SPECTRAL GRAPH FEATURES FOR LEARNED HEURISTICS IN AUTOMATED PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A common approach in learning for automated planning is to train a domain-dependent heuristic on small problems and apply it to larger ones. Recent work showed that 1-Weisfeiler–Leman (1-WL) can extract effective features from planning states, outperforming deep-learning approaches while reducing training time to seconds. This result motivates the study of alternative graph-based features. Spectral graph methods are a natural candidate because they are well studied and can capture global structure that finite-round 1-WL may miss. We present the first study of spectral features for learning heuristic functions in classical planning. Our results show that spectral features outperform 1-WL in multiple domains, achieving SOTA performance.

1 INTRODUCTION

Automated planning considers the problem of finding a sequence of actions that moves from an initial state to a goal state (Geffner & Bonet, 2013). We focus on classical planning problems, which are fully observable, deterministic, and have finite state spaces. In our setting, an individual problem is defined as an instantiation of a planning domain: the domain specifies a common structure and a set of parameterised actions, and the problem instances specify objects, initial states, and goals within their domain. Classical planners have enjoyed success with heuristic search algorithms (Geffner & Bonet, 2013; Taitler et al., 2024), but in general the classical planning problem is PSPACE-complete (Bylander, 1994), and no single heuristic works equally well in every domain. This has led to work on learning heuristic functions for a particular domain.

Due to the success of deep learning, an adaptation to classical planning was inevitable, resulting in Action Schema Networks, Graph Neural Network (GNN) policies, and transformer-based models (Toyer et al., 2018; Ståhlberg et al., 2022; Rossetti et al., 2024). A turning point was *Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning*, Chen et al. (2024b). Their WL-GOOSE system represents planning states as graphs, uses 1-Weisfeiler–Leman (1-WL) refinement to compute features, and fits a small classical regression model. Compared to Deep Learning models, their approach performs better on typical benchmarks, and takes only seconds to train, rather than hours. Their feature family has since been extended to planning with numeric state variables and combined with new data-generation and feature-selection methods (Chen & Thiébaux, 2024; Hao et al., 2025).

Fundamentally, 1-WL attempts to distinguish graphs by summarising local neighbourhoods, but inevitably some graphs cannot be distinguished because they produce the same 1-WL features. An alternative approach is with spectral methods, which describe a graph through the eigenvalues and eigenspaces of matrices; different matrix representations are used, such as the adjacency matrix or graph Laplacian (Chung, 1997). These methods are also invariant to object renaming, but their distinguishing power is different. Some graphs in which every vertex has the same degree receive identical 1-WL features but have different eigenvalues. Conversely, some graphs with identical eigenvalue lists receive different 1-WL features (Rattan & Seppelt, 2023). Spectral features can also have a fixed dimension for a domain, whereas the set of discrete 1-WL colours is learned from the training graphs.

We present the first controlled study that learns classical-planning heuristics from fixed-dimensional spectral descriptors of attributed planning-state graphs and compares them directly with 1-WL. In our setting, we keep the planning graphs, training states, regression models, and search procedure

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.

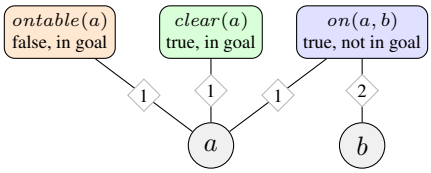

Figure 1: An attributed Instance-Learning Graph for a Blocksworld state. Circles denote objects and boxes denote atoms. Atom colours encode the predicate, truth value, and goal status, while edge labels encode argument positions. Object names and printed atom text are explanatory only.

fixed and evaluate only the impact of graph features. This isolates what spectral features retain, when their invariances hold, how they transfer to larger graphs, and whether their evaluation cost pays off in search. The strongest spectral configuration closely competes with 1-WL, beating it in some domains while losing in others. These complementary strengths suggest that spectral and 1-WL features capture different aspects of planning-state graphs, motivating their combination in learned planning heuristics.

2 Background

Classical planning. We consider finite, deterministic STRIPS planning with unit-cost actions (Geffner & Bonet, 2013). A planning domain $D = \langle \mathcal{P}, \mathcal{A} \rangle$ consists of predicate symbols $\mathcal{P}$ and parameterised action schemas $\mathcal{A}$. An atom such as $on(x, y)$ is *lifted* if it contains variables and *ground* if all arguments are objects, as in $on(a, b)$.

An action schema $\alpha = \langle X_\alpha, \mathrm{pre}(\alpha), \mathrm{add}(\alpha), \mathrm{del}(\alpha) \rangle$ contains parameters X_α and sets of lifted precondition, add, and delete atoms. We assume positive preconditions and disjoint add and delete effects. Types restrict which objects may instantiate parameters and are omitted from the notation.

For example, in Blocksworld (Slaney & Thiébaux, 2001), blocks must be rearranged using a hand that holds at most one block. A state is described by a set of atoms, such as $on(x, y)$, $clear(x)$, $handempty$, and $holding(x)$. The action $stack(x, y)$ requires the hand to hold block x and block y to be clear. Applying it places x on y, makes x clear and the hand empty, and makes y no longer clear; formally, it adds $on(x, y)$, $clear(x)$, and $handempty$, and deletes $holding(x)$ and $clear(y)$.

A problem instance $I = \langle O, s_0, g \rangle$ supplies a finite object set O, initial state s_0, and goal g. Let $\mathcal{F}_D(O)$ denote the well-typed ground atoms over O, with $s_0, g \subseteq \mathcal{F}_D(O)$. A state $s \subseteq \mathcal{F}_D(O)$ contains exactly the atoms that are true. Grounding an action schema with compatible objects produces a ground action $a = \langle \mathrm{pre}(a), \mathrm{add}(a), \mathrm{del}(a) \rangle$. Let A_I be the set of all ground actions. The actions applicable in state s are $A(s) = \{a \in A_I \mid \mathrm{pre}(a) \subseteq s\}$, and applying $a \in A(s)$ yields

$$\mathrm{succ}(s, a) = \big(s \setminus \mathrm{del}(a)\big) \cup \mathrm{add}(a).$$

A state is a goal state if $g \subseteq s$. A plan from s is a sequence of applicable actions leading to a goal state. Since all actions have unit cost, the optimal goal distance $h^*(s)$ is the length of a shortest plan from s, with $h^*(s) = \infty$ if no such plan exists.

Our setting requires *structural generalisation*: models are trained on states from small instances and applied, without retraining, to larger instances of the same domain (Jiménez et al., 2019). As instance size grows, the state space and graph representations typically grow as well; transferable features must therefore handle variable-sized graphs, be invariant to object names, and preserve relational structure relevant to the optimal goal distance.

Attributed Instance-Learning Graphs (ILGs). Following WL-GOOSE, we encode a state s and its goal as an instance-learning graph $G_s = (V, E, \chi, r)$, abbreviated ILG. The graph is bipartite, with object vertices and vertices representing ground atoms. The colour $\chi(v)$ of an atom vertex records its predicate and whether the atom is (i) true but not a goal, (ii) true and a goal, or (iii) false and a goal; all object vertices share one colour. Each edge connects an atom to one of its arguments, and its label $r(e)$ records the argument position. Thus, the two arguments of $on(a, b)$ remain distinguishable.

Figure 1 illustrates the construction. The state contains $on(a, b)$ and $clear(a)$, while the goal requires $clear(a)$ and $ontable(a)$. The ILG contains one vertex for each object and for each atom that is true or appears in the goal.

Weisfeiler–Leman (WL) features. The 1-WL algorithm iteratively refines the node colours of an ILG. Let $c_v^{(0)} = \chi(v)$ be the initial colour of vertex v. Following WL-GOOSE, edge labels are incorporated by defining

$$c_v^{(t+1)} = \mathrm{HASH}\Big(c_v^{(t)}, \Big\{\!\!\Big\{ \big(r(\{u, v\}), c_u^{(t)}\big) : u \in N(v) \Big\}\!\!\Big\}\Big), \tag{1}$$

where $\{\!\{\cdot\}\!\}$ represents a multiset, and HASH injectively maps distinct inputs to discrete colours (Chen et al., 2024b). Thus, two vertices receive the same new colour exactly when their previous colours and multisets of labelled neighbour colours agree. Different colour histograms at any round certify that two graphs are non-isomorphic, whereas equal histograms do not imply isomorphism. Under standard assumptions, message-passing GNNs have at most the distinguishing power of 1-WL (Xu et al., 2019; Barceló et al., 2020).

The released WL-GOOSE configuration performs four refinement rounds. For each $t \in \{0, \ldots, 4\}$, it forms the histogram

$$h_c^{(t)} = \Big|\{v \in V : c_v^{(t)} = c\}\Big|, \tag{2}$$

and concatenates the five histograms into one feature vector. A dictionary built from the training graphs maps observed colours to feature coordinates; colours first encountered at test time therefore have no corresponding learned weight. The feature vector is passed to a regression model, such as support-vector regression (Drucker et al., 1996), to predict $h^*(s)$.

The pipeline is therefore

$$(s, g) \longrightarrow G_s \longrightarrow \text{four rounds of 1-WL} \longrightarrow \text{colour-count vector} \longrightarrow \widehat{h}(s).$$

Although 1-WL eventually stabilises on any fixed finite graph, WL-GOOSE stops after four rounds to keep the number of seen colours manageable. Moreover, its feature coordinates depend on colours observed during training. Spectral representations instead allow a fixed set of statistics to be defined independently of the training graphs, although this alone does not guarantee useful distinguishing power or transfer.

Spectral graph features. A well-established technique for distinguishing graphs, alternative to WL, is to inspect their graph spectra, which are the sorted eigenvalues for some matrix representations of the graphs. We assume familiarity with basic linear algebra, in particular eigenvalues, eigenvectors, eigenspaces, and eigendecomposition (see Deisenroth et al. (2020) for an overview). We give a brief summary of key definitions from spectral graph theory, and refer the reader to Chung (1997) for further details. A graph G can be represented in various matrix forms; for undirected graphs, we are mainly interested in the adjacency matrix A and the combinatorial graph Laplacian $L_{\mathrm{comb}} = D_G - A$, where D_G is G's degree matrix. For such a matrix representation, the sorted list of all its eigenvalues, including repeated values, forms a graph spectrum. This spectrum captures global properties of the graph, and it is unaffected by vertex relabelling, which makes it a convenient tool for testing graph isomorphism: if the spectra of two graphs differ, then the graphs can not be isomorphic; however, it can happen that two non-isomorphic graphs are *cospectral*, i.e., indistinguishable via their spectra. For computing the spectrum, a full dense eigendecomposition takes $\Theta(n^3)$ time and $\Theta(n^2)$ working memory for an n-vertex graph. An iterative sparse solver can compute the k smallest or largest eigenvalues and their eigenvectors by repeated sparse matrix–vector products; its number of iterations depends on the gaps between eigenvalues and on the requested numerical accuracy. The cubic cost can be avoided entirely for features that never need individual eigenvectors, but only reweight each eigenvector by a function of its eigenvalue, a so-called spectral filter, as several of our feature families in Section 4 do. If this function is a polynomial, e.g., λ^2, the reweighted matrix is the same polynomial of the graph matrix, e.g., its square, so multiplying it with a vector needs only repeated sparse matrix–vector products; other functions can be approximated by polynomials, e.g., by truncated Chebyshev expansions (Defferrard et al., 2016). In graph neural networks, spectral filters define graph convolutions (Bruna et al., 2014; Defferrard et al., 2016), and Laplacian eigenvectors serve as positional encodings in GNNs and graph transformers, e.g., to make message passing more expressive than 1-WL (Dwivedi et al., 2022; Lim et al., 2023). A difficulty with using eigenvectors is that they need not be unique: a unit-length eigenvector is only determined

up to its sign, since v and $-v$ are both valid, and within a repeated eigenspace any orthonormal basis may be chosen. SignNet and BasisNet are designed to give the same output under these choices (Lim et al., 2023).

3 LITERATURE

Graph features for learned planning heuristics. Our work builds on WL-GOOSE, which combines WL features of ILGs with support-vector or Gaussian-process regression (Chen et al., 2024b). Its colour-histogram construction follows the WL graph-kernel framework (Shervashidze et al., 2011). Subsequent work extends these representations to numeric planning (Chen et al., 2024a) and improves data generation and feature selection (Hao et al., 2025). A large-scale hyperparameter study further examines trade-offs between WL expressivity, feature size, evaluation cost, and planning performance (Chen et al., 2025). We investigate a complementary question: how changing the graph features affects heuristic learning when the ILGs, training states, regressors, and search procedure are held fixed. We use the four-round WL-GOOSE configuration as our reference rather than exhaustively comparing WL variants.

Spectral and random-walk graph descriptors. Our feature families draw on established spectral representations. The heat-kernel signature describes multiscale diffusion at individual points (Sun et al., 2009), whereas NetLSD uses heat or wave traces to construct fixed-length graph descriptors (Tsitsulin et al., 2018). RetGK combines random-walk return probabilities with node attributes for graph-kernel learning (Zhang et al., 2018); return probabilities also underpin random-walk structural encodings (Dwivedi et al., 2022). Our return moments pool these diagonal statistics by ILG node colour, while transfer features summarise walks or diffusion between colour groups. Chebyshev polynomial filtering provides an eigendecomposition-free approach to evaluating diffusion operators (Defferrard et al., 2016). We study these descriptors and their attribute-aware variants as fixed-dimensional inputs to affine heuristic predictors, focusing on transfer to larger planning problems and feature evaluation cost during search.

Expressivity and spectral invariance. Standard message-passing GNNs are bounded by 1-WL (Xu et al., 2019), motivating more expressive relational architectures for planning (Ståhlberg et al., 2022; 2025). Spectral information provides different rather than uniformly stronger invariants: adjacency spectra and 1-WL are incomparable (Rattan & Seppelt, 2023), while suitable spectral pre-colourings can strengthen WL refinement (Feldman et al., 2023). These results do not imply the same expressivity for truncated or pooled spectral descriptors. SignNet and BasisNet address eigenvector sign and basis ambiguities (Lim et al., 2023), while SPE additionally addresses stability under graph perturbations (Huang et al., 2024). Our SignInv and SignBasis use analogous invariance choices through fixed pooling, without inheriting the guarantees of these neural architectures.

Spectral methods on state-transition graphs. Spectral representations have a history in planning and control. Proto-value functions use Laplacian eigenvectors of state-transition graphs as basis functions for value approximation (Mahadevan & Maggioni, 2007), while Schmalz (2025) studies spectral heuristics for graph search. These graphs differ from ours: their vertices are search states, whereas an ILG represents the objects and facts of a single planning state together with its goal. We instead compute one descriptor per state to learn heuristics across problem instances, rather than spectral coordinates over a state-transition graph.

Other learning objectives and generalised planning. STRIPS-HGN learns heuristic values from delete-relaxation hypergraphs (Shen et al., 2020), while ranking-based objectives directly target the state orderings needed for search rather than cost-to-goal accuracy (Chrestien et al., 2023). We retain regression but therefore evaluate search behaviour separately from prediction error. Generalised policies and planning sketches also transfer across instances through reusable action-selection or subgoal rules (Toyer et al., 2018; Francès et al., 2021; Drexler et al., 2022). In contrast, we learn a scalar search heuristic and focus on the information and computational cost of its graph features.

4 SPECTRAL REPRESENTATIONS FOR PLANNING STATES

We replace WL-GOOSE's four-round 1-WL features with spectral features of the same attributed ILGs, while keeping the training states, regressors, and search procedure fixed. Each representation

maps a state s to a fixed-dimensional vector $\phi(s) \in \mathbb{R}^d$, from which the heuristic is predicted as $\widehat{h}(s) = w^\top \phi(s) + b$. Our goal is therefore to construct features that support affine prediction and transfer to larger problems, requiring fixed-size spectral summaries, encoding of node colours and argument-position edge labels, invariance to arbitrary spectral choices, and efficient evaluation during search.

4.1 Required invariances

Renaming objects or reordering the vertices of an ILG must not change its feature vector. The 1-WL histogram already satisfies this requirement. For spectral methods, a vertex permutation represented by a permutation matrix Π transforms a graph matrix M into $\Pi^\top M \Pi$. Its eigenvalues remain unchanged, but each eigenvector u is reordered as $\Pi^\top u$. Consequently, eigenvector-based features must pool vertex coordinates in a way that respects the corresponding reordering of node attributes.

Eigenvectors introduce two further ambiguities. First, if $Mu = \lambda u$, then $M(-u) = \lambda(-u)$, so an eigensolver may return either sign. Squaring the coordinates removes this ambiguity because $(-u) \odot (-u) = u \odot u$, where $\odot$ denotes entrywise multiplication. Second, when an eigenvalue occurs more than once, its eigenvectors span a multidimensional *eigenspace*. There is no unique orthonormal basis for this space: if the columns of U_λ form one such basis, then $U_\lambda Q$ is another for any orthogonal matrix Q. Squaring individual eigenvectors does not remove this basis ambiguity.

To obtain basis-invariant features, we can summarise the complete eigenspace through its orthogonal projector $P_\lambda = U_\lambda U_\lambda^\top$. This projector describes the space rather than a particular choice of basis:

$$(U_\lambda Q)(U_\lambda Q)^\top = U_\lambda Q Q^\top U_\lambda^\top = U_\lambda U_\lambda^\top .$$

Following Lim et al. (2023), we compare *SignInv*, which removes eigenvector sign ambiguity, with *SignBasis*, which also removes basis ambiguity within complete eigenspaces; all other feature families use eigenvalues or matrix functions and are therefore sign- and basis-invariant by construction.

4.2 Matrix representations of ILG

Here, we describe different matrix representations for the ILG to use in the next section. The choice of matrix is important since different representations can capture different information about the graph. For an undirected graph $G = (V, E)$, the adjacency matrix A is defined by $A_{uv} = \mathbf{1}[(u, v) \in E]$, where $\mathbf{1}[\cdot]$ is a scalar indicator equal to zero or one. D_G is G's degree matrix, so it is a diagonal matrix with $(D_G)_{vv} = \deg(v) = \sum_u A_{vu}$. The normalised adjacency matrix S is $S = D_G^{-1/2} A D_G^{-1/2}$. Note that for isolated vertices with $\deg(v) = 0$ we set $(D_G^{-1/2})_{vv} = 0$. The adjacency matrix is able to count the number of walks of length k from u to v with $(A^k)_{uv}$, and A's spectrum is able to capture this information. In contrast, S is closely related to the random walk matrix $D_G^{-1} A$; it hides information about the vertex degrees, and thereby captures information about the probability that a random walk of length k from u ends at v, rather than the number of such walks. The combinatorial and normalised Laplacians are defined as

$$L_{\text{comb}} = D_G - A, \qquad L_{\text{norm}} = I_+ - S, \qquad (I_+)_{uv} = \begin{cases} \mathbf{1}[(D_G)_{vv} > 0] & \text{if } u = v \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

These capture information about the connectivity of G. For example, the multiplicity of the eigenvalue 0 is the number of connected components of G, and the second smallest eigenvalue is small if G has a bottleneck, i.e. it can be split into two large parts by removing few edges. The combinatorial Laplacian $L_{\text{comb}} = D_G - A$ retains the absolute degree scale and gives the standard heat-diffusion operator used by HKS. The normalised Laplacian $L_{\text{norm}} = I_+ - D_G^{-1/2} A D_G^{-1/2}$ rescales edges by vertex degrees and has spectrum in $[0, 2]$, making fixed spectral summaries more comparable across ILGs of different sizes and degree scales. This level of understanding is sufficient for our purposes, and we point the interested reader to Chung (1997) for more details.

These standard matrices cannot capture information about the edge labels of an ILG, so we show some representations that can, to be used by our feature families. Let $G_s = (V, E, \chi, r)$ be the ILG for state s with $n = |V| > 0$ vertices and $m = |E|$ edges. For a domain with p predicates and maximum predicate arity R, the ILG has $C = 1 + 3p$ node colours: one object colour and three

colours per predicate for true non-goal, true goal, and false goal atoms. Let $B \in \{0,1\}^{n\times C}$ be the node-colour matrix, where $B_{vc} = 1$ exactly when vertex v has colour c. Each column of B is thus an indicator signal for one colour. We index colours by $c \in \{0,\dots,C-1\}$ and argument positions by $\ell \in \{0,\dots,R-1\}$. For each argument position ℓ, let $A_\ell \in \{0,1\}^{n\times n}$ be the adjacency matrix of the ILG restricted to edges with the edge-label ℓ. Because the ILG is undirected, each A_ℓ is symmetric. This relates to the standard adjacency with $A = \sum_{\ell=0}^{R-1} A_\ell$.

Alternatively, edge-label information can be encoded with an edge-subdivision construction. An edge (u, v) with label ℓ is replaced by the path $u - x_{uv\ell} - v$, and the inserted vertex $x_{uv\ell}$ receives a colour identifying ℓ. Contracting these inserted vertices recovers the original attributed ILG. The transformed graph has $n' = n + m$ vertices, $m' = 2m$ edges, and $C' = C + R$ colours. We use primes, such as B' and L'_{norm}, for its matrices.

4.3 Spectral feature families

Smallest and largest eigenvalues. Our simplest baseline retains the ends of the normalised-Laplacian spectrum. Let $\lambda_1 \le \cdots \le \lambda_n$ be the eigenvalues of L_{norm}. We retain the k smallest and k largest values, with $k = 16$. When $n < k$, each list is padded to length k, and a binary validity mask distinguishes padding from genuine zero eigenvalues. The two lists are formed independently, so an eigenvalue can appear in both when $n < 2k$.

Spectral distribution. Rather than retaining individual eigenvalue indices, this representation summarises the entire spectrum. We rescale the eigenvalues as $\mu_i = \lambda_i/2 \in [0, 1]$ and define their empirical cumulative distribution function

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[\mu_i \le x].$$

We evaluate F at $x \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ and include linearly interpolated quantiles at the same five probabilities. This gives ten features describing the spectral distribution. The representation still requires all eigenvalues and does not associate them with node colours.

Sign-invariant eigenvectors (SignInv). The eigenvector baselines incorporate ILG attributes by operating on the subdivided graph and pooling by its node colours. Let u_i be a unit eigenvector of L'_{norm} with eigenvalue λ_i. SignInv retains the eigenvectors corresponding to the eight smallest nonzero and eight largest eigenvalues. For each retained eigenvector, it records a validity bit, λ_i, and the colour-pooled squared coordinates $a_i = B'^{\top}(u_i \odot u_i)$. This construction removes the sign ambiguity but remains sensitive to the basis chosen for a repeated eigenspace.

Sign- and basis-invariant eigenspaces (SignBasis). SignBasis instead pools complete eigenspaces, avoiding dependence on their individual basis vectors. We group numerically equal eigenvalues using the same fixed tolerance in the Python and native implementations. For cluster r, let U_r contain its complete orthonormal basis and let $P_r = U_r U_r^{\top}$ be its projector. We record $a_r = B'^{\top}\,\mathrm{diag}(P_r)$, together with a validity bit, the cluster's mean eigenvalue, and its multiplicity, meaning the number of eigenvectors in the cluster.

Each spectral end has an eight-vector budget: we retain only complete clusters whose total multiplicity fits the budget, omit any cluster that would exceed it, and pad the remaining slots. This preserves basis invariance, which would be lost by truncating an eigenspace.

Diagonal heat-kernel features. The heat-kernel signature (HKS) describes diffusion at different time scales (Sun et al., 2009). Using the combinatorial Laplacian $L_{\mathrm{comb}} = U\Lambda U^{\top}$, we define the heat kernel at time $t > 0$ as

$$H_t = \exp(-tL_{\mathrm{comb}}) = U\exp(-t\Lambda)U^{\top}. \tag{4}$$

Small values of t emphasise nearby structure, while larger values allow diffusion to spread farther through the graph. The diagonal entry $(H_t)_{vv}$ measures the response at vertex v to heat initially placed at that vertex. We pool these entries by node colour:

$$\mathrm{HKS}^{\mathrm{sum}}_{t,c} = \sum_{v=1}^{n} B_{vc}(H_t)_{vv}, \qquad \mathrm{HKS}^{\mathrm{mean}}_{t,c} = \frac{\mathrm{HKS}^{\mathrm{sum}}_{t,c}}{\max(1, \sum_v B_{vc})}. \tag{5}$$

We compute both sums and means for A and separately for each A_ℓ, using the combinatorial Laplacian of each matrix. For the ten diffusion times $t_j = 10^{-1+2j/9}$, $j \in \{0, \ldots, 9\}$, concatenating these values gives $20C(R+1)$ features. This exact calculation uses dense eigendecompositions.

Heat transfer between node colours. Diagonal HKS records diffusion returning to individual vertices. Heat-transfer features instead measure diffusion between colour groups. Let $N_c = \max(1, \sum_v B_{vc})$, $N_B = \mathrm{diag}(N_0, \ldots, N_{C-1})$, and $\widetilde{B} = BN_B^{-1/2}$. Every nonempty column of $\widetilde{B}$ has unit Euclidean norm. We define

$$K_t = \widetilde{B}^\top \exp(-tL_{\mathrm{norm}})\widetilde{B}. \tag{6}$$

Entry (c, d) measures the response of colour group c to a signal on group d; it is not a transition probability because the normalised heat kernel need not have row sums of one. Following Defferrard et al. (2016), we approximate $\lambda \mapsto \exp(-t\lambda)$ on $[0, 2]$ by a degree-32 Chebyshev polynomial, evaluated on L_{norm} and the colour signals using sparse matrix–vector products without eigendecomposition. For the ten HKS diffusion times, we retain the upper triangle of each symmetric K_t. The attributed variant uses L'_{norm} and B', and we compare raw and normalised colour signals, with and without initial colour counts.

Return moments. Matrix powers provide spectral summaries without explicitly computing eigenvectors. If $S = U\Lambda U^\top$, then $S^q = U\Lambda^q U^\top$. Thus, powers of S are spectral moments, even when evaluated directly through matrix multiplication. Our return features are related to random-walk structural encodings based on diagonal matrix powers (Dwivedi et al., 2022).

Let $B_{:c}$ denote column c of B and let $J_c = \mathrm{diag}(B_{:c})$. The colour-rooted return moment is

$$\mathrm{return}_{q,c} = \mathrm{tr}(J_c S^q) = \sum_{v=1}^{n} B_{vc}(S^q)_{vv}. \tag{7}$$

It is a degree-weighted sum of closed walks of length q starting at vertices of colour c. Each traversed edge (u, v) contributes a factor of $1/\sqrt{(D_G)_{uu}(D_G)_{vv}}$ to the walk's weight. Our return-moment baseline uses $q = 4$. This counts length-four closed walks rather than performing the recursive neighbourhood refinement of four-round 1-WL. Since $S^0 = I_n$, initial colour counts are exactly the zeroth-order return moments, $\mathrm{return}_{0,c} = \sum_v B_{vc}$.

Transfer moments. To retain walks between different colour groups, we use

$$T_q = B^\top S^q B = B^\top U\Lambda^q U^\top B, \qquad q \in \{1, \ldots, 8\}.$$

Entry (c, d) is a degree-weighted sum of length-q walks between vertices of colours c and d. We compute these features through the recurrence

$$X_0 = B, \qquad X_q = SX_{q-1}, \qquad T_q = B^\top X_q.$$

This avoids both eigendecomposition and explicit construction of the matrix powers S^q. Retaining the upper triangle, including the diagonal, for orders 1 through 8 gives $8C(C+1)/2$ features.

Attributed transfer moments apply the same construction to S' and B' on the subdivided graph. They use orders 1 through 16, producing $16C'(C'+1)/2$ features. Traversing an original ILG edge takes two steps after subdivision. Appendix A.1 summarises the attributes and graph information retained by each representation.

Expressivity and extrapolation. A representation may assign the same feature vector to states with different targets. Even without such collisions, the targets may not be well approximated by an affine function of the features. Features that work on small training graphs may also become less informative on larger graphs, particularly when eigenvalue cutoffs, diffusion times, and moment orders remain fixed. We therefore examine prediction error and graph-size shift alongside search performance. Appendix A.4 gives examples of these limitations and the corresponding error bounds.

Computational cost. The cost of computing a feature vector is critical because the heuristic is evaluated for every state considered during search. Four-round 1-WL, polynomial heat transfer, and transfer moments can exploit sparse graph operations, whereas a full dense eigendecomposition requires $\Theta(n^3)$ time and $\Theta(n^2)$ working memory for an n-vertex graph. Subdivision increases the graph size to $n' = n + m$ vertices (and $m' = 2m$ edges). Partial eigensolvers can reduce the cost when only a small number of eigenpairs is required, although their runtime also depends on convergence, iteration count, and orthogonalisation costs. Exact return moments can still incur cubic worst-case cost because intermediate matrix powers may become dense and require dense matrix operations. Appendix A.3 gives the detailed per-state bounds.

5 EXPERIMENTS

We compare these spectral representations against the four-round 1-WL baseline of Chen et al. (2024b) as feature generators for the GOOSE pipeline. All representations use the same training data and search procedure; for each, we report from the search the number of node expansions and the time taken. Search time reflects both how well the learned heuristic guides search and how expensive it is to compute, whereas the number of expansions isolates guidance quality. Together, the two metrics show how much guidance each heuristic provides and at what computational cost.

5.1 EXPERIMENTAL SETUP

We use the 900 test problems from the ten domains of the IPC 2023 Learning Track, with 30 easy, 30 medium, and 30 hard problems per domain (Taitler et al., 2024). Following WL-GOOSE (Chen et al., 2024b), for each domain, we generate optimal plans with the classical planner Scorpion (Seipp et al., 2020), and train the domain model on states along these plans labelled with the length of their remaining plan suffix. The plans are optimal and all domains have unit action costs, so every target is the exact goal distance h^* of its state. All methods receive the same attributed ILG. We compare the resulting learned heuristics by using them to guide greedy best-first search in Fast Downward (Helmert, 2006), with common tie-breaking and rounded predictions. Following the IPC 2023 Learning Track rules (Taitler et al., 2024), training is limited to 24 hours and 32 GiB, and search is limited to 1,800 seconds and 8 GiB, using a single linear-algebra thread. One benefit of using 1-WL and our spectral methods is that our training is much cheaper than these limits: we only use 8 GiB and finish in seconds. We report coverage, which counts the number of tasks for which a validated plan is found within these limits.

We evaluate all feature families from Section 4, both with and without appended initial node-colour counts. Counts never include auxiliary vertices introduced for edge labels. Each representation is paired with linear support-vector regression (LSVR), Gaussian-process regression with a dot-product kernel (GPR), and Ridge regression. All three predict an unbounded affine function $\widehat{h}(s) = w^\top \phi(s) + b$. Appendix A.2 gives the regression details and all 63 combinations.

5.2 COVERAGE AND SEARCH BEHAVIOUR

Table 1 reports the best-performing configuration of each family (we include two heat-transfer normalisations, since neither clearly dominates the other). Appendix A.2 gives all 63 configurations.

What are the best configurations? Four-round 1-WL with GPR achieves the highest overall coverage (516 tasks), closely followed by return moments with initial colour counts and GPR (513). The similar totals hide large per-domain differences: 1-WL solves 33 more tasks in Blocksworld and 15 more in Childsnack, whereas return moments solve 29 more in Transport and 14 more in Rovers, with similar coverage on the other domains. The two representations are in this sense complementary, which we view as a positive result: since planning is PSPACE-complete (Bylander, 1994), no single method is likely to work well across all domains, so identifying complementary techniques is valuable.

Do argument-position edge labels help? Mostly, although the best spectral configuration does not use them. Adding the labels to transfer moments increases total coverage in five of the six matched configurations in Appendix A.2, e.g., from 467 to 480 tasks with Ridge and without

Table 1: Coverage on the IPC 2023 Learning Track, with 1,800 seconds and 8 GiB per problem. Each column is one configuration from Table 3; "Colour counts" marks appended initial node-colour counts. All columns contain recorded outcomes for all 900 problems, and bold marks row maxima. "Best on" indicates the number of domains in which the configuration reached best coverage.

		Eigenvalues and eigenvectors				Heat diffusion			Walk moments		
	1-WL	End-points	Distri-bution	Sign Inv	Sign Basis	Kernel	Transfer raw	Transfer norm.	Return	Transfer	Attrib. transfer
Regression	GPR	Ridge	LSVR	Ridge	Ridge	LSVR	Ridge	Ridge	GPR	Ridge	Ridge
Colour counts	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
Blocksworld	**78**	34	33	31	32	32	63	27	45	58	65
Childsnack	**33**	18	17	22	13	18	18	22	18	19	19
Ferry	**77**	60	60	60	60	60	**77**	75	**77**	**77**	**77**
Floortile	**2**	**2**	**2**	1	1	1	1	1	**2**	1	1
Miconic	**90**	69	69	67	67	65	**90**	**90**	**90**	**90**	**90**
Rovers	36	39	39	31	32	31	22	16	**50**	40	38
Satellite	57	48	42	40	34	28	31	33	**61**	30	38
Sokoban	38	32	31	29	25	29	31	37	**39**	35	32
Spanner	**76**	60	60	53	35	50	43	69	73	71	73
Transport	29	41	41	32	31	40	47	44	**58**	48	47
Total	**516**	403	394	366	330	354	423	414	513	469	480
Best on	6	1	1	0	0	0	2	1	**7**	2	2

counts, and gives the best spectral Blocksworld coverage in Table 1 (65). However, the leading return-moment configuration discards the labels and only uses node colours through pooling and the appended initial colour counts.

Do the initial colour counts help? Mostly for features that do not already scale with the colour counts. Appending counts adds 220–227 tasks to the eigenvalue endpoints, which ignore node colours, and 134–196 to heat transfer with normalised probes, but at most 5 to raw heat transfer and the diagonal heat kernel, whose sums over the vertices of each colour already grow with the counts (Appendix A.2). For the leading return moments, counts add 33 tasks with GPR (480 to 513), whereas they lower the coverage of attributed transfer moments by 4–16 tasks with every regression model.

Does the choice of regression matter? Yes, and no model is best for every representation. 1-WL solves 351 tasks with Ridge but 516 with GPR, whereas return moments with counts are robust (484–513). Conversely, GPR is the worst choice for most other spectral families, e.g., transfer moments with counts solve 469 tasks with Ridge but only 396 with GPR (Appendix A.2).

On the 461 tasks solved by both leading configurations, return moments need 1.14 times as many expansions as 1-WL at similar feature time per state. Appendix A.2 compares further design choices, and Appendix B analyses guidance, computational cost, and out-of-fold prediction error.

6 Conclusion and Future Work

We presented the first controlled comparison of learned spectral and 1-WL features for classical-planning heuristics. With all other inputs fixed, return moments closely match four-round 1-WL in terms of coverage, winning four domains, tying three, and losing three at similar evaluation cost. Out-of-fold prediction error depends on both the representation and the regression model, and in Sokoban it is limited by feature collisions already present in 1-WL; future work should learn from off-trajectory states and evaluate frozen configurations on a fresh problem set. Future work could also study richer attributed operators that preserve interactions between argument-position labels at shared vertices, and directed operators for planning graphs with nontrivial directional cycle structure.

AI USE STATEMENT

Generative AI has been used for both writing and code generation. For writing, we used LLMs for initial section and figure generation generation and final polishing. The sections served only as initial inspiration and have been almost completely rewritten in the process.

As for code, we used generative AI for quick prototyping new feature families and suggestions for alternative spectral families. These have all been collected and moved into a final repository including scripts the LLMs generated for final experiments we executed on a compute cluster.

REPRODUCIBILITY STATEMENT

The supplementary material contains all code, the training and test data, and a single experiment script that trains every model and runs all 63 configurations reported in Table 3. Section 5 and Appendix A.2 give all hyperparameters and resource limits.

REFERENCES

Pablo Barceló, Egor Kostylev, Mikaël Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*, 2014.

Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

Dillon Z. Chen and Sylvie Thiébaux. Graph learning for numeric planning. In *Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 91156–91183, 2024.

Dillon Z. Chen, Sylvie Thiébaux, and Felipe Trevizan. Learning domain-independent heuristics for grounded and lifted planning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 20078–20086. AAAI Press, 2024a.

Dillon Z. Chen, Felipe Trevizan, and Sylvie Thiébaux. Return to tradition: Learning reliable heuristics with classical machine learning. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 68–76. AAAI Press, 2024b.

Dillon Ze Chen, Johannes Zenn, Tristan Cinquin, and Sheila A. McIlraith. Language models for generalised PDDL planning: Synthesising sound and programmatic policies. In *ICAPS 2025 Workshop on Planning in the Era of LLMs (LM4Plan)*, 2025.

Leah Chrestien, Stefan Edelkamp, Antonin Komenda, and Tomas Pevny. Optimize planning heuristics to rank, not to estimate cost-to-goal. In *Proceedings of the Thirty-Seventh Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 25508–25527, 2023.

Fan R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the Thirtieth Conference on Neural Information Processing Systems (NIPS)*, pp. 3837–3845. Curran Associates, Inc., 2016.

Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

Dominik Drexler, Jendrik Seipp, and Hector Geffner. Learning sketches for decomposing planning problems into subproblems of bounded width. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 62–70. AAAI Press, 2022.

Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems (NIPS)*, pp. 155–161, 1996.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations (ICLR)*, 2022.

Or Feldman, Amit Boyarski, Shai Feldman, Dani Kogan, Avi Mendelson, and Chaim Baskin. Weisfeiler and leman go infinite: Spectral and combinatorial pre-colorings. *Transactions on Machine Learning Research*, 2023.

Guillem Francès, Blai Bonet, and Hector Geffner. Learning general planning policies from small examples without supervision. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 11801–11808. AAAI Press, 2021.

Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*, volume 7 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2013.

Mingyu Hao, Dillon Z. Chen, Felipe Trevizan, and Sylvie Thiébaux. Effective data generation and feature selection in learning for planning. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*, pp. 4969–4976. IOS Press, 2025.

Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. In *International Conference on Learning Representations (ICLR)*, 2024.

Sergio Jiménez, Javier Segovia-Aguas, and Anders Jonsson. A review of generalized planning. *The Knowledge Engineering Review*, 34:e5, 2019.

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess E. Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(74):2169–2231, 2007.

Gaurav Rattan and Tim Seppelt. Weisfeiler–leman and graph spectra. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2268–2285, 2023.

Nicholas Rossetti, Massimiliano Tummolo, Alfonso Emilio Gerevini, Luca Putelli, Ivan Serina, Mattia Chiari, and Matteo Olivato. Learning general policies for planning through GPT models. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 500–508. AAAI Press, 2024.

Johannes Schmalz. Finding plans and heuristics with spectral graph theory. In *ICAPS 2025 Workshop on Heuristics and Search for Domain-Independent Planning (HSDIP)*, 2025. Workshop paper.

Jendrik Seipp, Thomas Keller, and Malte Helmert. Saturated cost partitioning for optimal classical planning. *Journal of Artificial Intelligence Research*, 67:129–167, 2020.

William Shen, Felipe Trevizan, and Sylvie Thiébaux. Learning domain-independent planning heuristics with hypergraph networks. In *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 574–584. AAAI Press, 2020.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561, 2011.

John Slaney and Sylvie Thiébaux. Blocks World revisited. *Artificial Intelligence*, 125(1–2):119–153, 2001.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 629–637. AAAI Press, 2022.

Simon Ståhlberg, Blai Bonet, and Hector Geffner. Learning more expressive general policies for classical planning domains. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 26697–26706. AAAI Press, 2025.

Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5):1383–1392, 2009.

Ayal Taitler, Ron Alford, Joan Espasa, Gregor Behnke, Daniel Fišer, Michael Gimelfarb, Florian Pommerening, Scott Sanner, Enrico Scala, Dominik Schreiber, Javier Segovia-Aguas, and Jendrik Seipp. The 2023 International Planning Competition. *AI Magazine*, 45(2):280–296, 2024. doi: 10.1002/aaai.12169.

Sam Toyer, Felipe Trevizan, Sylvie Thiébaux, and Lexing Xie. Action schema networks: Generalised policies with deep learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pp. 6294–6301. AAAI Press, 2018.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander M. Bronstein, and Emmanuel Müller. Netlsd: Hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2347–2356, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the Seventh International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. RetGK: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

A Additional details on the spectral representations

A.1 Summary of the feature families

Table 2 summarises how each representation utilises the ILG. The entries describe the information provided before feeding it into the linear heuristic-prediction model.

Table 2: How each spectral representation uses node colours and argument-position edge labels. "Subdivision" denotes the edge-subdivision construction, and a dash means the attribute is not used.

Method	Node colours	Edge labels	Information retained
Eigenvalue endpoints	–	–	extreme eigenvalues
Spectral distribution	–	–	eigenvalue distribution
SignInv	squared eigenvectors	subdivision	low/high eigenvectors
SignBasis	eigenspace projectors	subdivision	low/high eigenspaces
Diagonal heat kernel	pooling groups	per-position matrices	diffusion at nodes
Chebyshev heat transfer	signals	subdivision	diffusion between colours
Return moments	pooling groups	–	closed walks
Transfer moments	signals	–	walks between colours
Attributed transfer moments	signals	subdivision	labelled walks

A.2 Coverage ablation

Table 3 reports every combination of representation, whether we use counts, and regression model. It contains 63 configurations in total. Without colour counts, a configuration uses only the representation itself; with colour counts, it adds the initial node-colour counts as well. 1-WL uses four refinement rounds. For edge-subdivision methods, counts include only the original vertex colours and exclude inserted argument-position vertices. Heat-transfer probes are reported both raw and normalised, where normalised probes scale each colour indicator by the inverse square root of its class size. LSVR uses the ϵ-insensitive loss with $C = 10^5$, $\epsilon = 0$, no intercept, and at most 25,000 iterations; GPR uses a dot-product kernel with observation noise 10^{-7}. Ridge standardises features and selects its regularisation strength from $10^{-6}, 10^{-5}, \ldots, 10^{6}$ by problem-grouped five-fold cross-validation. For spectral Ridge, coordinates that are nearly constant in the training fold are suppressed to avoid amplifying numerical noise.

Other spectral representations. None matches the leading configurations, but the matched configurations in Table 3 answer two design questions. First, removing the basis ambiguity does not pay off: SignInv solves more tasks than SignBasis in five of the six matched configurations, e.g., 366 versus 330 with Ridge and counts. Second, raw heat-transfer probes outperform normalised ones in all six matched configurations; the best raw configuration solves 468 tasks with counts and LSVR. Summarising the whole spectrum helps only without counts: the spectral distribution then solves 240–244 tasks against 176–179 for the eigenvalue endpoints, whereas with counts the endpoints are ahead (399–403 against 379–394).

A.3 Detailed computational complexity

Table 4 gives the asymptotic time and memory costs of each feature. The partial-spectrum row is an alternative to the dense decompositions used by our implementation; its bound includes orthogonalization for a block of k vectors, not only sparse matrix–vector products. SignInv and SignBasis use a dense eigendecomposition of the graph with inserted argument-position vertices; their pooling cost is lower order than this decomposition. Exact return moments are not automatically linear in the number of edges because powers of a sparse matrix can become dense.

A.4 Limits of fixed graph summaries

Invariance under vertex renaming. Let Π be the permutation matrix for a renaming of the graph vertices. Every graph matrix M becomes $\Pi^\top M \Pi$. This transformation leaves eigenvalues and

Table 3: Coverage for all 63 tested configurations on the 900-problem test set. All configurations contain recorded outcomes for all 900 problems. Bold marks the highest recorded coverage in each row.

Representation	Colour counts	Ridge	LSVR	GPR
Four-round 1-WL	-	351	507	**516**
Eigenvalue endpoints	No	176	**179**	**179**
Eigenvalue endpoints	Yes	**403**	400	399
Spectral distribution	No	**244**	240	243
Spectral distribution	Yes	379	**394**	381
Diagonal heat kernel	No	322	**350**	268
Diagonal heat kernel	Yes	320	**354**	270
Return moments	No	477	476	**480**
Return moments	Yes	484	498	**513**
Transfer moments	No	**467**	448	398
Transfer moments	Yes	**469**	465	396
Attributed transfer moments	No	**480**	468	432
Attributed transfer moments	Yes	**476**	457	416
Chebyshev heat transfer, raw	No	424	**463**	395
Chebyshev heat transfer, raw	Yes	423	**468**	395
Chebyshev heat transfer, normalised	No	235	239	**242**
Chebyshev heat transfer, normalised	Yes	414	**435**	376
SignInv	No	**261**	240	238
SignInv	Yes	**366**	346	325
SignBasis	No	**220**	201	200
SignBasis	Yes	330	**341**	327

Table 4: Per-state asymptotic costs. Here $m = |E|$, C is the number of node colours, R the number of argument positions, q a moment or polynomial order, k the number of eigenpairs, and N_{it} the number of iterative-solver iterations. For the graph with inserted argument-position vertices, $n' = n + m$, $m' = 2m$, and $C' = C + R$.

Method	Time	Working memory
four-round 1-WL	$O(4(m \log n + n))$	$O(m + n)$ plus colour dictionary
dense spectrum or heat kernel	$\Theta(n^3)$	$\Theta(n^2)$
R separate dense matrices	$\Theta(Rn^3)$	$\Theta(n^2)$ one at a time
partial k-spectrum (alternative)	$O(N_{\mathrm{it}}(mk + nk^2))$	$O(m + kn)$
SignInv on lifted graph	$O(n'^3 + kn'C')$	$O(n'^2 + n'C')$
SignBasis on lifted graph	$O(n'^3 + kn'C')$	$O(n'^2 + n'C')$
Chebyshev heat transfer	$O(q(m'C' + n'C'^2))$	$O(m' + n'C' + C'^2)$
transfer moments to order q	$O(q(mC + nC^2))$	$O(m + nC + C^2)$
attributed transfer to order q	$O(q(m'C' + n'C'^2))$	$O(m' + n'C' + C'^2)$
fourth-order return moments	fill-in dependent ($O(n^3)$ worst case)	up to $O(n^2)$

traces unchanged. Additionally, our heat, return, and transfer features remain unchanged because the node-colour matrix is transformed by the same permutation. SignInv remains sensitive to the basis returned for a repeated eigenspace. For complete eigenspaces, SignBasis avoids this ambiguity because $U_r U_r^\top$ is unchanged when U_r is replaced by $U_r R$ for any orthogonal matrix R of the appropriate dimension.

A linear heuristic requires the same number of features for every state, so we keep the eight lowest and eight highest eigenvectors. Eigenvalues that agree up to 10^{-9} are treated as one repeated eigenvalue and never split: if its eigenspace does not fit into the remaining slots, we leave it out and fill the slots with zeros.

Consequence of a feature collision. Let ϕ map planning states to $\mathbb{R}^d$. Suppose that $h^*(s)$ and $h^*(s')$ are finite, $\phi(s) = \phi(s')$, and $|h^*(s) - h^*(s')| = \Delta > 0$. Every model over ϕ assigns the same value to both states. Its two absolute errors therefore sum to at least Δ, and at least one error is at least $\Delta/2$. Within a feature-collision class, the median target minimises absolute error and the

mean target minimises squared error. These bounds apply to every predictor, not only to a linear model.

Limits of the representations. With uniform initial labels, 1-WL cannot separate equally sized regular graphs of the same degree, although their spectra can differ. Conversely, cospectral graphs can receive different 1-WL histograms. Moments through order q only describe walks of corresponding lengths. Spectra computed separately for each argument position can lose how differently labelled edges meet at a vertex. Products of the argument-position matrices retain more of this information, but neither a spectrum nor a fixed finite collection of moments reconstructs every ILG.

Transfer to larger paths. We show an example of why fixed spectral features can fail on graphs larger than those from the training set. Consider path graphs with n vertices and a prediction target of $n - 1$. The normalised-Laplacian eigenvalues of such a path are $\lambda_j = 1 - \cos(\pi j/(n-1))$ for $j \in \{0, \ldots, n-1\}$. For fixed $j \geq 1$ and $n \to \infty$, the smallest nonzero eigenvalues decrease as $\Theta(j^2/n^2)$, while each of the finitely many largest eigenvalues converges to two. A fixed affine combination of finitely many endpoint eigenvalues therefore converges to a constant, whereas the target can be unbounded. Similarly, a transfer moment of order q is zero between colour groups more than q edges apart, and heat transfer at a fixed diffusion time decays with distance. Recovering the size requires either an additional size feature, such as the appended colour counts, or a nonlinear readout. Learning such a nonlinear readout is what we wanted to avoid as motivated in the main paper.

B Search and representation diagnostics

B.1 Expansions and runtime

We compare the best 1-WL configuration with the best spectral configuration with initial colour counts (return moments with GPR). The scatterplots in Figure 2 compare expansions, search time and feature evaluation time in the problems solved by both.

On the 461 tasks solved by both leading configurations, return moments require 1.14 times as many expansions, similar feature time per state (1.04 times), and 1.01 times as much search time as 1-WL (geometric means). The per-domain picture differs: in Blocksworld, return moments expand 15.6 times as many states while each feature evaluation costs 0.56 times as much, whereas in Transport they expand only 0.10 times as many states and achieve higher coverage. Neither representation is thus a clear winner, which suggests that combining the two feature families is a promising direction. Heat-kernel features are far more expensive, taking about 0.65–0.94 ms per state compared with 0.02–0.03 ms for return moments and 1-WL, and solve only 354 tasks. These per-state timings are measured along each method's own search trajectory, not on identical states, so they describe end-to-end behaviour rather than a controlled per-state comparison. The only timing comparison on identical states concerns transfer moments: an exactly equivalent sparse implementation reduces their feature time by factors of 5.7–10.2, yet still solves fewer tasks than 1-WL on a 90-task, 60-second screen (44 versus 47), so faster evaluation alone does not close the gap.

B.2 Out-of-fold prediction error

We measure the error of each representation's prediction of h^* on planning problems outside the training set. We split the training problems into five folds and predict each fold with a model fitted on the other four, giving out-of-fold (OOF) predictions. All states from one planning problem remain in the same fold.

Figure 3 compares four-round 1-WL with the six spectral families of the main screen (all except heat transfer, SignInv, and SignBasis) under a Ridge readout, with initial colour counts except for attributed transfer moments. Most methods predict h^* closely in Childsnack, Ferry, and Miconic. The spectral methods are clearly more accurate than 1-WL in Blocksworld, Floortile, Rovers, Satellite, and Transport; for example, transfer moments with counts reach an MAE of 1.05 in Blocksworld compared with 3.42 for 1-WL. This matches the Ridge search totals in Table 3, where every spectral family with counts except the diagonal heat kernel solves more problems than 1-WL. The last two rows of Figure 3 use GPR, the regression model of the two best search configurations: it barely

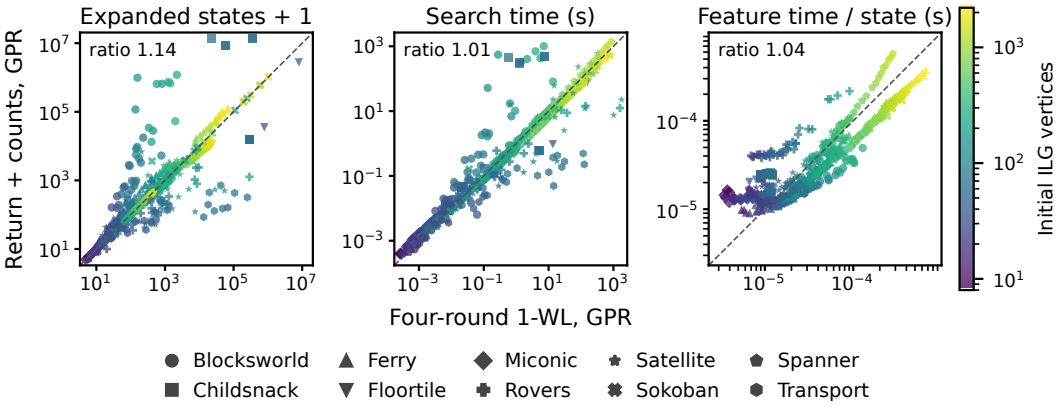

Figure 2: Best 1-WL versus best return moments with counts, both with GPR, on the 461 jointly solved problems. The panels show expanded states plus one (left), search time (centre), and feature-extraction time per evaluated state (right). Marker shape identifies the domain, and colour gives the initial ILG size. The points below the diagonal favour return moments. Each panel reports the geometric-mean ratio of return moments to 1-WL.

	Blocksworld	Childsnack	Ferry	Floortile	Miconic	Rovers	Satellite	Sokoban	Spanner	Transport
1-WL	3.42	0.53	0.72	1.33	**0.70**	2.64	2.30	5.57	0.99	1.57
Endpoints + counts	2.47	0.60	0.73	0.78	0.78	1.42	0.79	**5.53**	1.14	1.11
Distribution + counts	2.35	0.53	0.69	0.77	0.74	1.26	0.73	5.60	1.07	1.06
Heat kernel + counts	1.58	**0.51**	0.66	0.74	**0.70**	1.12	0.46	5.57	**0.95**	1.09
Return + counts	2.13	0.52	0.69	0.77	0.71	**1.03**	0.62	5.56	1.06	0.99
Transfer + counts	**1.05**	0.52	**0.22**	**0.73**	0.71	1.10	0.46	5.57	0.98	**0.98**
Attributed transfer	1.22	**0.51**	**0.22**	0.74	0.71	1.11	**0.43**	5.57	0.99	1.00
1-WL (GPR)	1.30	0.53	0.27	0.75	0.71	1.60	0.67	5.61	0.99	1.47
Return + counts (GPR)	2.13	0.52	0.69	0.77	0.72	1.04	0.62	5.61	1.06	1.00

Figure 3: Problem-grouped OOF MAE on the training states of every domain. The first seven rows use Ridge and the last two rows use GPR. Shading gives the MAE relative to the best row in the same domain, and bold marks that best value.

changes the error of return moments, but reduces the error of 1-WL substantially, for example from 3.42 to 1.30 in Blocksworld and from 2.30 to 0.67 in Satellite, so that 1-WL becomes clearly more accurate than return moments in Blocksworld and Ferry, whereas return moments remain clearly more accurate in Rovers and Transport. Sokoban is difficult for every representation because four-round 1-WL already assigns identical features to training states with different h^*, which bounds the RMSE of any predictor over these features from below by 4.69 (Appendix A.4). Low regression error is also not sufficient for good search: it averages over states, whereas greedy search depends on the ordering of successors and frontier states.

Figure 4 shows the individual OOF predictions of the two strongest search configurations, four-round 1-WL and return moments with counts, both with GPR. In Sokoban, both predict nearly constant values over wide ranges of h^*, which produces horizontal bands. Return moments are more accurate than 1-WL in Rovers and Transport but spread more widely in Blocksworld.

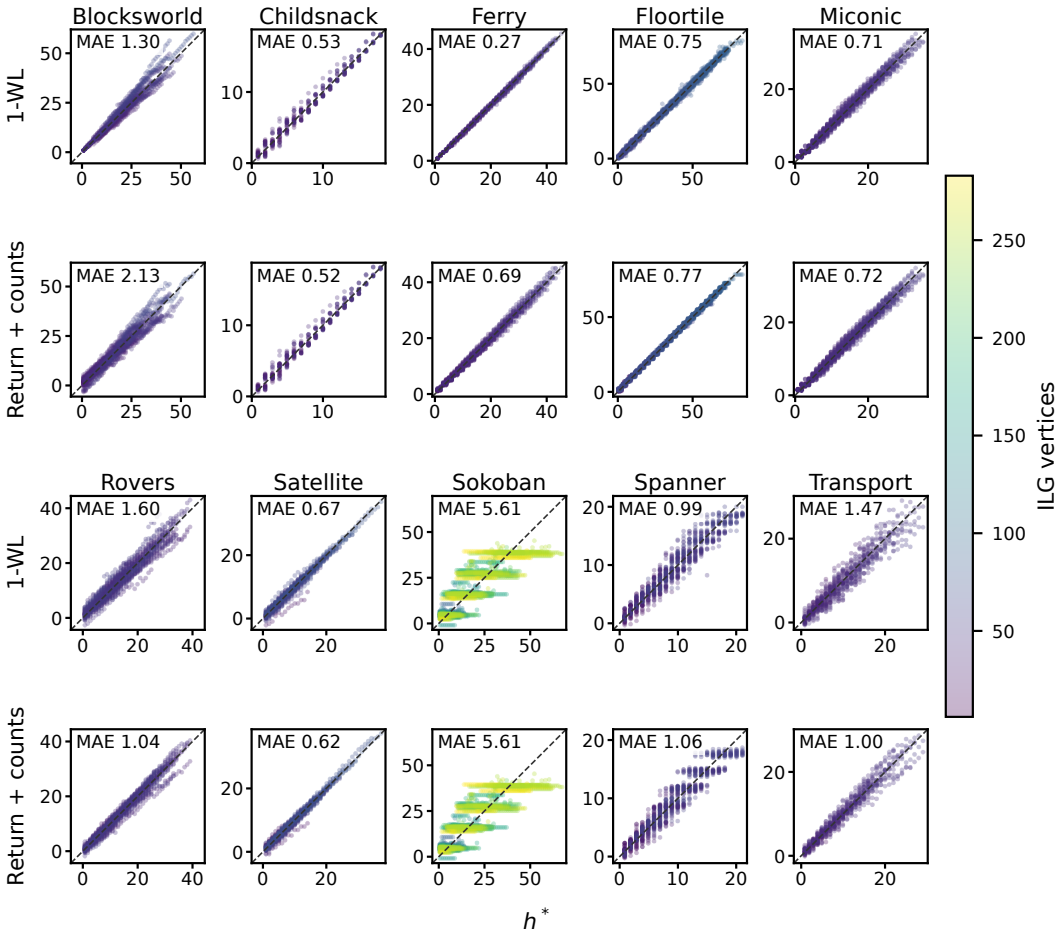

Figure 4: Problem-grouped OOF GPR predictions against h^* for four-round 1-WL and return moments with counts. Each domain has one 1-WL panel above one return-moment panel; colour gives the number of ILG vertices, and the dashed diagonal denotes exact prediction.