Polarisation and influence in online social networks

Annealed Heuristic Search for Local Community Detection in Probabilistically Weighted Graphs

Andrei Bardas[1]

Supervisor: Dr. Anna L.D. Latour[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Andrei Bardas
Final project course: CSE3000 Research Project
Thesis committee: Dr. Anna L.D. Latour, Dr. Megha Khosla

Abstract

In a polarised online social network, a user is exposed to many posts and accounts from across the platform. As a result, the social graph is dense and only weakly clustered. However, most real, meaningful interactions stay within a user's own community. Community membership is therefore hard to read from the topology (a user links to many groups), but clear in the interaction intensity (the edge weights). We study how to extract a local community from a seed user when these edge weights are probabilities rather than certainties. We use the standard negative-log transform to turn each edge probability into an additive cost, so a shortest-path search grows the community by selecting the most likely ties. Our method is an *annealing schedule* for a *structural heuristic* that we suppress while the set is small and phase in once the community is large enough to have a stable boundary. A single run produces one ordering of nodes, from which a budget-free *patience* policy selects the community without knowing its size in advance. We tune every method on training graphs and test on disjoint graphs. Under this fair comparison, the schedule helps in one specific scenario: when the community structure is weak, but the edge weights still carry a partial community signal. There it beats strong, individually tuned diffusion baselines by up to $+0.115$ in F1 on hard LFR graphs. The margin remains significant after correction for multiple comparisons. Elsewhere, it does no better, and often worse. We confirm the same effect on two real school contact networks, where it outperforms the best tuned baseline on both schools ($+0.041$ and $+0.044$ F1).

1 Introduction

Imagine that you open a social media feed and posts appear from across the whole platform: news, strangers, viral clips, and accounts of your friends. Through these connections, you can reach almost anyone. However, your attention stays with a far smaller group: the people and views you already agree with. You like their posts, reply to them, and share them, while the rest of the feed scrolls past.

This gap between who you can reach and who you engage with is the core of our problem. In graph terms, each user is a *node* and each interaction between users is an *edge*. What separates a group you frequently engage with from the rest of the people you rarely see is the *weight* of an edge: how often, and how strongly, two users interact. The tendency for similar people to interact more, known as *homophily*, makes ties inside a community stronger than ties that cross between communities. Therefore, a community can be hard to detect by only looking at the structure of the graph, where you are linked to nearly everyone. Yet, it stands out clearly in the interaction intensity. This is the structure of a polarised online community, whose formation has become a pressing social concern.

We study how to recover one such community. We start from a single *seed* user and grow the community outward, one node at a time. This approach avoids loading the entire network into memory. In contrast, global community detection methods must load the whole network at once, which is why they become computationally expensive when applied to massive datasets. The edge weights we observe are not certainties but *probabilities*: we estimate each tie from noisy interaction data, so its weight is the probability that the tie is a genuine within-community connection. Recovering the most likely community exactly is intractable, so we use heuristic search.

Two existing ideas offer partial solutions, but both are incomplete on their own. Local *diffusion* methods grow a community by simulating a random walk from the seed, but they treat edges as transition probabilities and tend to over-grow when communities are weakly separated. A shortest-path search instead treats edge weights as costs and follows the strongest ties exactly, but it has no sense of where the community ends. Our method builds on the shortest-path search and adds a clear way to find community boundaries.

To convert probabilities into costs, we use the standard *negative-log transform*. This way, we shift the objective from maximising probability to minimising additive cost. A shortest-path search like A^* can now solve this problem. The A^* algorithm chooses the next node to explore by adding a heuristic estimate to the known cost [Hart *et al.*, 1968]. This heuristic provides a computationally cheap way to guide the search.

The primary contribution of our work is how we guide the search: an *annealing schedule* for a *structural heuristic*. The heuristic measures the total edge weight crossing a set's boundary. It then compares this external weight to the edge weight contained inside the set. At the very start, the community is too small to have a clear structure, so we mostly follow the strong connections. As the community grows and its structure becomes clear, we rely more and more on the structural heuristic to decide who to add next. A conceptual contribution is the insight that community membership can hide in interaction intensity rather than in topology, together with a characterisation of when exploiting it pays off.

We frame this as a single question: *when does adding a structural heuristic to a shortest-path search improve local community detection in probabilistically weighted graphs?* The answer is narrow. The schedule helps in one specific scenario: when the community structure is weak, but the edge weights still carry a partial community signal. There it beats strong, individually tuned diffusion baselines. When the structure is already clear, or the weights are uninformative, it does no better, and often worse.

The remainder of this paper is organised as follows. We formalise the network model and the extraction problem in Section 2. We then review related work in Section 3. Section 4 presents our approach, and Section 5 reports our experiments on both synthetic and real social networks: the setup, our research questions, and the results. We discuss the findings in Section 6, reflect on responsible research in Section 7, and conclude in Section 8.

2 Background and Problem Description

In this section we formally define the problem. First, we introduce our network model, define community membership, and explain the negative-log transformation and why it is useful. Then we introduce two properties of a weighted network: how clear its community structure is, and how informative its edge weights are. Finally, we explain why a pure cost-minimisation search needs structural guidance.

We keep the core problem separate from the choices we later make to solve and evaluate it. The core problem is to recover a community starting from a seed in a probabilistically weighted graph. Conductance, the annealing schedule, the synthetic graph models, and the weight model are design and evaluation choices, which we introduce where they are used.

2.1 The Probabilistic Network Model

We model a social network as an undirected, probabilistically weighted graph $G = (V, E, P)$. The set V contains the users, which we call *nodes*. The set E contains the observed interactions between users, which we call *edges*. The function $P : E \to (0, 1]$ assigns a probability to each edge. We write $p(e)$ for the probability of edge e, and read it as the probability that e is a genuine *within-community* tie. We estimate $p(e)$ from how often, or how strongly, the two users interact, so a stronger interaction results in a higher probability.

We extract a local *community* $S \subseteq V$ around a known *seed* node $v_{\text{seed}} \in S$, growing it outward from the seed. A node belongs to a community if it has highly probable connections to the seed. The probability of a path is the product of its individual edge probabilities. Therefore, the strongest connection between two nodes is the path that maximises this product.

Finding exactly these most probable connections is difficult. The number of candidate node sets grows combinatorially with $|V|$. Multiplying many probabilities also leads to floating-point underflow: the product tends to zero.

We address both issues with one standard mathematical transformation. We replace each edge probability with its negative logarithm, which gives an additive *cost*:

$$C(e) = -\ln(p(e)) \tag{1}$$

A product of probabilities now becomes a sum of costs, so maximising joint probability becomes minimising total cost [Potamias *et al.*, 2010]. Therefore, a shortest-path search starting from the seed selects nodes by how strongly they connect to the seed, and grows the community along the most probable ties.

2.2 Community Structure and Weight Signal

A probabilistically weighted graph carries two kinds of information, and we define each one of them here. The synthetic models we use to vary them appear in Section 5.1.

Community structure. The *community structure* refers to how easy it is to separate communities based on the network topology alone. It only takes into account what edges exist, completely ignoring their weights. Structure is *strong* when edges within a community are much more likely to exist than edges between communities, so each community maps to a dense region in the graph. It is *weak* when within-community and between-community edges are similar in number, so it is very hard to distinguish the communities in the topology. A social graph in which a user is connected to almost everyone has weak community structure by this definition.

Weight signal. The *weight signal* is how much the edge weights reveal about what nodes belong to which communities. The weights carry no signal when an edge's probability is independent of whether the nodes that share that edge belong to the same community or not. They carry a *partial* signal when within-community ties tend to be stronger than cross-community ties, and a *strong* signal when the weights almost perfectly separate the two. Homophily, previously defined as the tendency of similar users to interact more, is what produces this signal, by making within-community ties stronger. In our synthetic experiments we control the signal strength with a parameter η, where $\eta = 0$ leaves the weights uninformative and larger η makes them increasingly reveal the community.

2.3 The Cost-versus-Structure Problem

A shortest-path search follows the strongest ties, adding nodes in order of accumulated path likelihood. This works inside a community, but path cost has no notion of the set's boundary. A single strong edge that bridges into a neighbouring community looks just as attractive as an edge into the community's core. Thus, relying only on cost interleaves genuine members with nodes from adjacent groups, and worsens the ordering it produces.

Avoiding this requires a signal about the set, not just the next edge. A community keeps most of its edge weight inside the set, with little weight crossing to the rest of the graph. Sometimes, a new node has a very strong connection to the group, but adding it would grow the boundary too much. When this happens, we should push it lower in the order rather than treat it like a core member. This structural rule forces the algorithm to keep the search focused on the dense core of the community, so the algorithm produces a cleaner order and selects actual members first.

This structural guidance is unreliable on a small set. This is because small sets contain very few internal edges, so adding just one new connection can derail the search completely. As a result, structural guidance must be suppressed while the set is small, and phased in once the community is large enough to have a stable boundary.

Therefore, we face a trade-off. The algorithm sometimes ignores the strongest connection to keep the community tightly connected. This means sacrificing some path probability for better overall structure. Inside the community, both of these goals point in the same direction. They only conflict when the search reaches the border. Yet, the boundary is the exact place where this decision is important. Our method aims to find a *high-likelihood* community rather than a maximum-likelihood one. The rest of the paper investigates when this trade-off is justified.

3 Related Work

Our work combines three main research areas: local community detection, search on probabilistic graphs, and the measurement of polarisation. We review each in turn, then state the gap that none of them closes on its own.

Local community detection. We build on local methods rather than global ones. Global methods recover every community at once. They need the whole network in memory, so they become expensive as the graph grows. Local methods avoid this: they start from a seed node and grow one community outward. The standard local methods are diffusion-based. Approximate Personalised PageRank (APPR) pushes probability mass out from the seed and ranks nodes by a degree-normalised score [Andersen *et al.*, 2006]. The Heat Kernel method instead spreads heat from the seed for a fixed time [Kloster and Gleich, 2014]. Both follow random-walk dynamics: they treat edges as transition probabilities, not as path costs. As a result, both tend to over-grow or merge groups when communities are only weakly separated.

Search on probabilistic graphs. The negative-log transform turns the most likely path into a shortest path, which Dijkstra and A^* then find exactly [Potamias *et al.*, 2010]. A^* is a classic best-first search algorithm for shortest-path and routing problems. To our knowledge, it has not been adapted to extract local communities in social networks.

Polarisation. Existing work measures polarisation at the level of the whole network, quantifying controversy between two partitioned sides [Garimella *et al.*, 2018], or modelling attitude-and-tie dynamics across the whole graph [Mepham *et al.*, 2025]. It rarely frames the task as we do, as a local, seed-based extraction that reads community membership from informative, homophily-based edge weights.

The gap. Our problem needs three things at once: cost-based expansion along the strongest ties, structural guidance that keeps the search from crossing the community boundary, and protection against the unstable structural signal on the small sets early in the search. No existing approach provides all three. Diffusion models ignore edge costs, pure cost-minimisation has no structural-awareness so will cross the boundary, and naive structural guidance is unreliable on the small sets that arise early in the search, as we explain in Section 2. We close this gap with an annealed structural heuristic, and we test it to find out exactly when it improves accuracy.

4 Approach

Recall the trade-off from Section 2: grow the community along the most probable ties, while adding enough structural awareness to keep the search from wrongly crossing the boundary of the community. Our key idea is a search priority that balances the two, annealing the structural term in as the set grows. We first define this priority, then the annealing schedule that shapes it, and finally the shared rule that turns the resulting node ordering into a community.

4.1 Annealed A^* Priority

We first define how our method orders nodes. Following the A^* rule, we expand the *frontier* node with the smallest priority:

$$f(n) = g(n) + \alpha(k)\, h(n), \tag{2}$$

where $k = |S|$ is the current community size. The path cost $g(n)$ is the total negative-log probability from the seed to n. A greedy rule would look only at the candidate node, but $g(n)$ remembers the whole path taken to reach it. Minimising g maximises path likelihood. The heuristic $h(n)$ measures structure: it is the ratio of the weight that leaks out of $S \cup \{n\}$ to the weight inside it. For a set S, let $W_{\text{in}}(S) = \sum_{e \in E(S)} p(e)$ be its internal edge weight and $W_{\text{out}}(S) = \sum_{e \in \partial S} p(e)$ its boundary weight, where $E(S)$ is the set of edges with both endpoints in S, and ∂S the set with exactly one. Then

$$h(n) = \frac{W_{\text{out}}(S \cup \{n\})}{\max\big(W_{\text{in}}(S \cup \{n\}),\, 1\big)}\, C_{\max}, \tag{3}$$

where the floor of 1 avoids division by zero on a set with no internal edges, and the largest edge cost $C_{\max}$ rescales the ratio into cost units so the two terms in $f(n)$ are comparable. The ratio uses only the edges around S, never the volume of the whole graph, so it stays *local*. The weight $\alpha(k)$ sets how much this structural term counts. We define it in the next section.

Non-stationary heuristic. The heuristic is *non-stationary*, because $h(n)$ depends on the growing set S. Priorities already stored in the queue become stale as S changes. We handle this with *lazy re-evaluation*. When we pop a node, we recompute its priority against the current S. If its recomputed priority is higher, we re-insert the node instead of adding it. This keeps the order consistent without rebuilding the queue. Because $h(n)$ does not lower-bound the remaining cost and is non-stationary, the search does not inherit A^*'s optimality guarantees. It is a best-first search with an A^*-form priority, and finds a high-likelihood community rather than a maximum-likelihood one.

4.2 The Annealing Schedule

The schedule $\alpha(k)$ is the core of our method. The structural heuristic is less relevant when S is small, and using it early just adds noise. Therefore, we suppress the heuristic at the start. We phase it in with a sigmoid schedule centred at a warm-up size w:

$$\alpha(k) = \frac{\alpha_{\max}}{1 + e^{-s(k-w)}}, \tag{4}$$

where $\alpha_{\max}$ is the full heuristic weight and s sets how sharply it switches on. For $k \ll w$ the search behaves like Dijkstra: it follows the strongest ties. Once S is large enough to have a stable boundary, the heuristic switches on and penalises external edges. The sigmoid function is a deliberate design choice. The phase-in needs a schedule that stays near zero while the set is small, then rises once the signal becomes reliable. A step function is too abrupt: it turns the heuristic on all at once at a single size. A linear function is non-zero from the first step, so it never fully suppresses the heuristic early. Only the sigmoid meets both requirements.

4.3 From Ordering to Community

A single run produces one node ordering. To turn that ordering into a community, we use *conductance*, the standard measure of how well a set is separated from the rest of the graph. Reusing W_{in} and W_{out} from above, the *volume* of S is $\text{vol}(S) = 2\,W_{\text{in}}(S) + W_{\text{out}}(S)$, and with $\text{vol}(V)$ the volume of the whole graph the conductance is

$$\phi(S) = \frac{W_{\text{out}}(S)}{\min\big(\text{vol}(S),\ \text{vol}(V) - \text{vol}(S)\big)}. \tag{5}$$

Conductance lies between 0 and 1. A low value means a tight community, with most of its edge weight inside. The structural heuristic h and conductance ϕ play separate, non-overlapping roles: h *guides* the search (the term added to g in the priority), while ϕ *scores* the resulting community and decides where to *stop*.

From one ordering and its conductance values we obtain two stopping policies at no extra cost. *Sweep* takes the prefix of lowest conductance: the classic cut and the rule the diffusion baselines use natively [Andersen *et al.*, 2006]. *Patience* is budget-free: it stops once conductance has not improved for a small, fixed number of steps and returns the best prefix, without knowing the community size in advance. We score every method with both policies on the same weighted conductance, which keeps the comparison fair.

5 Experiments

We evaluate our method on synthetic graphs, where we can vary the topology and the weights independently, and then confirm the findings on two real-world networks. We first describe the setup, then state our research questions. Finally, we present and discuss the results.

5.1 Experimental Setup

Synthetic graphs let us vary the two properties from Section 2 (community structure and weight signal) independently, which real data does not allow. By varying them, we can isolate exactly when our method helps. We describe the synthetic and real data, the baselines, how we tune them, the software, and how we evaluate them.

Synthetic graphs. We generate the topology and the edge weights separately, so we can control the two properties independently. For topology we use two models. The *SBM* (Stochastic Block Model) [Holland *et al.*, 1983] has 100 nodes, split into two equal blocks. We vary the block-connection probabilities from trivial (clear communities) to very hard (weakly separated). The *LFR* (Lancichinetti–Fortunato–Radicchi) benchmark has 500 nodes and a *mixing parameter* $\mu \in \{0.1, \ldots, 0.5\}$ that sets how blended the communities are [Lancichinetti *et al.*, 2008]. LFR also provides power-law degrees and varied community sizes, like real social graphs.

Edge weights. We draw edge probabilities from a latent-space homophily model [Hoff *et al.*, 2002]. Each node i gets a feature vector $x_i \sim \mathcal{N}(\mu_{c(i)}, I)$ drawn around its community's hidden centre $\mu_c \sim \mathcal{N}(0, \eta^2 I)$ in $d = 8$ dimensions, where I is the $d \times d$ identity matrix. An edge's probability is the Gaussian similarity of its endpoints, $p(i,j) = \exp(-\|x_i - x_j\|^2 / 2\ell^2)$ with $\ell = \sqrt{2d}$. The parameter η controls how strongly the weights reveal the community: at $\eta = 0$ the weights are pure noise, while at $\eta = 4$ they almost perfectly reveal the community. The within/between signal ratio works out to $\exp(\eta^2/2)$, set entirely by η, so d is a fixed constant rather than a tuned parameter.

Method	Hyper-parameter	SBM	LFR
A^*	$\alpha_{\max}$, w	10, 30	10, 35
Heat Kernel	t (truncation K)	20 (57)	10 (39)
APPR	c, ε	0.05, 10^{-6}	0.05, 10^{-7}
Dijkstra	—	—	—

Table 1: Tuned hyper-parameters, selected on the training graphs under the patience policy. Fixed and shared across all methods: extraction cap $= 150$, a patience window of ten steps, and the sigmoid steepness $s = 0.2$. Heat Kernel's truncation order K is scaled with t. The K-12 experiments reuse the LFR-tuned settings.

Real-world networks. We confirm the findings on real networks: the two Pittsburgh high schools (C-HS and P-HS) from a public K–12 contact-network study [Guclu *et al.*, 2016]. In these high schools, students make contact across all grades, so the topology reveals little about the grade communities. However, contacts within a grade last far longer than contacts across grades, so the grade is a strong factor in the connection weights. We therefore define each community by the student's grade, and take the edge weight to be the total contact time. A duration is not a probability, so we map it to one with a parameter-free, scale-invariant *rank* transform: an edge's probability is its duration rank divided by the number of edges, $p(e) = \text{rank}(e)/|E|$, with ranks from 1 (shortest) to $|E|$ (longest). We seed every method from every node and report the average per school.

Baselines and tuning. We compare against three baselines: Dijkstra, APPR [Andersen *et al.*, 2006], and Heat Kernel [Kloster and Gleich, 2014]. All run on the same probabilistically weighted graph. APPR spreads probability mass from the seed by a random walk, controlled by a restart probability c, written α in the original formulation [Andersen *et al.*, 2006], where a higher c keeps the walk closer to the seed, and an approximation tolerance ε (smaller ε explores more nodes). Heat Kernel has a single diffusion time t, with its power series truncated at order K. We tune each method's hyper-parameters on a training set of graphs and evaluate on a disjoint set, so no method sees its test data (Table 1). We fix three constants rather than tuning them: the sigmoid steepness s, the patience window of ten steps that all methods share, and an extraction cap of 150 nodes (comfortably above the size of the communities the patience policy recovers, so it does not affect the results).

Software. We implement all methods in Python 3.13. We generate Stochastic Block Model graphs with python-igraph 0.11.8 and LFR graphs with NetworkX 3.6.1. We also use NumPy 2.4.6 for mathematics and Matplotlib 3.10.9 for the figures. The baselines (Dijkstra, APPR, and Heat Kernel) and our annealed A^* search are implemented from scratch, following the original papers [Andersen *et al.*, 2006;

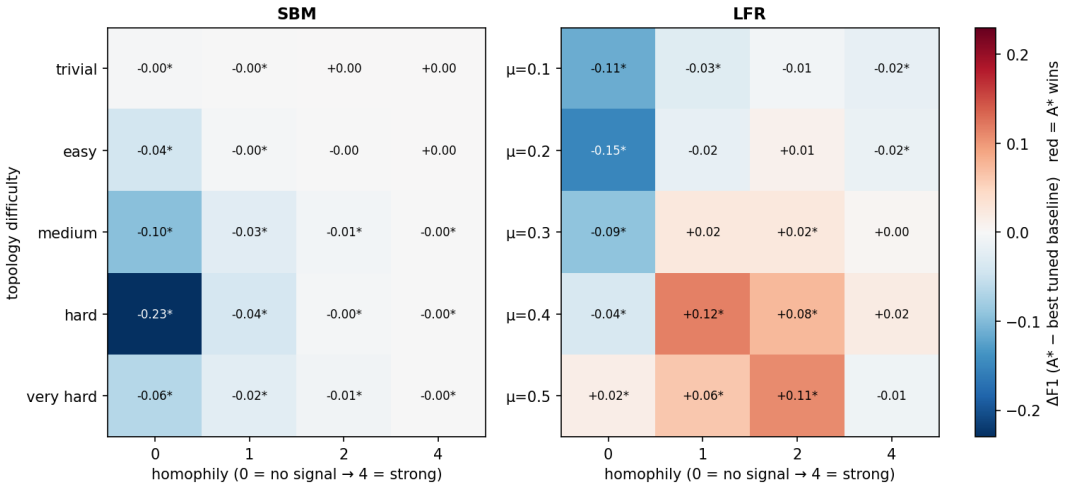


Figure 1: Win map on the synthetic benchmark, under the budget-free patience policy. Each cell shows the signed F1 gap between annealed A^* and the best tuned baseline (positive = A^* better), across community structure (rows) and weight signal η (columns), for SBM (left) and LFR (right). Cells are shaded blue (A* worse) through white (tie) to red (A* better), and the signed value is printed in each cell so the sign stays legible in greyscale; a $*$ marks cells that remain significant after Benjamini–Hochberg FDR correction. A^* loses across SBM and wins on LFR at the hardest mixing (μ=0.3–0.5), where the largest gains need a partial weight signal (η=1–2).

Kloster and Gleich, 2014]. We do not rely on an external community-detection library.

Evaluation. We score each recovered community by its *F1* against the seed's true community: the harmonic mean of precision (the fraction of recovered nodes that are correct) and recall (the fraction of the true community recovered). On the synthetic graphs we seed each method from a sample of nodes (20 per SBM graph, 10 per LFR graph, spread across communities). On the real networks we seed from every node. We call each per-seed F1 the *seed score*.

Statistical testing. We generate 30 independent graphs per (structure, η) cell. Seeds within one graph share its structure, so they are not independent. Our unit of analysis is therefore the graph, not the seed. We average the seed scores within each graph, then compare methods across the 30 graphs with a sign-flip permutation test and a bootstrap confidence interval (CI), correcting for multiple comparisons across cells with the Benjamini–Hochberg procedure [Benjamini and Hochberg, 1995].

5.2 Research Questions

The experiments in this section are designed to answer four research questions:

- **RQ1.** Does our annealed search recover communities more accurately than strong, individually tuned diffusion baselines?
- **RQ2.** Under which conditions does it help?
- **RQ3.** Do the annealing schedule and the stopping rule behave as designed?
- **RQ4.** Does the effect carry over from synthetic graphs to real networks?

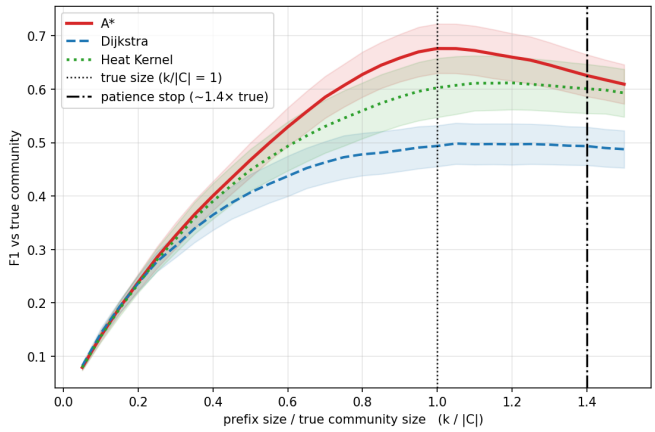
In summary, the answer is narrow but consistent. Our method beats the best tuned baseline only when the community structure is weak and the weights carry a partial signal. It wins by up to $+0.115$ F1 on hard LFR graphs (RQ1, RQ2), and does no better, often worse, when the structure is already clear or the weights are uninformative. We also confirm that the annealing schedule and stopping rule behave as designed (RQ3), and the same pattern holds on the two real schools (RQ4: C-HS $+0.041$, P-HS $+0.044$).

5.3 Experimental Results

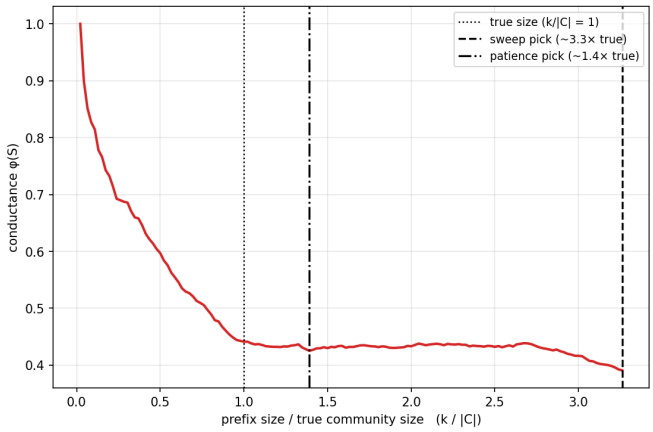
We answer the four questions in turn. We first compare accuracy against the tuned baselines on the synthetic benchmark (RQ1), then map where the gain appears (RQ2). Next, we discuss whether the method behaves as designed (RQ3) and finally confirm the pattern on real contact networks (RQ4).

RQ1: accuracy against tuned baselines. Our method wins only in a specific scenario, and is no better, often worse, elsewhere. On SBM graphs it wins no cells: a tuned diffusion baseline matches or beats it everywhere, often significantly (Figure 1). On LFR graphs it wins a clear, FDR-significant set of cells. Under the budget-free patience policy it beats the best tuned baseline in six of twenty cells, concentrated at the hardest mixing ($\mu = 0.3$–0.5). The largest gains are located where the topology is hard *and* the weights carry a partial signal: $+0.115$ at μ=0.4, η=1 (over Heat Kernel) and $+0.109$ at μ=0.5, η=2 (over Dijkstra), both surviving FDR correction. The advantage comes from a better node *ordering* rather than a lucky stopping rule. This is clear from the growth trajectory in Figure 2a: A^* leads the baselines across the prefix range, so its mean F1 sits above the baselines at essentially all cut points, not only at the patience stop.

RQ2: under which conditions. The win map (Figure 1) shows exactly where the gain is. The advantage requires *both*



(a) Ordering: F1 of the prefix vs. prefix size, LFR (μ=0.4, η=1). A^* leads across the prefix range. Dotted = true community size; dash-dot = mean patience stop; shaded band = 95% CI.



(b) Stopping: conductance vs. set size, same graph. A^*'s conductance keeps descending, so the global sweep over-grows, while patience stops near the true size.

Figure 2: Why A^* wins on weak structure: a better *ordering* (left) and a stopping rule that does not over-grow (right).

a weak topology and an informative weight signal. When the structure is already clear ($\mu < 0.3$) the topology alone is enough and the baselines match our method. When the weights are absent (η=0) the structural heuristic mostly reads noise and A^* is worse across the η=0 column, with one exception at the hardest topology (μ=0.5), where a small $+0.016$ win appears to come from the topology, not from the weights. When the weights are very informative (η=4), the methods are close, though A^* is still significantly worse at the easiest topologies (μ=0.1–0.2). The gain therefore concentrates where the topology is weak and the weights carry a partial signal (η=1–2, μ=0.3–0.5).

RQ3: behaving as designed. Two observations confirm the method behaves as intended. First, accuracy peaks when the warm-up size w sits below the true community size and falls toward plain Dijkstra when the heuristic is suppressed for too long, supporting the annealing idea and confirming the schedule behaves as intended. Second, the conductance curve (Figure 2b) explains the stopping behaviour: A^*'s ordering keeps lowering conductance, so the standard global-min sweep runs far past the true boundary ($\sim 3.3\times$), while the patience rule stops near the true size ($\sim 1.4\times$).

RQ4: confirmation on real networks. The pattern holds on real contact networks. On the two Pittsburgh high schools, A^* recovers grade communities more accurately than every baseline under the patience policy (Figure 3): it reaches F1 0.643 on C-HS and 0.570 on P-HS, beating the best baseline (Heat Kernel) by $+0.041$ and $+0.044$. With only two schools, this is more of a qualitative confirmation rather than a significance test. What matters is that the *direction* and *size* of the gain match the behaviour from synthetic data. The reason matches too: grade barely determines *which* contacts occur (within- vs cross-grade densities are 0.81 vs 0.73 at C-HS and 0.56 vs 0.37 at P-HS), but strongly determines *how long* they last (within- vs cross-grade mean durations are 46.8 vs 20.7 at C-HS, $\approx 2.3\times$, and 30.8 vs 9.7 at P-HS, $\approx 3.2\times$).

6 Discussion

Our results are narrow but precise. In this section we discuss when the heuristic helps, why those conditions are not always enough, and what are the limitations of our approach. Finally, we outline the limitations of our study and suggest future work.

When the heuristic hurts. The annealed heuristic lowers accuracy whenever either of two conditions is missing. The first is a weak topology: when communities are already well separated, the diffusion baselines find the boundaries on their own, and the structural heuristic adds nothing. The second is an informative weight signal: when the weights are noise, the structural heuristic derails the search and the method performs worse than the baseline. The same pattern appears on the two real schools, where the grade is weak in the topology but strong in the contact durations.

Necessary but not sufficient. Even when both conditions are met, the heuristic is not guaranteed to win, which can be seen on SBM. Its hardest graphs are beaten by the baselines on every cell. SBM graphs have equal-sized blocks and a uniform structure, so a standard method can simply tune its diffusion time to fit this uniform scale perfectly. There is no structural variety for our adaptive method to exploit. LFR is different: its communities vary in size and its degrees follow a power law, so no single diffusion setting fits every community.

Limitations. Our study has clear limitations. First, we test on only two real graphs, so our real-world results show a general trend, not a statistically proven rule. Second, we also tune our hyper-parameters on synthetic graphs and transfer them, rather than tuning directly on the real data. This is because we only have two real graphs. Third, the school networks fit the exact conditions we target, but they are not the divided online networks that originally motivated this research.

Future work. These limits point to several areas for future work. The most obvious next step is to test the method on

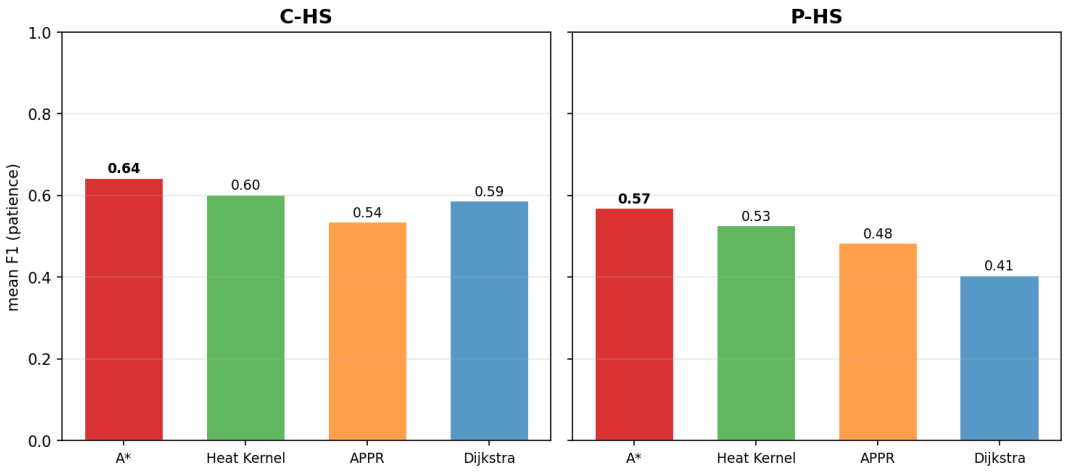


Figure 3: Mean F1 under the patience policy (rank transform), for each method per school. A^* leads every baseline on both.

divided online networks. In that setting, we would use shared opinions as the edge weights instead of contact time, but the core algorithm would stay exactly the same. Other natural next steps include tuning the settings directly on real data, and also expanding the method to extract multiple communities at once instead of just one.

7 Responsible Research

In this section, we address reproducibility, data ethics, and our use of generative AI.

Reproducibility. We release all code, the fixed random seeds, and the exact train / test graph split.[1] We also report every hyper-parameter (Table 1). Anyone can re-run the full pipeline (tuning, evaluation, figures) and obtain the numbers in this paper. The tuning and evaluation graphs are disjoint, so no method is ever tuned on its test data. The patience window is fixed at ten steps, shared across all methods and not tuned per run.

Data. The K–12 contact networks are public and anonymised. The labels of the nodes are grade identifiers, with no names or personal content. We use only the contact durations and the grade labels, and we don't make any attempt to identify any individual. The synthetic graphs don't contain any real people. We use all data only to measure how well a method recovers a community.

Generative AI. We used generative AI throughout this project and we list our prompts, with each response and how we used and checked it, in Appendix A. We used Gemini 3.1 Pro and Claude Opus 4.7 and 4.8 on the paid tier: Gemini for clarifications and concept explanations, Claude Opus 4.7 for code changes (in the terminal), and Claude Opus 4.8 for the final revision of the paper. We cannot give a complete log, as we used several chats for over roughly two months across these tools and a coding terminal. However, the prompts in the appendix are representative of how we used AI in our research. In hindsight, keeping a single log of AI interactions from the start would have made this easier to document, and we would do so in future work. We used the tools for five clear kinds of task. For *understanding*, we asked them to explain concepts that we met in the literature: the APPR and Heat Kernel parameters, the FDR significance, the sweep cut, the stochastic block model, and the F1 metric. For *literature*, we used them to find relevant local maximum-likelihood methods and provide search terms. After that, we discovered relevant papers for our research and read them ourselves. For *writing*, we used AI tools to shorten sentences and to suggest simpler wording in some of our sentences. For *figures and code*, we used AI to improve plots (the win-map heat map) and to help us find bugs in our implementation (stale heuristic values). For *revision*, we used Claude Opus 4.8 to review the paper section by section against a checklist document that we wrote ourselves. We used it to point out issues and proposing rewrites for structure and clarity. We checked every suggestion against our own results and edited or rejected each before applying it.

These uses are appropriate because all the output we received was one that we could check directly. We verified each concept and explanation, and confirmed the definitions against trusted references rather than the model's text. We read every source in full ourselves and nothing was cited on the basis of an AI summary. For the figures, we reimplemented the plots on our own results and checked the displayed values against our own data. For the code bugs, we used AI to locate the error, reason over the proposed fix ourselves, and decided to apply its fixes only after we were sure that they were correct and only on small, isolated fragments of code. We then re-ran the experiments to confirm the fix actually landed correctly.

The research is our own. We formulated the research question and its scope, made the design choices for the method and implemented it. We also designed, ran, and interpreted the experiments. In terms of writing, we reviewed and revised every AI-generated text and every rephrasing suggestion before relying on it. We discarded those that did not fit, for instance an informal analogy that was unsuitable for an

[1] https://github.com/andibardas/research-project

academic paper, and a rephrasing whose flow we considered to be too poor to use.

8 Conclusion

We adapted A^* search to extract high-likelihood communities from probabilistically weighted graphs. We converted edge probabilities into additive negative-log costs, so that the search expands along the most likely ties, and added an annealed structural heuristic that switches on only once the node set is large enough to have a stable boundary. A single run produces a node ordering, from which a budget-free policy extracts the community without knowing its size in advance.

We tuned every method on training graphs and evaluated on a disjoint test set, so no method saw its test data. Under this fair comparison, our annealed search wins on graphs whose community structure is weak but whose edge weights still carry a partial signal. There it beats strong, individually tuned diffusion baselines, an advantage we confirm on real school contact networks. Elsewhere, when the structure is already clear, or the weights are uninformative — it does no better, and often worse.

More generally, our result shows that community membership can hide in interaction intensity rather than in topology. A natural next step is to apply this in the setting that motivated the work: polarised online networks with opinion alignment in place of contact duration as the weight signal.

References

[Andersen *et al.*, 2006] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 475–486, 2006.

[Benjamini and Hochberg, 1995] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[Garimella *et al.*, 2018] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Quantifying controversy on social media. *ACM Transactions on Social Computing*, 1(1):1–27, 2018.

[Guclu *et al.*, 2016] Hasan Guclu, Jonathan Read, Charles J. Vukotich Jr., David D. Galloway, Hongjiang Gao, Jeanette J. Rainey, Amra Uzicanin, Shanta M. Zimmer, and Derek A. T. Cummings. Social contact networks and mixing among students in K-12 schools in Pittsburgh, PA. *PLOS ONE*, 11(3):e0151139, 2016.

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

[Hoff *et al.*, 2002] Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

[Holland *et al.*, 1983] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[Kloster and Gleich, 2014] Kyle Kloster and David F. Gleich. Heat kernel based community detection. In *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 1386–1395, 2014.

[Lancichinetti *et al.*, 2008] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[Mepham *et al.*, 2025] K. Mepham, A. Vörös, and C. Stadtfeld. Network polarization: The study of political attitudes and social ties as dynamic multilevel networks. *Network Science*, 13:e7, 2025.

[Potamias *et al.*, 2010] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. k-nearest neighbors in uncertain graphs. *Proceedings of the VLDB Endowment*, 3(1):997–1008, 2010.

A Generative AI prompts

These are the main, non-trivial prompts we used, a short summary of each response and how we used and checked it.

1. Rephrasing: "Make this use first person plural and cut the sentences shorter (one thought per sentence)" [followed by a draft paragraph].
Response (summary). Rewrote the outline as short, first-person-plural sentences, one thought each.
Use & outcome. The result was too robotic and lacked any normal writing flow. We did not use the AI's version, but it helped us better understand how to express our ideas more concisely.

2. Explain Concept: "What are the hyper-parameters t for Heat Kernel and α for APPR?"
Response (summary). Explained t as Heat Kernel diffusion time (small t stays local, large t becomes global), and α as APPR's restart/teleport probability (high α stays tight around the seed, low α explores deeper).
Use & outcome. We read the APPR and Heat Kernel papers first, but we were unsure of our understanding of the parameters. The explanation from AI was reasonable and we checked it by re-reading the parts where these were introduced in the papers. The two explanations were consistent. The AI made the parameters easier to reason about, which also helped us when tuning them in our code.

3. Explain Concept: "What is FDR significance?"
Response (summary). Explained FDR as controlling the expected proportion of false positives among the results called significant.
Use & outcome. We came across the notion of FDR significance and thought it would be a good use of AI to get a quick insight into whether it was indeed a useful concept for our research. It turned out to be useful, so we took it into consideration and used it, after consulting some more credible sources.

4. Rephrasing: "Rephrase this to use simpler words" [followed by the sentence].
Response (summary). Offered several rewrites, including an analogy.
Use & outcome. The AI's suggestions were somehow helpful, but we used only parts of its rephrasing and then integrated them into the report. These were only small parts, especially synonyms. In the case of synonyms, we think the AI is helpful, but the analogy was too informal and inappropriate for an academic paper.

5. Explain Concept: "How do APPR and Heat Kernel use a sweep cut?"
Response (summary). Described how the diffusion produces node scores, scores are degree-normalised, nodes are added in ranked order, and the boundary is taken at the conductance minimum. Also outlined that the baselines only score and the sweep cut turns scores into a community.
Use & outcome. We used this to better understand the sweep-cut step while reading the APPR and Heat Kernel papers. We checked the AI summary against those papers and they matched. The main value was clarifying that the baselines only produce a node ordering and that the cut is a separate step. This was also useful when we implemented the code for them, because we were able to keep the code much cleaner with this separation in mind.

6. Explain Concept: "Explain the stochastic block model."
Response (summary). Explained SBM as a generative model where edge probability depends on block membership (dense within-block, sparser between-block) and gave the formal parameters.
Use & outcome. We discovered this term in one of the initial papers we read for this project and used AI to give us an overview before reading more sources on the SBM. We verified the definition and discovered that there are Python libraries that implement it natively. It became part of our experimental setup.

7. Explain Concept: "How does F1 work for SBM and LFR?"
Response (summary). Explained how precision, recall, and F1 behave on LFR and SBM, and gave their definitions.
Use & outcome. This made us understand why F1 is appropriate to use for both LFR and SBM, especially for our budget-free stopping policies where an algorithm can achieve high recall by just selecting all the nodes. We confirmed the precision, recall, and F1 definitions we received from AI against other references online rather than trusting only the AI text before using them in the evaluation. We ended up using this concept, and the explanation we received was useful. It helped us move from a fixed-budget policy to a more realistic budget-free one.

8. Find Relevant Literature: "Are there any maximum-likelihood-estimator algorithms for local community extraction?"
Response (summary). Said that exact local MLE is NP-hard and rare. Also mentioned that global MLE uses SBM/DC-SBM via MCMC.
Use & outcome. We used this prompt only to have a brief overview of what local maximum-likelihood algorithms currently exist. The contribution of AI in this case is that it gave us search terms (SBM, DC-SBM, the NP-hard problem) that later helped us find relevant literature. It also prevented us from spending too much time on trying to work on a local MLE, which is indeed intractable without a heuristic.

9. Improve Figures: "Can I plot these values in a more meaningful way? How?"
Response (summary). Gave us feedback on the original trajectory plot (wrong horizontal axis, overlapping curves, no error bands, mixed concepts) and proposed clearer figures with code.
Use & outcome. The most useful suggestion was a heat map, which we adopted as the win map in our report. It shows the exact F1 gain over the best baseline in each cell of the topology $\times$ weight-signal, which a line plot could not achieve as clearly. We reused the code provided for the heat map, but we introduced our empirical results and checked what was shown on the plot against the actual values.

10. Code Debugging: "This is what I get. Is there a bug in my A^* implementation" [shared a result plot showing A^*'s accuracy is very low compared to Dijkstra].
Response (summary). Identified a bug in how the search priority was computed. The structural heuristic was evaluated only when a node was added, not also when it was popped from the queue, which led to stale values and wrong extractions.
Use & outcome. The AI's answer was correct. We were indeed evaluating the structural heuristic only when adding a node, not also when popping it, so the priorities were stale. We fixed this by re-evaluating a node's priority against the current set when it is popped (the lazy re-evaluation described in the paper). We confirmed the fix by re-running the experiments and the accuracy increased to the expected level.

11. Revision: "Review this section against the checklist and tell me if there are any problems. Give me some possible rewrites to improve. Keep the changes minimal." [followed by a section of the paper and a written feedback checklist].
Response (summary). Gave feedback on terminology, structure, and over-claims, while also suggesting rephrasing and removals of some figures and tables.
Use & outcome. We took each suggestion as a proposal. Often the AI suggested minor changes to the text, which we considered not important enough to implement. The AI helped us revise the paper more systematically because we provided it with a document with points to improve alongside what was already in good shape. All decisions regarding content and wording were ours and we later re-read the entire paper to ensure that local changes did not break the general flow and logic of our paper.