Finding Plans and Heuristics with Spectral Graph Theory

Johannes Schmalz

Saarland University, Germany

Abstract

Spectral graph theory can be used to construct plans and heuristics for undirected graphs with a goal. A recent algorithm computes the smallest eigenvector of a graph's Dirichlet Laplacian matrix and interprets this eigenvector as a heuristic function over the graph's vertices – greedily following this heuristic is guaranteed to lead to the goal. In this paper, we show that finding the smallest eigenvector is equivalent to finding a network flow over the graph, and then it becomes intuitively clear that following the flow must lead to the goal. We also show that the eigenvector produces a consistent, goal-aware heuristic, and discuss how this approach may be applicable to planning.

1 Introduction

Spectral graph theory is a field of mathematics that encodes graphs with various matrix representations and then investigates the properties of these matrices' eigenvectors and eigenvalues. For an overview, we refer to the text by Chung (1997) or the work-in-progress text by Spielman (2025). Spectral graph theory has applications in combinatorial optimisation, data mining, computer vision and pattern recognition, internet search, load balancing, and many other areas of computer science (Cvetković and Simić 2011). As a concrete example, specific eigenvalues can be used to upper bound the max cut of a regular graph, and generalisations of this connection have been exploited to produce efficient approximation algorithms (Trevisan 2009). Curiously, the problem of producing plans has only recently been investigated by Steinerberger (2021), who considers undirected graphs with a goal vertex and presents a spectral algorithm that produces plans from each non-goal vertex to the goal, and proves the algorithm's correctness. However, he admits that the effectiveness of this algorithm is not well understood. He relates his algorithm for this discrete problem to a more general continuous problem, and argues that any insight into the algorithm for graphs may give valuable insights into the generalisation.

In this paper, we attempt to demystify the algorithm by showing that it is equivalent to an optimisation problem that computes a feasible network-flow solution over the graph. By treating the solution as a way to route flow from all non-goal vertices to the goal as a sink, it becomes intuitively clear that "following the flow" must reach the goal, and gives insight into further properties of the spectral solution. Furthermore, we demonstrate that the algorithm produces a consistent, goal-aware heuristic, which gives us a lower bound on the minimal plan length to complement the upper bound produced by the spectral plans, i.e., the algorithm gives error bounds on its solution. Afterwards, we present some example graphs to illustrate the behaviours of the spectral algorithm's solutions, which gives more intuition for how the solution works and answers one of Steinerberger's open questions. Finally, we motivate that the spectral approach may be applicable to planning problems. There are graph operations that let us combine graphs into larger graphs in a way that the larger graph's eigenvectors can be easily computed from the eigenvectors of the smaller graphs. Then, in the spirit of planning techniques it may be possible to solve large problems via a decomposition into smaller subproblems.

2 Background: Spectral Graph Theory

We present some preliminaries of spectral graph theory, and explain the spectral algorithm for finding paths on undirected graphs by Steinerberger (2021).

We consider graphs $G = \langle V, E, g\rangle$, where V are the vertices, E are undirected edges, and $g \in V$ is a goal vertex. To denote an undirected edge between vertices i and j we write $\{i, j\}$ or equivalently $\{j, i\}$. For $i, j \in V$ we write $i \sim j$ to denote that i and j are neighbours, i.e., $\{i, j\} \in E$. Later, it will be useful to consider such an undirected edge $\{i, j\}$ as two directed edges, which we write as $\langle i, j\rangle$ and $\langle j, i\rangle$, noting that these two directed edges are not equal. We assume that our graphs are connected and simple, i.e., there is a path between any pair of vertices, there are no self loops with $i \sim i$, and there is at most one edge between any pair of vertices. We additionally require for ease of presentation that removing g from V does not make the graph disconnected, but this requirement is not necessary and can be handled.

The first step of spectral graph theory is to represent our graphs in matrix form. Arguably the most canonical representation of a graph is the *adjacency matrix* $A \in \mathbb{R}^{|V|\times|V|}$ which is defined such that the entry at i, j has

$$A_{i,j} = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{else.} \end{cases}$$

It is also useful to consider the number of edges that are incident to a vertex, called the vertex's degree $\deg(i) =$

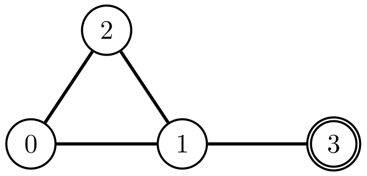


Figure 1: Simple, undirected, and connected graph with a goal.

$|\{i, j\} \in E : j \in V|$, which we can encode in the *degree matrix* $D \in \mathbb{R}^{|V|\times|V|}$ with

$$D_{i,j} = \begin{cases} \deg(i) & \text{if } i = i \\ 0 & \text{else.} \end{cases}$$

These two pieces of information are combined in the *Laplacian matrix* $L = D - A \in \mathbb{R}^{|V|\times|V|}$. Finally, the matrix that we will use for the spectral algorithm modifies L by removing the row and column for the goal g, thereby encoding that g is the goal. This gives us the *Dirichlet Laplacian* $L_g \in \mathbb{R}^{|V|-1\times|V|-1}$ (Biyikoğu, Leydold, and Stadler 2007).

To illustrate these matrices, consider the graph with $V = \{0, 1, 2, 3\}$, edges $\{\{0, 1\}, \{0, 2\}, \{1, 2\}, \{1, 3\}\}$ and goal 3 as shown in fig. 1. Its matrices are as follows, where we give the appropriate vertices' indexes above and to the left of each matrix, and empty entries represent 0:

$$\text{Adjacency } A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & & 1 & 1 & \\ 1 & 1 & & 1 & 1 \\ 2 & 1 & 1 & & \\ 3 & & 1 & & \end{array}$$

$$\text{Degree } D = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 2 & & & \\ 1 & & 3 & & \\ 2 & & & 2 & \\ 3 & & & & 1 \end{array}$$

$$\text{Laplacian } L = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 2 & -1 & -1 & \\ 1 & -1 & 3 & -1 & -1 \\ 2 & -1 & -1 & 2 & \\ 3 & & -1 & & 1 \end{array}$$

$$\text{Dirichlet } L_g = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2 & -1 & -1 \\ 1 & -1 & 3 & -1 \\ 2 & -1 & -1 & 2 \end{array}$$

We point out that the adjacency matrix is symmetric because the graph's edges are undirected, and it follows that all the other matrices are symmetric as well. This is an important property because it guarantees that its eigenvalues and eigenvectors are real with no imaginary parts.

For the matrix L_g, an eigenvector $\boldsymbol{x} \in \mathbb{R}^{|V|-1}$ is a vector that satisfies

$$L_g\boldsymbol{x} = \lambda\boldsymbol{x}$$

for the eigenvalue $\lambda \in \mathbb{R}$. L_g has $n = |V| - 1$ eigenvalues $\lambda_0, \ldots, \lambda_{n-1}$ and corresponding eigenvectors $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}$ which can be sorted according to the magnitude of the eigenvalues. Then, λ_0 is the smallest eigenvalue and we write $\boldsymbol{v}$ to denote the corresponding "smallest eigenvector."

Now, we are equipped to describe the spectral algorithm of Steinerberger (2021):

Input: a graph with goal $G = \langle V, E, g\rangle$ and initial vertex i.
Output: a plan from i to g.

1. Construct G's Dirichlet Laplacian L_g.
2. Find the smallest eigenvector $\boldsymbol{v}$ for L_g, which can be chosen so that $v_i > 0\ \forall i \in V$ without loss of generality.[1] We extend $\boldsymbol{v}$ to include an entry for the goal and set $v_g = 0$.
3. Starting from i, move through the graph greedily w.r.t. $\boldsymbol{v}$. That is, starting with $j = i$, move from j to a neighbour k with the smallest value v_k, setting $j \leftarrow k$ with each step, and repeat this until you reach the goal g.

Under the lens of planning, we can interpret $\boldsymbol{v}$ as a heuristic function that maps each vertex to an estimate of the cost to reach the goal, and then step 3 of the algorithm moves from i to g by following the heuristic greedily. We note that the first two steps of this algorithm are not dependent on the initial vertex i, so $\boldsymbol{v}$ can be computed once and then used to construct plans to the goal from any non-goal vertex in the graph, i.e., it is an all-sources single-target path algorithm. Steinerberger shows that this algorithm of following $\boldsymbol{v}$ greedily is guaranteed to reach the goal, although not necessarily with the minimal number of steps. In the case of tree graphs, i.e., graphs where exactly one path exists between any pair of vertices, this algorithm is proven to find minimal plans. Moreover, Steinerberger shows that from each non-goal vertex there is always a neighbour with lower $\boldsymbol{v}$ value, and that $v_i \leq 0$ only for the goal — consequently, one need not follow $\boldsymbol{v}$ greedily, and it is sufficient to move to some neighbour with a smaller $\boldsymbol{v}$ value to reach the goal.

We conclude this section by describing how the smallest eigenvector can be computed via optimisation, which will be important to establish a connection between $\boldsymbol{v}$ and network flow. In general, the smallest eigenvector $\boldsymbol{v} \in \mathbb{R}^n$ for a real symmetric matrix $M \in \mathbb{R}^{n\times n}$ can be computed with the Courant-Fischer formula (Saad 2011; Spielman 2025):

$$\boldsymbol{v} = \operatorname*{argmin}_{\boldsymbol{x}\neq 0} \boldsymbol{x}^T M\boldsymbol{x} \text{ s.t. } \|\boldsymbol{x}\| = 1. \tag{1}$$

The term $\boldsymbol{x}^T M\boldsymbol{x}/\boldsymbol{x}^T\boldsymbol{x}$ is called the Rayleigh quotient and if $\boldsymbol{x}$ is an eigenvector then its Rayleigh quotient computes $\boldsymbol{x}$'s corresponding eigenvalue. The term $\|\boldsymbol{x}\|$ is the Euclidean norm of $\boldsymbol{x}$, and the constraint $\|\boldsymbol{x}\| = 1$ allows us to remove the Rayleigh quotient's denominator while ensuring that all eigenvectors are normalised and their results are comparable. Unfortunately, this is a non-convex problem because the constraint forces a solution to lie on the unit sphere of $\mathbb{R}^n$.

3 Background: Network Flow

The Linear Program (LP) over Occupation Measure shown in LP 1 can be used to compute minimal plans from each non-goal vertex to the goal (Puterman 2005). With this LP it

[1]This fact is derived in Steinerberger's proof of theorem 1.

can be useful to imagine the graph as a network of pipes, and the LP's task is to route the flow into the goal with minimal cost. The LP has the variables $x_{i,j}$ for each directed edge $\langle i,j\rangle \in E$, which can be seen as the amount of flow being pushed through the pipe from i to j; we use the macros $out(i) = \sum_{\langle i,j\rangle\in E} x_{i,j}$ and $in(i) = \sum_{\langle j,i\rangle\in E} x_{j,i}$ to encode the amount of flow exiting and entering a vertex i, respectively. C1 ensures that the amount of flow entering a vertex i, from in-flow $in(i)$ and injection α_i, must all exit through $out(i)$. We do not allow negative flow to pass through any of the directed edges with C2. To encode the goal vertex g we say that there are no directed edges leaving g. We also have the so-called state-relevance weights $\boldsymbol{\alpha} \in \mathbb{R}^{|V|-1}$ for each non-goal vertex, which satisfy $\alpha_i > 0\ \forall i \in V$, but otherwise the choice does not matter, but it is convenient to pick them so that $\sum_{i\in V}\alpha_i = 1$. Each value α_i represents how much flow is injected into vertex i, and the cumulative injected flow must be routed to the goal. Consequently, if $\sum_{i\in V}\alpha_i = 1$, then a unit of flow must exit from the goal.

$$\min_{\boldsymbol{x}} \sum_{\langle i,j\rangle\in E} x_{i,j} \text{ s.t. C1–C2} \qquad \text{(LP 1)}$$

$$out(i) - in(i) = \alpha_i \qquad \forall i \in V \setminus \{g\} \quad \text{(C1)}$$

$$x_{i,j} \geq 0 \qquad \forall \langle i,j\rangle \in E \quad \text{(C2)}$$

A similar LP is used to compute optimal solutions for Markov Decision Processes (Puterman 2005), as well as Stochastic Shortest Path problems, which are similar but have goals and no discount factor (Trevizan et al. 2016). Solutions to such problems are called policies, which in our context either map a vertex to a single neighbour that we should move to (such a policy is called deterministic) or to a probability distribution over neighbours, from which we select the next neighbour randomly (called a stochastic policy). A feasible solution $\boldsymbol{x}$ induces the stochastic policy π with $\pi(i,j) = x_{i,j}/out(i)$, where $\pi(i,j)$ represents the probability with which we move to j when we are in i. Such policies are guaranteed to reach the goal (called proper policies) and an optimal solution $\boldsymbol{x}^*$ produces a proper policy that additionally minimises expected cost (called an optimal policy) (Puterman 2005). Moreover, basic solutions $\boldsymbol{x}$ produce deterministic policies, i.e., they have a degenerate probability distribution with a single j that has $\pi(i,j) = 1$ for each i. Basic feasible solutions are significant because they correspond to vertices on the polytope described by the linear equalities and inequalities of an LP, and importantly, these are the solutions that are found by the simplex method, which is the standard algorithm for solving LPs in practice (Bertsimas and Tsitsiklis 1997). In our setting, it is clear that we can obtain a plan from i by following a deterministic proper policy. We can also obtain plans by following a stochastic proper policy, but we have to be careful because stochastic policies can have cycles, e.g., for the graph in fig. 1, consider the policy π with $\pi(0,1) = 1, \pi(1,2) = 0.5, \pi(1,3) = 0.5, \pi(2,0) = 1$. This π is proper because it will eventually reach the goal from any vertex, but it may produce a plan that contains one or more cycles involving vertices $0,1,2$.

4 Solution to Network Flow

Recall that $\boldsymbol{v}$, the smallest eigenvector for L_g, can be computed with the Courant-Fischer minimisation in eq. (1) and can only have positive entries. This is sufficient to show that $\boldsymbol{v}$ must induce a solution for LP 1.

Theorem 1. *The smallest eigenvector for L_g, called $\boldsymbol{v}$, is a feasible solution for LP 1.*

Proof. Consider the smallest eigenvalue λ_0 and its eigenvector $\boldsymbol{v}$. Without loss of generality, we rescale $\boldsymbol{v}$ such that $\sum_{i\in V}\lambda_0 v_i = 1$. Then, we rewrite the i^{th} element of $L_g\boldsymbol{v}$ as follows (recall that we define $v_g = 0$):

$$\begin{aligned}(L_g\boldsymbol{v})_i &= v_i \cdot \deg(i) - \sum_{\{i,j\}\in E} v_j \\ &= \sum_{\{i,j\}\in E}(v_i - v_j) \\ &= \sum_{\langle i,j\rangle\in E: v_i > v_j}(v_i - v_j) + \sum_{\langle i,j\rangle\in E: v_i < v_j}(v_i - v_j) \\ &= \sum_{\langle i,j\rangle\in E: v_i > v_j}(v_i - v_j) - \sum_{\langle j,i\rangle\in E: v_j > v_i}(v_j - v_i)\end{aligned}$$

and we suggestively write $x_{i,j} = \max\{v_i - v_j, 0\}$ and $\alpha_i = \lambda_0\boldsymbol{v}_i$. Then, we get that

$$(L_g\boldsymbol{v})_i = \sum_{\langle i,j\rangle\in E} x_{i,j} - \sum_{\langle j,i\rangle\in E} x_{j,i} = \alpha_i \qquad (2)$$

$$x_{i,j} \geq 0 \qquad (3)$$

$$\sum_{i\in V\setminus\{g\}} \alpha_i = 1 \qquad (4)$$

which precisely satisfies the constraints of LP 1, i.e., it describes a feasible solution for the Occupation Measure LP. The subtlety of this proof is that L_g and $\boldsymbol{v}$ encode information for a graph with undirected edges where flow is allowed to go in either direction, and we need to split this undirected flow into non-negative flow across directed edges. □

Thus, $\boldsymbol{v}$ produces a feasible solution for LP 1, but not generally a basic one. Consider the modified house graph in fig. 2, where we give the value of $\boldsymbol{v}$ inside each vertex. From the bottom-right vertex with value 1.28, there are two neighbours with lower values, namely top-left with 1.00 and top-right with 0.88; this indicates that most of the flow from bottom-right moves to top-right, with the lower pressure, but some flow also moves into top-left. This means that the induced policy is stochastic, i.e., we randomly decide the next vertex according to a non-degenerate probability distribution. It follows that $\boldsymbol{v}$ is a non-basic solution for LP 1 (Puterman 2005). Alternatively, one can see that the solution is non-basic because $x_{i,j} > 0$ for 7 variables and there are only 4 constraints — a basic solution can not have more non-zero variables than constraints. The stochastic policy is still guaranteed to reach the goal, i.e., it is proper, but recall that there

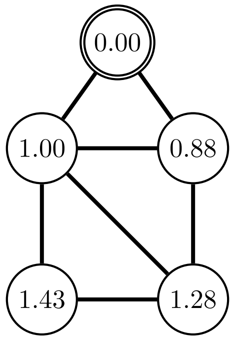


Figure 2: Modified house graph. We give the value of $\boldsymbol{v}$ at each vertex. In this case $\boldsymbol{v}$ produces a stochastic policy.

could be cycles that can only be escaped by eventually randomly picking a particular transition (see section 3). Therefore, this is not enough to conclude that following $\boldsymbol{v}$ greedily leads to the goal. However, we observe that $x_{i,j} > 0$ only if $v_i > v_j$, which has the effect that our policy can have no cycles, e.g., a cycle between i, j, k would require $v_i > v_j > v_k > v_i$, which is a contradiction. Now, we can re-prove Steinerberger's result:

Corollary 2. *The smallest eigenvector* $\mathbf{v}$ *describes a proper policy that has no cycles, so following it greedily is guaranteed to lead to the goal* g*.*

Thus, the smallest eigenvector $\boldsymbol{v}$ of a graph's Dirichlet Laplacian L_g, thanks to the way it is defined via the minimisation of Rayleigh quotients and to the additional properties it gets in the case of L_g, describes a network flow over the graph such that all flow exits through the goal state. This makes it intuitively clear that "following the flow" must reach the goal. Observe that v_i corresponds to α_i in LP 1, which can be interpreted as the amount of flow that is injected into vertex i, and then the flow between vertices is simply the difference of injected flow, in other words, the "difference in pressure." The injected flow can also be interpreted as voltage which induces electrical current, connecting to works such as (Anshelevich 2002). The end result is not new and has already been shown by Steinerberger (2021), but we argue that the intuition of network flow is quite valuable.

5 Consistent Heuristic

In this section, we show that the spectral solution $\boldsymbol{v}$, in addition to producing plans, also describes lower bounds on the minimal plan length by being a consistent, goal-aware heuristic. In the context of a graph with unit-cost edges and the goal g, a heuristic h is consistent if $h(i) \leq h(j) + 1$ and goal-aware if $h(g) = 0$ (Pommerening et al. 2015). It turns out that $\boldsymbol{v}$ satisfies these properties. If we assume that $\|\boldsymbol{v}\| = 1$ from the Courant-Fischer minimisation, then we can scale $\boldsymbol{v}$ by $\kappa \geq 1$ and it remains consistent when we pick κ so that it satisfies

$$\kappa \cdot \max_{\langle i,j \rangle \in E} (v_i - v_j) = 1.$$

Theorem 3. *The heuristic* h *with* $h(i) = \kappa v_i$ *is consistent and goal-aware.*

Proof. Goal-awareness is trivial because v_g is defined as 0. For consistency, our choice of κ guarantees that $\kappa\,(v_i - v_j) \leq 1\ \forall \langle i,j \rangle \in E$. Then:

$$h(i) = \kappa v_i = \kappa(v_i - v_j) + \kappa v_j \leq 1 + h(j).$$

□

Consequently, $h(i)$ gives a lower bound on the minimal plan length from i because h describes a consistent, goal-aware, and therefore admissible heuristic. At the same time, $\boldsymbol{v}$ gives us an upper bound on the minimal plan length from i by constructing some valid plan p_i, which means that $\boldsymbol{v}$ gives an optimality gap with length$(p_i) - h(i)$.

To conclude this section, we observe that $h = \kappa \boldsymbol{v}$ (and in fact the unscaled $\boldsymbol{v}$) is a solution to the dual LP of LP 1, given in LP 2.

$$\max_{\boldsymbol{y}} \sum_{i \in V \setminus \{g\}} \alpha_i y_i \text{ s.t. C3–C4} \qquad \text{(LP 2)}$$

$$y_i \leq y_j + 1 \qquad \forall \langle i,j \rangle \in E \quad \text{(C3)}$$

$$y_g = 0 \qquad \text{(C4)}$$

LP 2's constraints describe the consistency and goal-awareness constraints, and LP 2 computes the consistent, goal-aware potential heuristic over all vertices (Pommerening et al. 2015). An optimal solution for LP 2 produces the perfect heuristic h^*, also called V^* in the probabilistic setting (Puterman 2005). It follows that our eigenvector $\boldsymbol{v}$ describes a solution for the primal LP 1 with $x_{i,j} = \max\{v_i - v_j, 0\}$, and it also describes a solution for the dual LP 2 with $\kappa \boldsymbol{v}$ (and $\boldsymbol{v}$). This is surprising because a feasible solution for a primal LP is not generally easy to transform into a feasible solution for its dual – a primal LP's *optimal* solution is easily transformed into an *optimal* solution for its dual thanks to strong duality, but this is not generally true for feasible solutions. To understand this property of $\boldsymbol{v}$ we point out that thm. 3 applies to any vector $\boldsymbol{x} \in \mathbb{R}^{|V|}$ with $x_g = 0$, $\|\boldsymbol{x}\| = 1$, and $x_i \geq 0\ \forall i$, i.e., a vector $\boldsymbol{x}$ that satisfies these conditions can not have a difference of greater than 1 between any vertex pairs, making it a consistent heuristic. On the other hand, the Courant-Fischer formula eq. (1) restricts $\boldsymbol{x}$ to $\|\boldsymbol{x}\| = 1$. Then, any solution $\boldsymbol{x}$ that is feasible for the Courant-Fischer formula and additionally satisfies $x_i \geq 0\ \forall i \in V$ (and we assume $x_g = 0$) must satisfy consistency by thm. 3. If $\boldsymbol{x}$ is also an eigenvector, which occurs at the minimiser of the Courant-Fischer formula, then $\boldsymbol{x}$ describes a feasible network flow. Thus, $\boldsymbol{v}$ is the unique solution that is simultaneously an eigenvector, has non-zero entries, and has a norm of 1, thereby indirectly producing a satisfying solution for the primal LP 1 and directly giving a solution for the dual LP 2.

6 Descending Heuristic

In addition to being consistent and goal aware, we point out that $\boldsymbol{v}$ (and consequently $h = \kappa \boldsymbol{v}$) is a *descending* heuristic (Seipp et al. 2016). This means that each non-goal vertex has a neighbour with a strictly lower heuristic value. Formally, in our setting, a heuristic h is descending if $h(g) = 0$ and

$$\forall i \in V \setminus \{g\} \;\; \exists \langle i, j \rangle \in E : h(j) < h(i).$$

Steinerberger proves precisely that $\boldsymbol{v}$ is a descending heuristic (with different terminology) and uses this property to demonstrate that following $\boldsymbol{v}$ greedily leads to the goal, which reflects a similar result by Seipp et al. Note that Seipp et al. require additional properties such as *alive* and *solvable* vertices, and *dead-end avoiding* heuristics, but these properties are all trivially satisfied in our setting of undirected connected graphs. Under this lens it becomes clear that following $\boldsymbol{v}$ greedily must lead to the goal, but the fact that $\boldsymbol{v}$ is a descending heuristic remains non-obvious, and our network-flow argument helps to motivate this.

7 Examples

To give an intuition for the behaviour of the spectral algorithm we present some example graphs in fig. 3, where we report the found plan length, the minimal plan length, and the spectral heuristic defined by $h(i) = \kappa \boldsymbol{v}_i$ for each vertex.

In the maze graph (fig. 3a), we observe the found plan length matches the minimal plan length at all vertices, i.e., the eigenvector describes minimal plans for all vertices. This motivates that the spectral algorithm can produce minimal plans for interesting problems, and Steinerberger argues that the algorithm often produces minimal or close-to-minimal plans for a large variety of problems. Turning our attention to the spectral heuristic, it provides a perfect lower bound at the goal's immediate neighbour, but the bound becomes increasingly looser the further we move away from the goal.

To better understand the spectral heuristic's looseness, consider the path graph with 12 vertices, where one of the end points is the goal. We enumerate the heuristic values of vertices, ordered by the vertices' distances to the goal:

$$0, 1, 1.98, 2.93, 3.82, 4.63, 5.37, 6.00, 6.52, 6.92, 7.19, 7.33.$$

We also enumerate the error, i.e., the difference between the heuristic and the minimal plan length in the same order:

$$0, 0, 0.02, 0.07, 0.18, 0.37, 0.63, 1.00, 1.48, 2.08, 2.81, 3.67.$$

In this example it is clear that the heuristic becomes a looser lower bound the further it moves away from the goal. This makes sense because $\boldsymbol{v}$ minimises $\boldsymbol{x}^T L_g \boldsymbol{x}$ which can be expanded into (remember that v_g and x_g are defined as 0):

$$\begin{aligned}\boldsymbol{x}^T L_g \boldsymbol{x} &= \sum_{i \in V} \sum_{\{i,j\} \in E} x_i(x_i - x_j) \\ &= \sum_{\{i,j\} \in E} x_i(x_i - x_j) + x_j(x_j - x_i) \\ &= \sum_{\{i,j\} \in E} (x_i - x_j)^2.\end{aligned}$$

In words, $\boldsymbol{v}$ tries to reduce the squared difference between neighbours, with the effect that vertices with large v_i need to have closer values to their neighbours than vertices with smaller values. Unfortunately, the decay of heuristic value is complex (see appendix A), which makes it non-trivial to normalise values in a way to produce a tighter lower bound.

In the Barnette-Bosák-Lederberg graph (fig. 3b) the spectral algorithm fails to find minimal plans for 13 of the 37 non-goal vertices, where in the worst case the found plan is of length 10 (11) and the minimal plan length is 5 (6). This graph was identified as an example where the spectral algorithm fails to find minimal plans at certain vertices by Steinerberger (2021), although we use a different goal state.

We conclude this section by presenting a family of graphs where the spectral algorithm produces the worst possible plans. The (m, n)-tadpole graph is produced by attaching an m-cycle graph to an n-path graph. Let i be the vertex where the tail (path) attaches to the head (cycle), and let j and k be its neighbours on the cycle. If we set j to be the goal and ensure that the tail is long enough (n sufficiently large) then the spectral algorithm will direct k to go the long way round the cycle, thereby producing the longest possible simple plan from k to j. This is shown in fig. 4.

This phenomenon can be intuitively explained with network flow: if there is sufficiently high pressure coming from the tail along i to j, then even more pressure would be required to push flow from k to j along those pipes, and it is easier to push the flow the other way instead. Looking at it algebraically, if v_i at the connection between head and tail is sufficiently large, then a solution that minimises the difference of squares will choose a smaller value for v_k thereby sending flow the long way around, rather than making v_k even larger than v_i. This behaviour seems to appear on (m, n)-tadpole graphs with $n \geq m - 3$. We observe that the graph's tail does not need to be a path, and can equally well be any graph with enough vertices so that a sufficient amount of flow enters the graph's head to divert the flow from k.

This family of graphs addresses one of Steinerberger's open questions: *"Are there planar graphs such that, for some pair of vertices $i \neq j$, the path produced by the spectral method is c-times as long as the shortest path for any $c > 1$?"* Yes, for any c the $(2c + 2, n)$-tadpole graph with sufficiently large n produces a plan of length $2c$ even though the shortest path is 2.

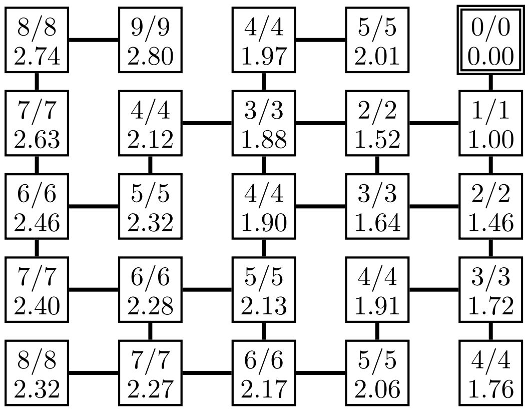
8 Potential Application to Planning

It is clear that for large planning problems it is completely impractical to construct L_g for the transition system, and it is even more impractical to compute the smallest eigenvector for it. However, there are certain graph operations that let us change the structure of a graph while preserving the original graph's eigenvectors. This potentially allows us to find eigenvectors for small graphs, combine the small graphs into larger, more interesting graphs, and then we can easily compute the large graph's eigenvectors.

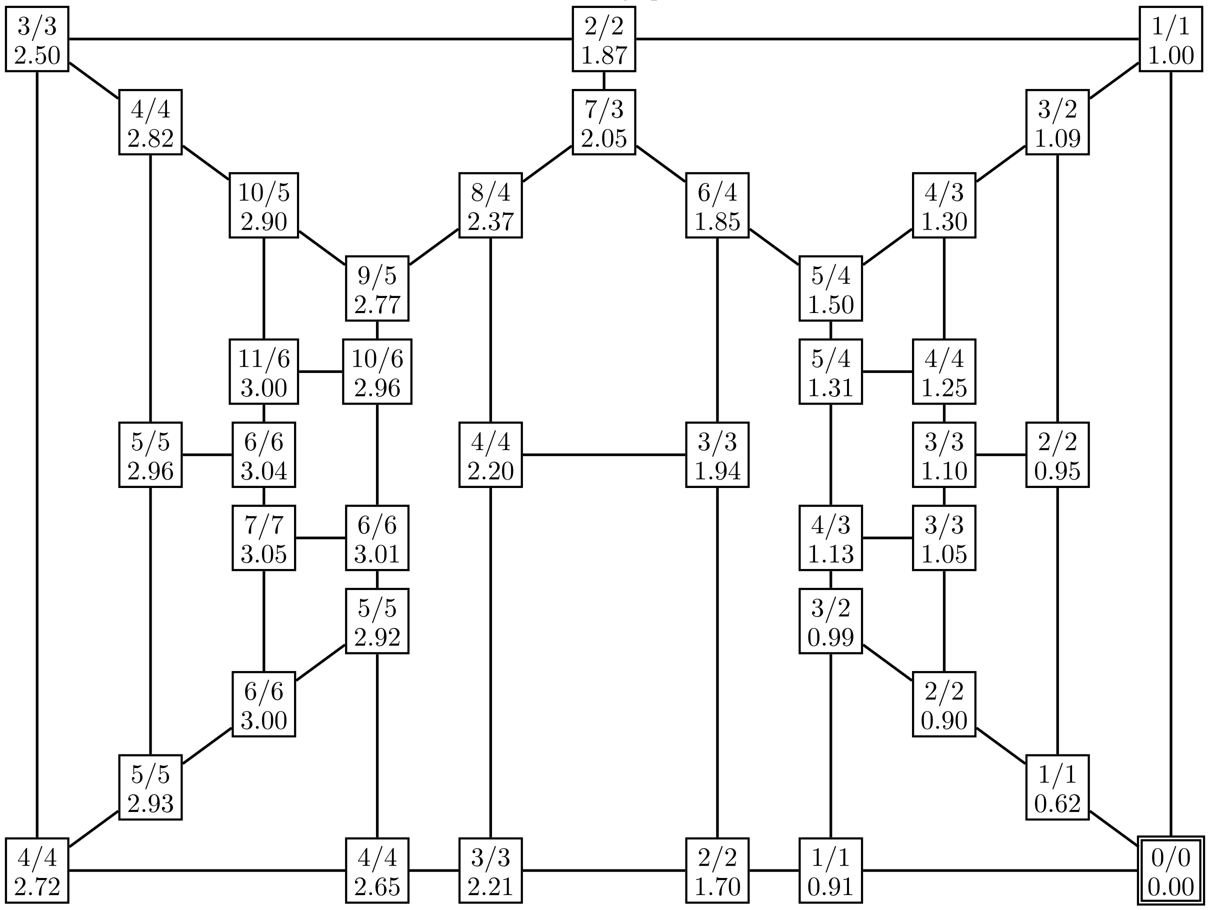
Consider the graphs $G = \langle V_G, E_G \rangle$ and $H = \langle V_H, E_H \rangle$. The Cartesian product for graphs $G \square H = \langle V, E \rangle$ has $V = V_G \times V_H$ and E such that

$$(i, i') \sim (j, j') \text{ iff } (i = j \wedge i' \sim j') \vee (i \sim j \wedge i' = j')$$

where $i \sim j$ iff $\{i, j\} \in E_G$ and $i' \sim j'$ iff $\{i', j'\} \in E_H$. If the G and H's corresponding Laplacian matrices have the

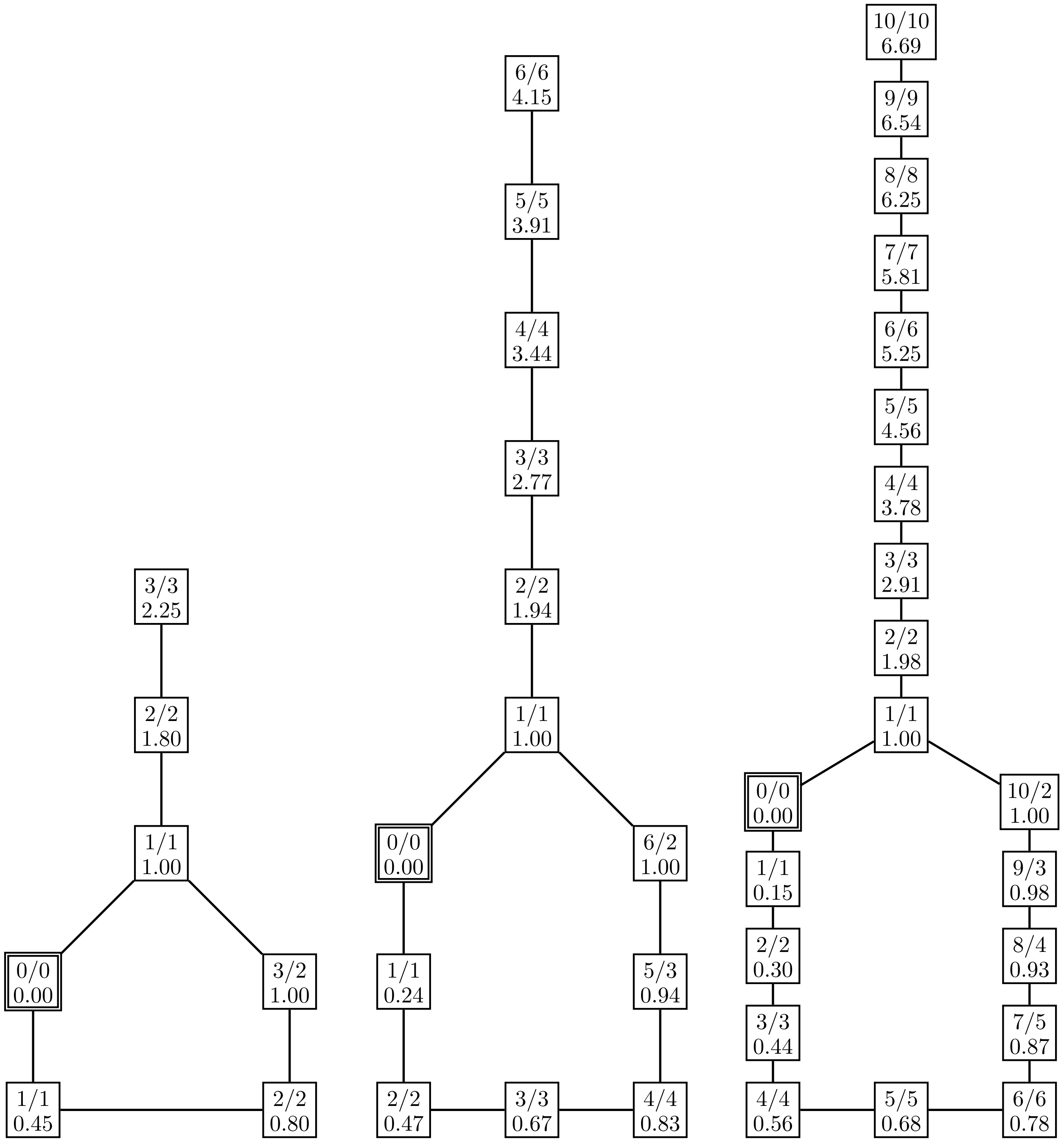


(a) Maze graph.



(b) Barnette-Bosák-Lederberg graph.

Figure 3: The found plan length X, minimal plan length Y, and spectral heuristic Z presented as $X/Y\ Z$.



(a) $(5, 2)$-tadpole graph. (b) $(8, 5)$-tadpole graph. (c) $(12, 9)$-tadpole graph.

Figure 4: The found plan length X, minimal plan length Y, and spectral heuristic Z presented as $X/Y\ Z$.

eigenvectors $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{m-1}$ and $\boldsymbol{y}_0, \dots, \boldsymbol{y}_{n-1}$ for $m = |V_G|$ and $n = |V_H|$, then $G \square H$ has the eigenvectors

$$\left\{ \boldsymbol{v}_i \otimes \boldsymbol{u}_j : i \in \{0, \dots, m-1\}, j \in \{0, \dots, n-1\} \right\}$$

where $\otimes$ is the Kronecker product, which is easy to compute. Consequently, if we are able to express a large graph in terms of Cartesian products of small graphs, then we only need to find solutions for the small graphs and combine them. Unfortunately, the Cartesian product does not allow the subgraphs to interact, in the sense that we can produce a plan for the Cartesian product by concatenating plans for each of the constituent subgraphs, without any concerns of the subplans interfering. This severely limits the graphs that we can construct, and suggests a more efficient approach for these kinds of problems: run Dijkstra's algorithm on each subgraph and concatenate its solutions.

We point out that the Cartesian product preserves all eigenvalues exactly, whereas we only need the smallest eigenvector and may be content with an approximation. For this reason, we optimistically argue that there may be a graph operation that lets us decompose interesting graphs into subgraphs, potentially approximately, such that the subgraphs are sufficiently easy to solve with the spectral algorithm and can be recombined into a solution for the large problem.

9 Related Work

As mentioned in the introduction, there has been surprisingly little work on the topic of finding plans with spectral graph theory. However, it has been used in planning in other contexts. Ren et al. (2024) use the property that the second-smallest eigenvector λ_1 of a graph's normalised Laplacian approximates how well-connected a graph is. They use this connection to estimate the difficulty of automatically generated 2D grid-based problem — well-connected graphs are typically easier to solve and have larger λ_1, whereas poorly-connected graphs, such as mazes, have smaller λ_1.

10 Conclusion

In this paper we investigated Steinerberger's spectral algorithm for finding plans on undirected graphs with goals, which works by computing the smallest eigenvector $\boldsymbol{v}$ for the graph's Dirichlet Laplacian matrix, interpreting this $\boldsymbol{v}$ as a heuristic function over the non-goal states, and then following this heuristic greedily. We gave an alternate proof for the algorithm's correctness by showing that $\boldsymbol{v}$ is a feasible solution to the network flow problem, which gives a simple intuition for why the method works. We further related the algorithm to planning by showing that $\boldsymbol{v}$ also produces a consistent, goal-aware heuristic, with the effect that $\boldsymbol{v}$ simultaneously produces an upper and lower bound on the minimal plan length. Then, we gave examples that demonstrate $\boldsymbol{v}$'s behaviour, and in particular, show that the spectral solution produces arbitrarily suboptimal solutions for a family of tadpole graphs, answering one of Steinerberger's open questions. Finally, we motivated that the spectral method can be practical for large graphs, as long as those graphs can be constructed out of the Cartesian product of smaller graphs: we can compute the smallest eigenvectors for each smaller graph, and then easily combine them into the smallest eigenvector of the large graph, without ever needing to construct the large graph's Dirichlet Laplacian directly. Unfortunately, the Cartesian product is limited, and we would need different operations for combining graphs in order to solve interesting problems.

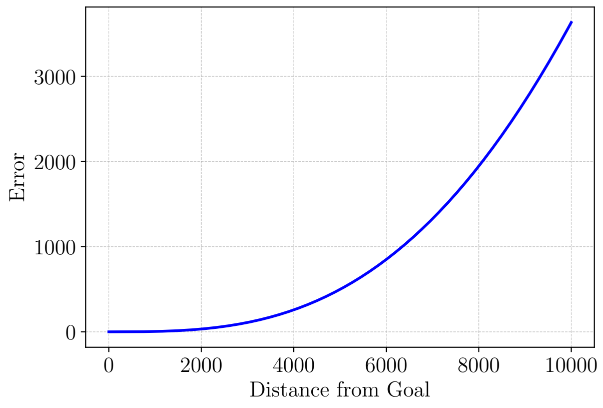


Figure 5: Error on a path graph as distance from goal increases.

A Spectral Heuristic for Path Graph

We consider a path graph with 10000 vertices and report the *error* for each vertex, which is computed by $h^*(i) - h(i)$ for each vertex i, where $h^*(i)$ is the optimal plan length and $h(i)$ is the spectral heuristic. In fig. 5 we show each vertex's error as a function of distance from the goal. Clearly the error grows very large — but at what rate? The best fitting curve we found was around $f(x) = ax^{2.85}$ for the path graphs, with a dependent on the number of vertices. This does not generalise to other graphs: consider a grid graph with the goal in the bottom left corner, as we move away from the goal towards the right, the error w.r.t. distance from the goal is fitted better described by a linear function $f(x) = ax$. At this stage, it is not clear how the error curve correlates to the graph, and whether it can be predicted.

References

Anshelevich, V. V. 2002. A hierarchical approach to computer Hex. *Artificial Intelligence*.

Bertsimas, D.; and Tsitsiklis, J. 1997. *Introduction to Linear Optimization*. Athena Scientific, 1st edition.

Biyikoğu, T.; Leydold, J.; and Stadler, P. F. 2007. *Laplacian Eigenvectors of Graphs*. Springer Berlin Heidelberg.

Chung, F. 1997. *Spectral Graph Theory*. American Mathematical Society.

Cvetković, D.; and Simić, S. 2011. Graph spectra in Computer Science. *Linear Algebra and its Applications*.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. *Proc. of AAAI Conf. on Artificial Intelligence*.

Puterman, M. L. 2005. *Markov Decision Processes*. Wiley-Interscience.

Ren, J.; Ewing, E.; Kumar, T. K. S.; Koenig, S.; and Ayanian, N. 2024. Map Connectivity and Empirical Hardness of Grid-based Multi-Agent Pathfinding Problem. In *Proc. of Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Saad, Y. 2011. *Numerical Methods for Large Eigenvalue Problems: Revised Edition*. Society for Industrial and Applied Mathematics (SIAM).

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016. Correlation complexity of classical planning domains. In *Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI)*.

Spielman, D. 2025. Spectral and Algebraic Graph Theory. http://cs-www.cs.yale.edu/homes/spielman/sagt/. Work in progress. Accessed July 2025.

Steinerberger, S. 2021. A spectral approach to the shortest path problem. *Linear Algebra and its Applications*.

Trevisan, L. 2009. Max cut and the smallest eigenvalue. In *Proc. of 41st ACM Symposium on Theory of computing*.

Trevizan, F.; Thiébaux, S.; Santana, P.; and Williams, B. 2016. Heuristic Search in Dual Space for Constrained Stochastic Shortest Path Problems. In *Proc. of 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.