

Operator	Operation	Removes / coarsens	Domain
α	attribute redaction	attribute content $\mathbf{a}$	graph
λ	label coarsening	granularity of σ	graph
ρ	edge-type collapse	distinctions in τ	graph
π	candidate restriction	sites in $V_{\text{act}} \setminus U$	decision set
γ_P	quotient by P	within-block site identity	support
χ	safe path contraction	intermediate vertices	enumeration graph
δ	structural noise	random edges	released graph

B.4 Exact Preprocessing of Typed Chains

The decision-side reductions developed in the main text apply once the generator hypergraph has been obtained. In graph instantiations, however, enumerating the paths that generate those constraints can be the expensive step. We therefore use one graph-side reduction before enumeration. It must preserve both the path language and the actionable nodes carried by every accepted path.

Contracting an intermediate chain can fail in three ways: it can change the edge-type word; it can remove an actionable intervention site; or it can remove an internal entry point or target where an exposure path could start or stop. The next theorem states the conditions that exclude all three failures.

Let L be recognised by a DFA $\mathcal{D} = (Q, R, \delta, q_0, F)$. For a word $w = r_1 \cdots r_k$, let $T_w(q) = \delta^*(q, w)$ be its transition summary. A contracted macro-edge carries T_w rather than a single edge type.

Theorem 4 (Exact Typed-Chain Contraction)**.** *Let $u \to v_1 \to \cdots \to v_{k-1} \to w$ be a maximal chain in the graph used for exposure-path enumeration. Assume that every internal vertex v_j has in- and out-degree one, $v_j \notin V_{\mathrm{act}}$, $v_j \notin X$, and $\sigma(v_j)$ is not a target label. Replace the chain by a macro-edge carrying the transition summary of its type word.*

Then contraction induces a bijection between exposure paths in the original and contracted graphs, preserving path-language acceptance and the actionable vertices carried by each path. Consequently, for intervention semantics whose generator depends only on those preserved actionable sites, including $e(p) = V(p) \cap V_{\mathrm{act}}$, the contraction preserves $\mathcal{H}$, $\min \mathcal{H}$, and the complete action family for every $c > 0$.

Proof. An exposure path containing no internal chain vertex is unchanged. If it contains an internal v_j, the degree-one condition prevents entry or exit through that vertex, while the entry- and target-exclusion conditions prevent the path from starting or ending there. The path must therefore traverse the whole chain from u to w.

Let ω be the chain's edge-type word and write the path word as $a\,\omega\,b$. The macro-edge applies exactly $T_\omega(q) = \delta^*(q, \omega)$ to every DFA state q, so replacing ω by its transition summary leaves the final automaton state unchanged for every prefix a and suffix b. Acceptance is therefore preserved, and every macro-edge path expands uniquely back through the original chain.

All removed internal vertices lie outside V_{act}, so corresponding paths carry the same actionable vertices. Under the intervention-semantics condition stated in the theorem, corresponding paths therefore induce the same generator. Hence the generator hypergraph and its minimal antichain are unchanged, and Theorem 2 gives equality of the complete action family. □

Each condition has a separate role. An actionable internal node may belong to a generator; an internal entry point or target may begin or end a valid exposure path; and the degree-one condition prevents a path from joining or leaving midway. This preprocessing acts on the graph *before* generator enumeration and is distinct from the decision-set restriction π introduced after $\min \mathcal{H}$ is known.

C Deferred Proofs

C.1 Proof of Theorem 1 – Fixed-Objective Separation

Witness family. For $b \in \{0,1\}^n$, let G_b^* have vertices $V = \{x_1, \ldots, x_n, h, r\}$, exposure set $X = \{x_1, \ldots, x_n\}$, a target label on r, the edge $h \to r$ in every graph, and $x_i \to h$ exactly when $b_i = 1$. Take L so that every $x_i \to h \to r$ is an exposure path when its first edge exists, and let $V_{\mathrm{act}} = V$. Fix

$$c_h^{\mathrm{fix}} = \tfrac{1}{2}, \qquad c_{x_i}^{\mathrm{fix}} = c_r^{\mathrm{fix}} = 1.$$

Query side. Let q_i ask whether x_i reaches r. Then $q_i(G_b^*) = b_i$, so the answer vector is b itself. A $\mathcal{Q}$-sufficient representation must distinguish all 2^n graphs, and any binary encoding therefore needs at least n bits in the worst case.

Action side. The generators are $\mathcal{H}(G_b^*) = \{\{x_i, h, r\} : b_i = 1\}$. If $b = 0$, the hypergraph is empty and strict positivity makes $\emptyset$ the unique optimum. If $b \neq 0$, $\{h\}$ is feasible at cost $1/2$. Any feasible plan not containing h must contain r or every active x_i, and therefore costs at least 1. Hence $\{h\}$ is the unique optimum.

Thus the action has only two values over the entire family. The one-bit representation $\mathbf{1}[b \neq 0]$ is sufficient for this fixed objective, whereas the query representation requires $\Omega(n)$ bits. □

Interpretation. The gap is $\Omega(n)$ versus $O(1)$ in description length, or equivalently 2^n query classes versus two action classes. The witness uses one shared low-cost hub, which is the natural role of a common wrapper or mediation point rather than an artificial cost construction.

C.2 Proof of Theorem 2 – Minimal-Generator Characterisation

Part (i): non-minimal generators are redundant. Every generator contains an inclusion-minimal generator. If a plan hits all minimal generators, then for any $e \in \mathcal{H}$ it hits some $e' \in \min \mathcal{H}$ with $e' \subseteq e$, and therefore also hits e. The reverse implication is immediate. Thus $\mathcal{F}(\mathcal{H}) = \mathcal{F}(\min \mathcal{H})$.

A useful converse. The feasible family determines the minimal-generator antichain. Indeed, $x \in \mathcal{F}(\mathcal{H})$ exactly when $V_{\text{act}} \setminus x$ contains no generator. Hence $\mathcal{F}(\mathcal{H})$ determines the downward-closed family

$$\mathcal{I}(\mathcal{H}) = \{S \subseteq V_{\text{act}} : S \text{ contains no generator}\},$$

and the inclusion-minimal sets outside $\mathcal{I}(\mathcal{H})$ are exactly $\min \mathcal{H}$. Therefore

$$\mathcal{F}(\mathcal{H}_1) = \mathcal{F}(\mathcal{H}_2) \quad \Longleftrightarrow \quad \min \mathcal{H}_1 = \min \mathcal{H}_2.$$

Part (ii): action equivalence. If the minimal antichains agree, part (i) gives identical feasible sets, so the complete Argmin sets agree for every cost vector.

For necessity, suppose the antichains differ. The converse above implies that the feasible families differ; without loss of generality choose $x_0 \in \mathcal{F}(\mathcal{H}_1) \setminus \mathcal{F}(\mathcal{H}_2)$. Set

$$c_v = \begin{cases} \epsilon, & v \in x_0, \\ 1, & v \notin x_0, \end{cases} \qquad 0 < \epsilon < 1/|V_{\text{act}}|.$$

Since x_0 is feasible for $\mathcal{H}_1$, $\mathrm{OPT}_1 \leq \epsilon |x_0| < 1$. Feasibility is upward closed; because x_0 is not feasible for $\mathcal{H}_2$, no subset of x_0 is feasible for $\mathcal{H}_2$. Every feasible plan for $\mathcal{H}_2$ therefore contains a node outside x_0 and costs at least 1. Hence $\mathrm{OPT}_2 \geq 1 > \mathrm{OPT}_1$, so the action sets differ for this strictly positive cost vector.

Part (iii): exact support. Let $U = \bigcup_{e \in \min \mathcal{H}} e$. If an optimal plan x contains $v \notin U$, then v occurs in no minimal generator. Removing v leaves every minimal generator hit and strictly lowers the cost, contradicting optimality. Thus every optimum is contained in U. Consequently any restriction R with $U \subseteq R$ preserves the complete Argmin set for every $c > 0$.

For the converse, suppose $u \in U \setminus R$. Restrict each minimal generator to the available candidates, keeping an empty intersection if one occurs. If some generator becomes empty, the restricted problem is infeasible and cannot preserve the original action. Otherwise the restricted minimal-generator family differs from the original one: a minimal generator containing u has been replaced by a strict subset, and no original minimal generator can be a strict subset of it. By part (ii), the original and restricted problems therefore have different complete Argmin sets for some strictly positive cost vector. Hence universal-cost exactness requires $U \subseteq R$. □

Computational checks. The proof above is analytic; the implementation also checks the identities by brute force on small random hypergraphs. We use those checks as regression tests rather than as evidence for the theorem.

C.3 Proof of Theorem 3 – Exact Admissible Quotient

Let $B \in P$ be a block. Signature compatibility implies an atomicity property that we use directly: if one $v \in B$ belongs to a minimal generator e, then every $u \in B$ has the same signature and also belongs to e. Thus every block is either contained in or disjoint from each minimal generator.

Next consider an optimal original plan x. It cannot contain two distinct nodes u, v from the same block. Any minimal generator hit by v contains the whole block and is therefore also hit by u. Removing v keeps feasibility and, because $c_v > 0$, strictly lowers the cost. Thus every optimum uses at most one representative per selected block. Notice that this step needs structural compatibility and strict positivity; block-constant costs are needed later to preserve objective values under lifting.

For such a plan define its block image $\bar{x} = \{B \in P : x \cap B \neq \emptyset\}$. By atomicity, x meets an original minimal generator e exactly when $\bar{x}$ meets its quotient generator $\bar{e}$. Because costs are constant inside each block,

$$c(x) = \sum_{B \in \bar{x}} \bar{c}_B = \bar{c}(\bar{x}).$$

Hence the block map preserves feasibility and cost, while its fibre over a quotient plan Y is exactly $\mathrm{Lift}_P(Y)$. Optimality therefore holds in both directions, proving Theorem 3.

Why both assumptions are present. We tested the failure modes before fixing the statement. Zero-cost nodes allow redundant representatives to remain in the complete Argmin set. Equal generator incidence with unequal costs makes representative choice affect the objective. Equal costs alone are also insufficient: for $\mathcal{H} = \{\{u\}, \{v\}\}$, merging u and v makes two required choices look like one. Redundant raw generators such as $\{\{u\}, \{u, v\}\}$ also show why signatures must be based on $\min \mathcal{H}$, not the unreduced family. The implementation additionally checked the quotient identity on 4416 admissible (hypergraph, partition, cost) triples with no mismatch; Section 4 repeats the check on hypergraphs produced by real repositories.

C.4 Why we do not claim a coarsest release

An earlier version considered a coarsestness claim for a composition such as $\alpha \circ \gamma_P \circ \pi$. That claim was invalid: it used raw generator incidence rather than $\min \mathcal{H}$ and relied on the false implication $\mathcal{H}_1 \neq \mathcal{H}_2 \Rightarrow \mathcal{F}_1 \neq \mathcal{F}_2$, refuted by $\{\{u\}, \{u, v\}\}$ versus $\{\{u\}\}$. We therefore make the narrower statement actually proved in this paper: Theorem 2 identifies the least deterministic statistic for the base action equivalence classes, while Theorem 3 gives exact sufficient conditions for a particular quotient. We do not claim that the operator composition itself is coarsest or least among all possible release mechanisms.

D Real-Code Experimental Details

D.1 Corpus and Construction

We build static in-repository import graphs for 52 open-source repositories across Go, Java, Python, and JavaScript. Nodes are source files in Python, Java, and JavaScript, and Go packages in Go; a directed edge $u \to v$ indicates that u statically imports v. Thus the real-code instantiation uses a single edge relation, $R = \{\texttt{import}\}$, with $L = R^*$.

A source unit is labelled as a cryptographic target when its source text matches a name associated with a Shor-vulnerable cryptographic scheme. This detector is intentionally lightweight and may capture textual references rather than executable cryptographic calls. Entry points are identified using language-specific patterns, with in-degree-zero nodes used as a fallback when no explicit entry point is detected.

Of the 52 repositories, **34 are analysable**: 10 contain no detected cryptographic target and 8 contain no detected exposure path. Two of the latter eight are security frameworks (`spring-security`, `shiro`), where the failure is most plausibly caused by the entry-point heuristic on library-shaped code rather than by an absence of security-relevant functionality. We report this skip rate because it bounds what the corpus can support.

An empirical exposure path starts at an entry node and terminates at the first crypto-labelled target encountered along a directed import path. Under the path-blocking abstraction used in this corpus, every actionable node on such a path is treated as an alternative atomic blocking intervention.

Actionable sites are a modelling choice, so we vary them. V_{act} is not given by the extractor. We report three operationalisations. STRICT makes only crypto-labelled target nodes actionable. MEDIATED, our default, additionally includes their direct importers, modelling an intervention that may be placed immediately before the vulnerable boundary. OPEN makes every node on an exposure path actionable. These settings vary where the path-blocking abstraction permits an intervention; they do not claim that an arbitrary source-code edit at each such node constitutes a complete PQC migration.

Exact enumeration, and what it costs. For the untyped $L = R^*$ corpus graphs, we first eliminate non-terminal nodes by reachability-preserving contraction. We then enumerate generators exactly, using pruning rules that preserve the inclusion-minimal generator family: enumeration stops when the first crypto-labelled target is reached, and a search prefix is abandoned when its actionable set already contains a known generator. The resulting family is reduced to its inclusion-minimal antichain $\min \mathcal{H}$.

This corpus-specific contraction exploits the untyped $L = R^*$ reachability setting and is not an instance of the typed-chain contraction theorem in Theorem 4.

Repositories exceeding a hard enumeration budget (2×10^6 search steps or $150\,\mathrm{s}$) are **excluded rather than approximated**. Under STRICT, none are excluded; under MEDIATED, four are excluded (`keycloak`, `matrix-js-sdk`, `salt`, and `teleport`); under OPEN, seven are excluded. Exact generator enumeration is therefore the principal computational bottleneck of the real-code pipeline.

D.2 CORPUS STATISTICS

Repository-level statistics for the 34 analysable codebases are reported in Table 9.

Table 9: The 34 repositories entering the real-code exposure analysis. $|V|$ and $|E|$ are the numbers of nodes and static in-repository import edges, respectively; $|V_{\text{crypto}}|$ is the number of crypto-labelled target nodes. $|U|$ is the exact action-support size and $|U/P|$ is the number of structural quotient blocks under the MEDIATED actionability model. *cap* denotes an exact-enumeration budget exceedance; such instances are excluded from analyses requiring the exact generator hypergraph rather than being approximated.

Repository	Lang	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V_{\text{crypto}}\rvert$	$\lvert U\rvert$	$\lvert U/P\rvert$
ansible	python	596	2465	3	3	3
caddy	go	47	199	4	26	26
certbot	python	220	421	5	6	4
certificates	go	55	167	13	11	11
cfssl	go	92	351	15	46	46
consul	go	390	1426	11	37	36
coredns	go	104	441	2	3	3
cryptography	python	108	164	1	1	1
dex	go	77	234	2	4	4
forge	js	55	215	1	1	1
gitea	go	347	2868	11	12	12
jose	js	86	44	32	59	59
keycloak	java	5777	27404	38	*cap*	*cap*
loki	go	407	1822	1	1	1
matrix-js-sdk	js	407	1971	24	*cap*	*cap*
minio	go	73	182	4	1	1
netty	java	2237	7221	2	3	3
nifi	java	5727	16978	1	3	3
node-jsonwebtoken	js	12	20	2	2	2
node-oidc-provider	js	200	614	5	9	7
nodemailer	js	38	0	2	2	2
nomad	go	249	886	8	12	8
openpgpjs	js	99	320	36	45	44
paramiko	python	44	172	11	10	10
rclone	go	298	1897	7	57	57
salt	python	1019	3687	11	*cap*	*cap*
sentry	python	4900	31217	2	7	6
syncthing	go	90	310	3	10	10
telegraf	go	572	1833	1	8	8
teleport	go	1013	6708	66	*cap*	*cap*
tomcat	java	1771	5747	6	9	9
traefik	go	155	567	5	12	11
twisted	python	402	1230	8	8	5
vault	go	340	1036	19	32	25

The remaining 18 of the 52 cloned repositories do not enter the real-code exposure corpus: 10 contain no detected cryptographic target (activemq, django, dropwizard, jupyterhub, passport, requests, restic, scrapy, session, sshuttle), and 8 contain no detected exposure path (borg, etcd, jetty.project, mitmproxy, pretix, shiro, spring-security, zookeeper). The corpus membership was fixed before computing the reported compression and re-identification outcomes; no repository was removed on the basis of those outcomes.

D.3 Theorem Checks on the Real Corpus

Theorem 3 is proved analytically and independently checked on synthetic instances in Appendix C.3. We additionally validate it on the hypergraphs produced by the real-code corpus. For every repository–actionability pair with $|U| \leq 16$, we draw 20 random strictly positive cost vectors that are constant

within each admissible block, and brute-force both

$$A(\mathcal{H}, c) \quad \text{and} \quad \bigcup_{Y \in A(\mathcal{H}/P, \bar{c})} \mathrm{Lift}_P(Y).$$

This covers 69 corpus-derived hypergraphs: 29 under STRICT, 23 under MEDIATED, and 17 under OPEN. Across the resulting **1,380 cost-vector checks** (580 STRICT, 460 MEDIATED, and 340 OPEN), we observe no mismatch.

When the optimisation is vacuous. Under STRICT actionability, all 274 minimal generators observed in the corpus are singletons. The path-blocking plan is therefore uniquely determined and independent of the cost vector. Under MEDIATED, the median generator has size 2, and only 64 of 428 minimal generators are singletons. The optimisation therefore becomes non-trivial once the model permits alternative blocking sites in addition to the crypto-labelled target. This behaviour is specific to the empirical intervention model and is not required by the general theory.

D.4 Edge-Ablation Robustness

Because the real-code study uses static import graphs as a lightweight structural abstraction, we test whether the main conclusions are robust to perturbations of those graphs. We delete a random fraction of import edges and recompute the full pipeline, including the exposure subgraph, exact generator enumeration, $\min \mathcal{H}$, and U, then compare the ranking of repositories by released fraction $|U|/|V(G^*)|$ with the ranking obtained from the unperturbed graphs.

Under 25% edge deletion, Spearman $\rho = 0.983$ ($n = 30$, $p = 3.4 \times 10^{-22}$); under 50% deletion, $\rho = 0.951$ ($n = 28$, $p = 8.4 \times 10^{-15}$). Repository–perturbation instances whose exact enumeration exceeds the budget are excluded from the comparison, which is why $n < 34$.

Absolute compression is not expected to remain invariant under this perturbation: deleting import edges can remove exposure paths and therefore shrink U. The robustness result concerns the *relative ordering* of repositories by released fraction, which remains stable under substantial edge deletion.

D.5 Structural Fingerprint

The re-identification experiment in Section 4.1 uses the optional graph-form releases $\alpha(G^*)$, G_U, and $G_{U/P}$. Its fingerprint uses predominantly size-normalised structural statistics: mean in-, out-, and total degree; the standard deviation and skew of each degree distribution; the fraction of nodes with zero in- or out-degree; maximum degree normalised by $|V|$; the numbers of weakly and strongly connected components normalised by $|V|$; the largest weak and strong components as fractions of $|V|$; average clustering coefficient; degree assortativity; triangle count normalised by $|V|$; and the five largest eigenvalues of the normalised Laplacian of the undirected graph.

The explicit graph-size features $\log|V|$ and $\log|E|$ are computed but **excluded** from all headline results. For tractability, however, the spectral and triangle feature blocks are disabled above implementation-specific graph-size thresholds; the fingerprint should therefore be interpreted as predominantly size-normalised rather than strictly size-free. Matching is performed by nearest neighbour in the z-scored feature space, with reference statistics computed on the drifted graphs so that the adversary is not given the target-side normalisation.

D.6 Reproduction

The reported numerical results are produced by deterministic graph-construction, exact-enumeration, theorem-checking, attack, and learning pipelines. The accompanying source package records the theorem-checking scripts and archived JSON outputs used for the reported quotient, robustness, and learning results.

The real-code corpus used for the archived run was obtained by shallow-cloning the then-current repository heads. The commit hashes were not retained after the local clones were removed. Consequently, the archived outputs preserve the measurements reported in this paper, but reconstructing the corpus from current repository heads need not reproduce the identical graphs. We therefore treat this

as a reproducibility limitation of the present real-code study; future reruns should pin and record the repository commit used for every corpus member.

End-to-end reproduction of the synthetic learning study additionally requires the corresponding dataset-generation and model-training code, which is included separately from the manuscript sources.

E LEARNED-REPRESENTATION EXPERIMENT DETAILS

Instances and exact ground truth. Learning requires substantially more labelled graphs than the real-code corpus provides, while the evaluation requires exact decision ground truth. We therefore generate typed layered DAGs with 14–30 nodes, six edge types, entry nodes, target nodes, an actionable set V_{act}, and the exposure language

$$L = \{w : w \text{ contains no } \texttt{test} \text{ edge and at least one } \texttt{flows} \text{ edge}\}.$$

For every retained instance we enumerate the exposure paths and compute exactly

$$\mathcal{H}, \qquad \min \mathcal{H}, \qquad U = \bigcup_{e \in \min \mathcal{H}} e.$$

The retained synthetic population satisfies

$$4 \le |U| \le 12.$$

The generator and corpus are frozen before model training.

We use 1200 instances from `corpus_0` for training, the next 400 instances from the same corpus for validation, and 400 disjoint instances from `corpus_1` for a single held-out test evaluation. A further 400 instances from `corpus_1` are kept untouched as reserve data. The validation set is used only for checkpoint selection; the test set is not accessed during training or model selection.

Selector inputs and features. At inference time, learned selectors receive only the graph, typed edges, entry and target indicators, and the actionable set. They do not receive $\mathcal{H}$, $\min \mathcal{H}$, U, or any quantity derived from exact generator enumeration.

The full node feature vector contains total in/out degree, entry, target, and actionability indicators, typed in/out degree counts for each of the six edge types, and an L-agnostic reachability indicator V_π. The latter is obtained by intersecting nodes reachable from an entry with nodes that can reach a target, ignoring edge types and the language L. It is therefore computable directly from the graph by graph search and is not an exact-support feature. Here V_π denotes only this reachability feature and is unrelated to the decision-set restriction operator π of Section 3.2. Because V_π is nevertheless a strong structural prior, we repeat the learned experiments after removing it.

Selectors. We compare three primary selectors under the same release budget. RANDOM draws a uniform subset of V_{act} without replacement. MLP-SUP applies a two-layer node-local MLP to the node features, whereas GNN-SUP uses the same initial features followed by four layers of typed message passing. Both learned selectors are supervised directly on the exact support target

$$y_v = \mathbf{1}[v \in U].$$

Thus the MLP–GNN comparison isolates the additional predictive value of graph context; the experiment is not intended as an architecture benchmark.

For a requested fixed budget k, all selectors use

$$k_{\text{eff}} = \min(k, |V_{\text{act}}|),$$

so instances with fewer than k actionable nodes release the full actionable set.

Random selection is averaged over 30 independent draws. Learned models are trained with three seeds. Training uses binary cross-entropy over actionable nodes only, with class reweighting by the negative-to-positive ratio. We use Adam with learning rate 2×10^{-3}, hidden width 64, gradient accumulation over 16 instances, and at most 20 epochs. For each seed, the checkpoint is selected using only the validation set: highest exact-U recovery at matched budget, then lowest validation loss, then the earliest epoch.

Budget regimes. For a selector producing $R_k \subseteq V_{\text{act}}$, universal preservation under a size-k release is possible only when $|U| \leq k$. We therefore evaluate fixed budgets

$$k \in \{4, 5, 6, 7, 8, 10, 12\}$$

and report

$$C(k) = \Pr(|U| \leq k),$$

$$S(k) = \Pr(U \subseteq R_k),$$

and

$$Q(k) = \Pr(U \subseteq R_k \mid |U| \leq k).$$

Here $C(k)$ is an oracle capacity ceiling rather than a model. $S(k)$ measures end-to-end universal-support recovery, whereas $Q(k)$ conditions out instances on which the budget is too small to contain U.

We additionally use a matched-budget evaluation

$$k_i = |U_i|.$$

Because $|R_i| = |U_i|$ in this setting,

$$U_i \subseteq R_i \qquad \Longleftrightarrow \qquad R_i = U_i.$$

Matched-budget exact recovery therefore measures selector error without conflating it with insufficient release capacity.

Decision evaluation. For a released candidate set $R \subseteq V_{\text{act}}$, define

$$\mathcal{F}_R(\mathcal{H}) = \{x \subseteq R : x \cap e \neq \varnothing \text{ for every } e \in \min \mathcal{H}\},$$

and

$$A_R(\mathcal{H}, c) = \operatorname*{arg\,min}_{x \in \mathcal{F}_R(\mathcal{H})} c(x).$$

For pure candidate restriction, the universal structural criterion is

$$A_R(\mathcal{H}, c) = A(\mathcal{H}, c) \quad \forall c > 0 \qquad \Longleftrightarrow \qquad U \subseteq R.$$

Thus sampled objectives are not used to establish universal sufficiency.

Instead, when $U \not\subseteq R$, we use 32 independently sampled strictly positive cost vectors per test instance to characterise the downstream consequence. Each (instance, cost) pair is assigned to exactly one of four categories:

1. *infeasible*: some $e \in \min \mathcal{H}$ satisfies $e \cap R = \varnothing$;
2. *positive regret*: the restricted problem is feasible but $\text{OPT}_R(c) > \text{OPT}(c)$;
3. *zero-regret Argmin mismatch*: the optimum value is unchanged but $A_R(\mathcal{H}, c) \neq A(\mathcal{H}, c)$;
4. *exact*: the complete Argmin sets agree.

For positive-regret cases we report relative regret

$$\frac{\text{OPT}_R(c) - \text{OPT}(c)}{\text{OPT}(c)}.$$

Support metrics are computed per instance, whereas decision outcomes are computed per (instance, cost) pair.

V_π ablation. Figure 3 compares the full feature set with the same models after removing V_π. At the matched budget, exact-U recovery for MLP-SUP decreases from $24.1 \pm 1.7\%$ to $12.8 \pm 1.6\%$, whereas GNN-SUP changes only from $37.0 \pm 2.4\%$ to $36.2 \pm 0.3\%$. The result shows that the graph-aware selector is much less sensitive to the explicit reachability prior; it does not imply that the GNN reconstructs V_π exactly.

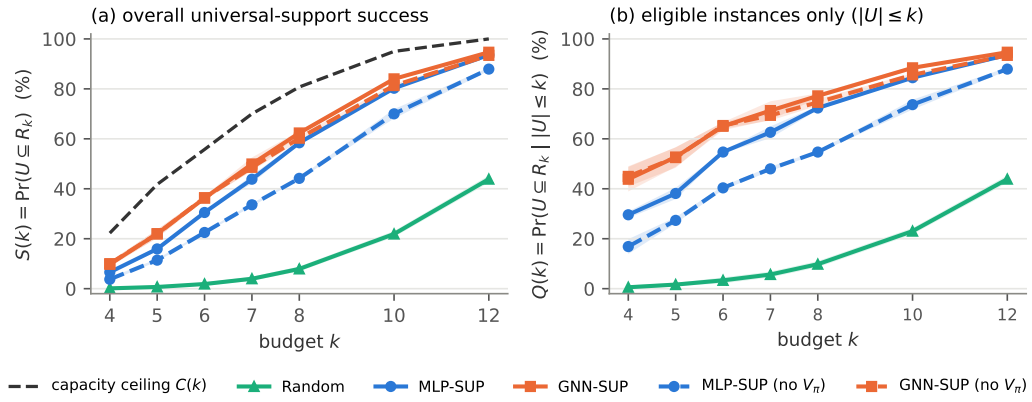

Figure 3: Ablation of the L-agnostic reachability feature V_π. Panel (a) reports overall universal-support success $S(k)$, and panel (b) reports $Q(k)$ over the same capacity-eligible subsets, $|U| \leq k$, as in Figure 2. Removing V_π substantially reduces MLP performance, while GNN performance changes comparatively little.

Task-driven auxiliary baseline. For completeness, we also train GNN-ACT, which uses the same GNN architecture but targets node membership in sampled optimal plans rather than membership in U. Its target for node v is the fraction of 16 sampled training costs under which v belongs to at least one optimal plan. We keep this method in the appendix because its target is cost-distribution dependent and therefore does not directly represent universal action support. Its checkpoint is selected by the same validation exact-U criterion used for the main learned selectors.

At matched budget, GNN-ACT recovers U exactly on $32.4 \pm 1.4\%$ of instances, with mean support recall 84.7%. It preserves the exact Argmin on $91.3 \pm 0.1\%$ of the sampled test (instance, cost) pairs, compared with $89.8 \pm 0.5\%$ for GNN-SUP. This sampled-cost advantage does not imply universal sufficiency: only $U \subseteq R$ provides the guarantee for every strictly positive cost.

Remark 5 (Scope)**.** The learned-representation experiment is deliberately synthetic and small so that $\mathcal{H}$, $\min \mathcal{H}$, U, and complete optimal-action sets can be computed exactly. Only one MLP and one representative GNN are evaluated, using three training seeds and fixed hyperparameters. Decision outcomes are estimated from 32 sampled continuous cost vectors per test instance. No runtime comparison between exact extraction and learned inference is performed, so we make no speedup claim. The numerical rates should therefore be interpreted as a controlled study of approximation failure modes, not as evidence that learning is necessary or as a direct prediction of repository-scale performance.

F Extended Limitations

The real-code graph is a lightweight structural abstraction. The real-code study uses static in-repository import graphs rather than sound whole-program call graphs. A directed edge therefore records a static import relation, not evidence that a particular cryptographic operation is executed along that dependency. Conversely, dynamic dispatch, reflection, dependency injection, unresolved imports, and other runtime behaviour may be absent from the extracted graph. The reported compression figures should therefore be interpreted as properties of this structural abstraction rather than estimates for a sound call graph. As a robustness check, repository rankings by released fraction $|U|/|V(G^*)|$ remain stable under substantial random edge deletion: Spearman $\rho = 0.983$ ($n = 30$) at 25% deletion and $\rho = 0.951$ ($n = 28$) at 50% deletion. This result concerns ranking stability under perturbation; it does not establish semantic soundness of the underlying program graph.

Cryptographic-target detection is lightweight and non-semantic. A source unit is labelled as a cryptographic target when its source text matches names associated with Shor-vulnerable cryptographic schemes. This procedure can produce false positives from textual references, including comments or strings, and can also miss configuration-driven or dynamically selected cryptographic

behaviour. Eight of the 52 repositories contain no detected exposure path. In at least two library-shaped repositories, the entry-point heuristic is the most plausible explanation, illustrating that extraction failures can arise from graph construction as well as from target detection.

The intervention model is deliberately simplified. In the real-code study, every actionable node on an exposure path is treated as an alternative atomic site at which that exposure may be blocked. Thus $e(p) = V(p) \cap V_{\mathrm{act}}$ is an empirical path-blocking abstraction, not a claim that an arbitrary edit at a selected source unit constitutes a complete PQC migration. The model also does not represent coordinated remediations that require several code changes jointly. Exposure paths terminate at the first crypto-labelled target, so the experiment captures exposure to the first detected cryptographic boundary rather than every downstream cryptographic use.

Program granularity may affect the induced decision structure. The extracted nodes are source files in Python, Java, and JavaScript and package directories in Go. This is substantially coarser than a function-, call-site-, or data-flow-level program representation. A finer graph could change both the exposure paths and the induced generator hypergraph, potentially altering U and the admissible quotient in either direction. The reported measurements therefore apply to the source-unit granularity used in this study; we do not infer how much compression would be obtained at finer program granularity.

Closed-world re-identification. The approximately $19\times$-chance re-identification result in Section 4.1 is measured in a closed-world setting with 24 candidate repositories. An open-world adversary facing a much larger candidate population poses a different identification problem. The experiment therefore demonstrates that pseudonymisation alone can leave substantial structural distinguishability; it should not be interpreted as an estimate of deployment-time re-identification probability.

Graph-form support projection does not eliminate structural disclosure. For the optional graph-form release G_U, re-identification remains at approximately 17–21% (4–5$\times$ chance). This leakage concerns the retained topology. The canonical decision release for the base universal-cost hitting-set problem is instead $\min \mathcal{H}$, which does not require releasing dependency edges. Privacy guarantees for richer graph-form releases therefore require mechanisms beyond the exact decision-sufficiency results developed here.

Scope of the formalism and of the empirical instantiation. The theoretical characterisation applies to any constraint-generated intervention problem whose downstream action is a minimum-cost hitting set over a finite family of nonempty generators. The PQC-motivated real-code study instantiates that framework using path-generated constraints for cryptographic exposure. Other migration tasks may induce different constraint mechanisms and intervention semantics. In particular, cryptographic changes that are local to individual occurrences, stateful across several program elements, or governed by compatibility, scheduling, hardware, or protocol-level constraints are not captured by the present real-code path-blocking model.

Cost-class dependence. Theorem 2 characterises action equivalence when the downstream cost vector may be any strictly positive vector. If the admissible costs belong to a restricted class $\mathcal{C} \subsetneq \mathbb{R}_{>0}^{V_{\mathrm{act}}}$, distinct minimal-generator antichains may become action-equivalent over that class, permitting a coarser sufficient release. Characterising the minimal statistic for structured cost classes is left for future work.

No certified approximation guarantee. Our main results concern exact preservation: the minimal-generator characterisation, candidate-support restriction, and admissible quotient preserve the relevant decision object under their stated assumptions. We do not provide a certified privacy–utility bound when those requirements are relaxed. Developing approximate action-sufficient releases with explicit guarantees on decision degradation or leakage remains an open direction.

Exact generator enumeration can exceed the computational budget. Under the default MEDIATED actionability model, four repositories exceed the fixed exact-enumeration budget and are excluded from analyses that require the resulting generator hypergraph. We do not replace them

with sampled generators, because an omitted generator can make two nodes appear structurally interchangeable and thereby invalidate the quotient condition. Candidate restriction admits a conservative alternative: any verified superset $U' \supseteq U$ retains every original optimum, although it may disclose more than the exact support.

Corpus reconstruction is not bitwise reproducible. The archived real-code run was produced from shallow clones of then-current repository heads, but the corresponding commit hashes were not retained after the local clones were removed. The archived outputs preserve the measurements reported in this paper, whereas rebuilding the corpus from current repository heads may produce different graphs. Future corpus releases should pin and record the exact commit of every repository.

F.1 The Admissible Cost Class

Remark 6 (The admissible cost class matters)**.** Restricting costs to $\mathcal{C} \subset \mathbb{R}_{>0}^{V_{\mathrm{act}}}$ can make the minimal sufficient statistic coarser: 165 of 399 distinct-antichain pairs are not separated by the all-ones cost vector. Thus disclosure depends on the permitted cost class.

Why universal positive costs appear in the base theorem. In our two-room setting, downstream costs may arrive only after the private room has committed to a release. Requiring exactness for every strictly positive cost vector therefore captures the case in which the release cannot be tailored to a particular future objective. Theorem 2 characterises the corresponding universal-cost decision statistic.