FAST TUNING-FREE DENSEST k-SUBGRAPH EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we investigate the densest k-subgraph (DkS) problem, which asks for the k vertices of a graph that induce the largest number of edges. DkS is NP-hard and beyond the reach of exact methods at the scale of real networks, so practical solvers work with a continuous relaxation (in the sense of losing tightness) instead. We build our solver on the diagonally loaded relaxation over the capped simplex, which is tight below the clique number, and we prove that a light loading keeps the graph itself in control of the search direction, where the classical heavy loading does not. The difficulty is thereby displaced from modelling to optimization, since the relaxed landscape is non-convex and densely populated with stationary points of very unequal quality. Our main contribution is F^3-DkS (Fast, Free of eigendecomposition, Free of hyperparameters for Densest k-Subgraph extraction), built on the minorization–maximization principle, with a family of separable minorizers indexed by a vector of weights and maximized over the capped simplex in closed form. Every member ascends monotonically to a Karush–Kuhn–Tucker point, and yet individual members reach markedly different stationary points, so that the weights select the solution rather than merely the step length. Exploiting this freedom, we single out a degree-adaptive member requiring neither tuning nor any eigenvalue computation, and we show that shrinking the weights lengthens every step while an inexpensive run-time test preserves monotone ascent. F^3-DkS completes each iteration in time linear in the size of the graph, and extensive experiments on real-world networks show that it outperforms the state-of-the-art solvers it is compared against.

1 INTRODUCTION

Extracting a small, tightly connected group of vertices from a large network is one of the recurring primitives of graph mining [13]. Communities in social and collaboration networks [33, 9, 35], regulatory motifs in biological networks [12, 30], coordinated behaviour in e-commerce and financial graphs [5, 15], and bursts of correlated activity in media streams [2] are all naturally phrased as the search for a subgraph whose internal density stands out against an otherwise sparse background. The nature of the resulting optimization problem is decided by how the size of that group is controlled, as a recent survey of the area makes clear [20]. When the size is left free and the average internal degree is taken as the objective, the problem is polynomially solvable by a maximum-flow computation [14], and a simple peeling procedure recovers a solution of guaranteed quality in linear time [8, 3]. In most applications, however, the size of the group is dictated by the downstream task rather than by the data, and the practitioner asks for exactly k vertices.

Prescribing the cardinality changes the problem completely. Given an undirected, unweighted and simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ vertices and $m = |\mathcal{E}|$ edges, the densest k-subgraph (DkS) problem asks for the subset of exactly k vertices inducing the largest number of edges. It is NP-hard and admits no polynomial-time approximation scheme under standard assumptions [17], and under the exponential time hypothesis it cannot be approximated to within an almost polynomial factor [26]. The strongest guarantee available at present is an $O(n^{1/4+\epsilon})$ approximation [6], whose factor grows with the size of the graph and which therefore offers little guidance for the design of a practical solver. What has proved effective on real instances is a different strategy, in which the combinatorial constraint is relaxed (in the sense of losing tightness), the resulting continuous program is optimized, and the solution is rounded back to a vertex set.

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.

The methods that follow this strategy differ mainly in the relaxation they choose and in what is sacrificed by choosing it. Spectral procedures form the earliest family. The truncated power method alternates a power iteration with a hard thresholding to k nonzeros [37], and a rank-one procedure reads a k-subset directly off the leading eigenvector of the adjacency matrix, refining it afterwards through a bilinear optimization over a low-rank factorization [28]. Both summarize the graph by a small number of leading directions, and the combinatorial structure that DkS is concerned with is largely lost in that summary. A second family replaces the objective or the feasible set by a convex surrogate, for instance through semidefinite or Lovász-theta relaxations. The per-iteration cost of these programs grows far too quickly for the graph sizes of interest here, and the recovery guarantees offered in return hold only under planted models [1, 18]. A third family keeps the quadratic objective $\mathbf{x}^\top \mathbf{A}\mathbf{x}$ intact and relaxes only the binary constraint, maximizing over the capped simplex $\mathcal{C}_k^n = \{\mathbf{x} \in [0,1]^n : \mathbf{1}^\top \mathbf{x} = k\}$, the convex hull of the indicator vectors with exactly k ones [23, 25, 24]. Its difficulty is that a maximizer of a general quadratic form over a polytope need not be a vertex, so that the relaxation is not tight and the rounding step may undo what the optimizer has achieved.

Tightness of this last relaxation can be restored by *diagonal loading*, that is, by replacing $\mathbf{A}$ with $\mathbf{A} + \lambda \mathbf{I}$. Since the added term is constant on the binary feasible set, the combinatorial problem is left unchanged, while on the relaxed set the continuous maximizers are drawn towards the vertices of $\mathcal{C}_k^n$. The classical requirement $\lambda \geq -\lambda_{\min}(\mathbf{A})$ makes the loaded objective convex and the relaxation exact, but it is of no practical use, because $-\lambda_{\min}(\mathbf{A})$ is large on real networks and a heavy loading produces a landscape in which first-order methods stop making progress almost immediately. What the loading should be, and what a solver gives up by setting it too high, are therefore left open by the formulation itself.

Even with the loading settled, tightness would decide only what the continuous problem is, and would say nothing about how difficult that problem is to solve. The loaded objective remains non-convex, and its landscape contains a great many stationary points, most of which correspond to poor subgraphs. This is not an artifact of the relaxation but a reflection of the combinatorial structure of DkS itself, since every locally maximal dense set induces a basin of attraction of its own. Two runs of the same solver, differing in nothing but their starting point, therefore routinely return subgraphs of very different quality, and a convergence proof that establishes stationarity of the limit remains silent about which one is reached. The operative question is consequently not which relaxation to choose, but *where to start the iteration and how to move within it*.

The solvers proposed for the loaded relaxation answer that question only implicitly, since in each case the formulation settles it rather than the designer. A Frank–Wolfe scheme takes a top-k selection of the gradient as its linear maximization oracle, and is paired with a smooth reparameterization optimized by AdamW [25]. An exact penalty built from the error-bound principle is treated by a non-convex proximal gradient method whose proximal operator is once again a top-k selection [24]. An extreme-point-pursuit formulation forces the objective to be strictly concave by a spectral shift, so that its optimum is attained at a vertex of the convex hull, and then approaches that optimum along a homotopy path deforming an easy convex problem into the concave one through a sequence of interpolated objectives [23]. In every one of these designs the iteration map is fixed as soon as the formulation is, and whatever remains at the designer's disposal, be it a step size, a continuation schedule or a starting point, acts on the trajectory from outside the map rather than on the map itself. Given that the quality of the returned subgraph is decided by the basin into which the iteration falls, controls of this kind have little influence on the outcome.

In this paper, we propose F^3-DkS (Fast, Free of eigendecomposition, Free of hyperparameters for Densest k-Subgraph extraction), a minorization–maximization treatment of the diagonally loaded relaxation in which the iteration map itself carries a free parameter. Taking inspiration from [25], we settle the loading first and build the solver on the light loading $\lambda = 1$, which gives up no tightness below the clique number, and we prove that it leaves the ranking of the candidate subgraphs untouched while keeping the graph itself in control of the search direction, so that the landscape remains traversable by a first-order method, as it does not under the classical requirement. The parameter that remains is of a different kind. It is worth mentioning that two senses of the word "free" have to be separated at the outset. A free parameter is one the minorization argument leaves undetermined, so that the designer rather than the formulation may decide it, whereas freedom from hyperparameters means that nothing in the deployed algorithm has to be tuned on the instance being solved, the stopping tolerance and the iteration budget not being counted as hyperparameters in

this context. Our surrogate keeps a separable quadratic term instead of linearizing it away, and the weight vector it carries is constrained only by a positive semidefiniteness condition with infinitely many solutions. The freedom this leaves is not a matter of step length alone. Members that are indistinguishable in theory reach stationary points of markedly different quality when launched from the same starting point, so that the weight vector acts as a selector among the basins of the relaxed landscape.

Main Contribution

The main contributions of this work are summarized as follows.

(i) **The loaded relaxation and its operating point.** We prove that the diagonal loading leaves the ranking of the candidate subgraphs untouched while taking a share of the search direction that grows with it, so that a light loading keeps the graph in control of the iteration where the classical requirement of convexity does not (Section 3.1).

(ii) **A family of separable minorizers with free weights.** We construct separable minorizers of the loaded objective on the capped simplex, indexed by a weight vector subject only to a positive semidefiniteness condition, and prove the admissible set convex and upward closed. It is far richer than the isotropic spectral shift ordinarily employed, and contains local members read off the degrees of a vertex and its neighbours at the cost of one sparse matrix–vector product (Section 3.2 and Appendix B).

(iii) **A closed-form inner problem.** Maximizing any member over the capped simplex is a strongly concave separable program whose unique solution is a clipped affine function of one scalar multiplier, obtained exactly in finitely many linear-time passes, so that the cost of an iteration grows only linearly with the number of vertices and edges (Section 3.3).

(iv) **A logarithmic minorizer and a multiplicative update.** From the arithmetic–geometric mean inequality we derive a second minorizer, separable in the logarithm, whose inner problem is a water-filling step coinciding, when no capacity constraint is active, with the replicator dynamics of the Motzkin–Straus theorem (Section 3.4).

(v) **Convergence guarantees.** Every member increases the objective by at least a weighted squared step length, the objective sequence is monotone and convergent, successive iterates draw together, and every limit point is a Karush–Kuhn–Tucker point of the relaxation (Appendix C).

(vi) **The minorization weights select the limit point.** Members launched from the same point with the same budget reach stationary points of very different quality, so the weight vector acts as a basin selector rather than a preconditioner. The isotropic shift, the classical choice and the only member needing an eigenvalue, is frequently among the worst (Section 3.5).

(vii) **Longer steps than the bound permits, made safe.** The minorization bound constrains the speed of the iteration rather than the quality of its limit, and shrinking the weights by a fixed constant reaches the same solution in a fraction of the iterations. The objective value, computed by the next iteration in any case, detects the rare overshooting steps, so that feasibility and monotonicity hold for *every* value of the constant (Section 4).

(viii) **State-of-the-art performance at linear cost.** On real-world networks spanning several orders of magnitude in size, and against the state-of-the-art and benchmark methods of this literature, F^3-DkS outperforms every solver it is compared against while requiring no tuning, no eigenvalue computation and no learning (Section 5).

Notation For the notation used throughout this paper, please refer to Appendix A.

2 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, unweighted and simple graph on $n = |\mathcal{V}|$ vertices with $m = |\mathcal{E}|$ edges, and let $\mathbf{A} \in \{0,1\}^{n \times n}$ be its symmetric adjacency matrix, whose diagonal vanishes because $\mathcal{G}$ carries no self-loops. We write $\deg_i$ for the degree of vertex i and $\mathcal{N}(i)$ for its neighbourhood, so that $\mathbf{A}\mathbf{1}$ is the vector of degrees. The adjacency matrix is held in sparse form and is never formed densely, and every operation performed on it in this paper is a sparse matrix–vector product.

A vertex subset $\mathcal{S} \subseteq \mathcal{V}$ is represented by its indicator vector $\mathbf{x} \in \{0,1\}^n$, in which $x_i = 1$ exactly when $i \in \mathcal{S}$. Each edge with both ends in $\mathcal{S}$ is counted twice by the quadratic form $\mathbf{x}^\top \mathbf{A}\mathbf{x}$, once as A_{ij} and once as A_{ji}, so that $\mathcal{S}$ induces $\frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}$ edges, while its cardinality is $\mathbf{1}^\top \mathbf{x}$. Maximizing the induced edge count over the subsets of a prescribed size k therefore reads

$$\max_{\mathbf{x}\in\mathbb{R}^n} \ \mathbf{x}^\top \mathbf{A}\mathbf{x} \qquad \text{s.t.} \qquad \mathbf{x} \in \mathcal{B}_k^n := \Big\{\mathbf{x} \in \{0,1\}^n : \textstyle\sum_i x_i = k\Big\}. \tag{1}$$

The factor $\frac{1}{2}$ has been dropped, since it scales the objective without moving its maximizers.

The quality of a returned set is measured by its *density*

$$\operatorname{dens}(\mathcal{S}) \ = \ \frac{\mathbf{x}^\top \mathbf{A}\mathbf{x}}{k(k-1)} \ = \ \frac{\text{edges induced by } \mathcal{S}}{\binom{k}{2}}, \tag{2}$$

the fraction of the $\binom{k}{2}$ possible edges that are present, so that the density is one exactly when $\mathcal{S}$ is a clique and zero when $\mathcal{S}$ induces no edge at all. Because k is fixed, (2) rescales the objective of (1) without changing what maximizes it, and it is the scale on which every result in this paper is reported, being comparable across graphs and subgraph sizes alike.

Problem (1) is the object of study for the remainder of the paper, and its feasible set, which has $\binom{n}{k}$ members, is what the continuous relaxation of the next section replaces.

3 The Proposed Method

3.1 The Loaded Relaxation

Since $x_i^2 = x_i$ whenever $x_i \in \{0,1\}$, adding $\lambda \mathbf{x}^\top \mathbf{x}$ to the objective of (1) merely adds the constant λk on $\mathcal{B}_k^n$ and leaves the set of solutions unchanged. Performing this modification first and relaxing $\mathcal{B}_k^n$ to its convex hull afterwards yields the loaded relaxation on which our solver is built, in which no tightness is lost,

$$\max_{\mathbf{x}\in\mathbb{R}^n} \ f(\mathbf{x}) := \mathbf{x}^\top (\mathbf{A} + \lambda \mathbf{I})\mathbf{x} \qquad \text{s.t.} \qquad \mathbf{x} \in \mathcal{C}_k^n := \Big\{\mathbf{x} \in [0,1]^n : \textstyle\sum_i x_i = k\Big\}, \tag{3}$$

and we abbreviate $\mathbf{M} := \mathbf{A} + \lambda \mathbf{I}$ throughout.

The loading level is the first decision the solver has to make. We work with the light loading $\lambda = 1$, which is enough for the relaxation to be tight as long as k does not exceed the clique number of $\mathcal{G}$, and the following proposition shows what that choice buys and what a heavier one would cost.

Proposition 1 (The price of the loading)**.** *Let $\lambda > 0$ and let $f(\mathbf{x}) = \mathbf{x}^\top (\mathbf{A} + \lambda \mathbf{I})\mathbf{x}$ on $\mathcal{C}_k^n$.*

(i) On the binary feasible set $\mathcal{B}_k^n$ the loading contributes the constant λk, so it leaves the ranking of the candidate subgraphs of (1) *untouched.*

(ii) At the uniform point $\mathbf{x}^0 = (k/n)\mathbf{1}$ the gradient splits into a graph term and an isotropic term, $\nabla f(\mathbf{x}^0) = 2\mathbf{A}\mathbf{x}^0 + 2\lambda \mathbf{x}^0$, whose norms satisfy

$$\frac{\|\mathbf{A}\mathbf{x}^0\|_2}{\|\lambda \mathbf{x}^0\|_2} = \frac{\bar{d}}{\lambda}, \qquad \textit{where} \qquad \bar{d} := \Big(\tfrac{1}{n}\textstyle\sum_i \deg_i^2\Big)^{1/2}. \tag{4}$$

The graph term therefore governs the search direction exactly when $\lambda < \bar{d}$, and $\lambda = 1$ meets this condition on every graph whose average degree exceeds one (for the proof, see Appendix D.1).

Part (i) says that the loading costs nothing on the set that matters, since it shifts every feasible point by the same amount, and part (ii) says where it is paid for, since the isotropic term carries no information about the graph and claims a share of the search direction that grows with λ. The loading should therefore be as small as tightness permits. At $\lambda = 1$ the ratio in (4) equals $\bar{d}$, which exceeds one on every graph in our study, whereas at the classical $\lambda \geq -\lambda_{\min}(\mathbf{A})$ it falls below one on all but the smallest and to between a fifth and two fifths on the three largest, so that a method following the gradient is steered by a term blind to the graph. We fix $\lambda = 1$ throughout, and Appendix G reports what happens when it is varied.

The geometry of $\mathcal{C}_k^n$ has shaped the solvers built on it. The Euclidean projection onto it cannot be written as a single algebraic expression, which is the reason usually given for preferring designs that avoid it altogether, an assumption Section 3.3 reconsiders.

3.2 A Separable Quadratic Minorizer and its Admissible Weights

We treat (3) with the minorization–maximization (MM) principle [21, 16, 31, 32, 34, 19]. Given the current iterate $\mathbf{x}^t$, one constructs a surrogate $g(\mathbf{x} \mid \mathbf{x}^t)$ that *minorizes* f at $\mathbf{x}^t$, that is,

$$g(\mathbf{x} \mid \mathbf{x}^t) \le f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{C}_k^n, \qquad \text{and} \qquad g(\mathbf{x}^t \mid \mathbf{x}^t) = f(\mathbf{x}^t), \tag{5}$$

and maximizes the surrogate in place of the objective, $\mathbf{x}^{t+1} \in \arg\max_{\mathbf{x} \in \mathcal{C}_k^n} g(\mathbf{x} \mid \mathbf{x}^t)$. These two conditions alone force the objective never to decrease, with no line search and no step-size selection, so the entire design problem is to find a surrogate that is at once a valid minorizer of f and cheap to maximize over $\mathcal{C}_k^n$. Both requirements are addressed in this section, the first by the construction below and the second by Section 3.3.

The obstruction to a simple surrogate is that $\mathbf{M}$ is indefinite, so that f is neither convex nor concave and no single tangent bounds it. The standard remedy is to split f into a convex part and a separable remainder, and to bound only the former. Let $\mathbf{d} \in \mathbb{R}^n$ be a vector of *minorization weights* and write

$$f(\mathbf{x}) = \underbrace{\mathbf{x}^\top \big(\mathbf{M} + \mathrm{Diag}(\mathbf{d})\big)\mathbf{x}}_{\text{convex if } \mathbf{M}+\mathrm{Diag}(\mathbf{d}) \succeq \mathbf{0}} \; - \; \underbrace{\sum_{i=1}^{n} d_i x_i^2}_{\text{separable}}. \tag{6}$$

If the first term of (6) is convex, then it lies above its tangent plane at $\mathbf{x}^t$, and substituting that linear lower bound for it leaves a function separable in the coordinates of $\mathbf{x}$. This is the whole construction.

Proposition 2 (Separable quadratic minorizer)**.** *Let $\mathbf{d} \in \mathbb{R}^n$ satisfy $\mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) \succeq \mathbf{0}$ and set $\mathbf{b}^t := (\mathbf{M} + \mathrm{Diag}(\mathbf{d}))\mathbf{x}^t$. Then*

$$g_{\mathrm{q}}(\mathbf{x} \mid \mathbf{x}^t) \;:=\; 2\,(\mathbf{b}^t)^\top \mathbf{x} - \sum_{i=1}^{n} d_i x_i^2 \;-\; (\mathbf{x}^t)^\top (\mathbf{M} + \mathrm{Diag}(\mathbf{d}))\mathbf{x}^t \tag{7}$$

satisfies the minorization conditions (5)*, and the gap between the objective and the surrogate is given by the quadratic form*

$$f(\mathbf{x}) - g_{\mathrm{q}}(\mathbf{x} \mid \mathbf{x}^t) = (\mathbf{x} - \mathbf{x}^t)^\top \big(\mathbf{M} + \mathrm{Diag}(\mathbf{d})\big)(\mathbf{x} - \mathbf{x}^t), \tag{8}$$

so the surrogate is tight exactly along the directions lying in the null space of $\mathbf{M} + \mathrm{Diag}(\mathbf{d})$ (for the proof, see Appendix D.2).

Two features of Proposition 2 deserve emphasis. The first is that the surrogate is separable, so the inner problem couples the coordinates of $\mathbf{x}$ only through the two constraints defining $\mathcal{C}_k^n$; this is what makes it solvable in closed form in Section 3.3. The second, and the one this paper is built on, is that the hypothesis $\mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) \succeq \mathbf{0}$ does not determine $\mathbf{d}$. Writing

$$\mathcal{D} := \big\{ \mathbf{d} \in \mathbb{R}^n : \; \mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) \succeq \mathbf{0} \big\} \tag{9}$$

for the set of admissible weights, every $\mathbf{d} \in \mathcal{D}$ generates a valid iteration, and (8) shows how the choice acts, since the larger the weights the larger the gap between f and its surrogate away from $\mathbf{x}^t$, and hence the more conservative the step. The set $\mathcal{D}$ is convex, being the preimage of the positive semidefinite cone under an affine map, and upward closed, since adding a nonnegative diagonal preserves positive semidefiniteness, both of which Section 4 uses. Three members are worth recording.

Proposition 3 (Admissible weights)**.** *Each of the following choices of $\mathbf{d}$ belongs to $\mathcal{D}$.*

- *(i)* Spectral (isotropic). $d_i = \max\{-\lambda_{\min}(\mathbf{A}) - \lambda,\, 0\}$ *for all i.*
- *(ii)* Signless Laplacian. $d_i = \deg_i - \lambda$. *Indeed $\mathbf{A} + \mathrm{Diag}(\deg) = \mathbf{Q}$ is the signless Laplacian of $\mathcal{G}$, and $\mathbf{x}^\top \mathbf{Q}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} (x_i + x_j)^2 \ge 0$ for every $\mathbf{x} \in \mathbb{R}^n$.*
- *(iii)* Degree family. *For any $\beta \in \mathbb{R}$, $d_i = s_i(\beta) - \lambda$ with $s_i(\beta) = \deg_i^{-\beta/2} \sum_{j \in \mathcal{N}(i)} \deg_j^{\beta/2}$. Choice (ii) is recovered by setting the exponent to zero (for the proof, see Appendix D.3).*

The distinction between member (i) and members (ii)–(iii) is not cosmetic. The isotropic shift is governed by the single worst direction of the spectrum of $\mathbf{A}$ and imposed on all n coordinates alike, whereas the local members assign a small weight wherever the graph is locally sparse and a large one only where it is locally dense, at the cost of one sparse matrix–vector product and no eigenvalue computation (Section 3.5).

3.3 The Inner Problem Admits a Closed-Form Solution

Discarding the terms independent of $\mathbf{x}$, maximizing (7) over the set $\mathcal{C}_k^n$ amounts to

$$\max_{\mathbf{x}} \; 2\,\mathbf{b}^\top \mathbf{x} - \sum_{i=1}^{n} d_i x_i^2 \qquad \text{s.t.} \qquad \mathbf{0} \le \mathbf{x} \le \mathbf{1},\; \mathbf{1}^\top \mathbf{x} = k, \tag{10}$$

a separable program, strongly concave whenever $\mathbf{d} > \mathbf{0}$, over a nonempty compact polytope. Its solution follows from the Karush–Kuhn–Tucker conditions and is described entirely by one scalar.

Proposition 4 (Closed form of the inner problem)**.** *Let* $\mathbf{d} > \mathbf{0}$*. Problem* (10) *has the unique solution*

$$x_i(\theta) = \Big[\tfrac{b_i}{d_i} - \tfrac{\theta}{2d_i}\Big]_0^1, \qquad i = 1, \dots, n, \tag{11}$$

where $\theta \in \mathbb{R}$ *is the multiplier of the equality constraint and is the unique root of* $\phi(\theta) := \sum_i x_i(\theta) - k$*. The function* ϕ *is continuous, non-increasing and piecewise linear with at most* $2n$ *breakpoints, so the root is obtained exactly by sorting the breakpoints in* $O(n \log n)$ *operations (for the proof, see Appendix D.4).*

Proposition 4 says that the whole inner problem reduces to a one-dimensional monotone root find, and it also puts the design of the existing projection-free solvers in perspective. Setting $\mathbf{d} = \mathbf{1}$ in (11) recovers the Euclidean projection onto $\mathcal{C}_k^n$, for which an exact $O(n \log n)$ algorithm has long been known [36]. That projection does not lack a closed form, it is merely not expressible as a single algebraic formula, because θ is defined implicitly. In practice we never sort. Since ϕ is monotone and piecewise linear, a Newton iteration bracketed by bisection reaches the exact root in finitely many $O(n)$ passes, and warm-starting θ keeps their number small and essentially independent of n (Remark 12). One MM iteration therefore costs one sparse matrix–vector product plus a handful of vector operations, that is $O(m + n)$, the same order as the Frank–Wolfe iteration built on the same relaxation.

Combining Propositions 2 and 4 gives the basic iteration of this paper, which we call SQM,

$$\boxed{\mathbf{x}^{t+1} = \arg\max_{\mathbf{x} \in \mathcal{C}_k^n} \Big\{ 2\big[(\mathbf{A} + \lambda \mathbf{I} + \mathrm{Diag}(\mathbf{d}))\mathbf{x}^t\big]^\top \mathbf{x} - \sum_i d_i x_i^2 \Big\}} \tag{12}$$

evaluated in closed form through (11).

3.4 A Logarithmic Minorizer and a Multiplicative Update

The surrogate of Section 3.2 exploits the spectrum of $\mathbf{A}$, whereas a second one can be obtained from its sign structure and leads to an iteration of a different flavour. Applying the elementary inequality $u \ge 1 + \log u$ to each bilinear term of $\mathbf{x}^\top \mathbf{M} \mathbf{x}$ with $u = x_i x_j / (x_i^t x_j^t)$, and using the nonnegativity of $\mathbf{M}$, which is what lets the inequality be multiplied by M_{ij} without being reversed, produces a minorizer separable in $\log \mathbf{x}$.

Proposition 5 (Logarithmic minorizer)**.** *Let* $\mathbf{x}^t > \mathbf{0}$ *and set* $w_i := x_i^t (\mathbf{M}\mathbf{x}^t)_i \ge 0$*. Then*

$$g_\ell(\mathbf{x} \mid \mathbf{x}^t) := f(\mathbf{x}^t) + 2 \sum_{i=1}^{n} w_i \big(\log x_i - \log x_i^t \big) \tag{13}$$

satisfies the minorization conditions (5) *on the positive orthant, and its maximizer over* $\mathcal{C}_k^n$ *is the water-filling step* $x_i = \min\{1, w_i/\theta\}$*, where* $\theta > 0$ *is the unique root of* $\sum_i \min\{1, w_i/\theta\} = k$ *(for the proof, see Appendix D.5).*

We call the resulting iteration SLM. Its update is multiplicative, $x_i^{t+1} \propto x_i^t (\mathbf{M}\mathbf{x}^t)_i$ truncated at one. When no capacity constraint is active the truncation is inactive as well and the update becomes exactly $x_i^{t+1} = k\, x_i^t (\mathbf{M}\mathbf{x}^t)_i / (\mathbf{x}^t)^\top \mathbf{M} \mathbf{x}^t$. For $k = 1$ this is the replicator dynamics of evolutionary game theory, whose fixed points characterize the maximal cliques of $\mathcal{G}$ through the Motzkin–Straus theorem [27, 4, 7], a coincidence we record without making further use of it. The multiplicative form also exposes a practical weakness that explains the behaviour of SLM in the experiments, since a coordinate reaching zero can never recover, so the iteration sparsifies aggressively and its outcome depends strongly on where it is started.

Algorithm 1 F^3-DkS for the diagonally loaded DkS relaxation

Input: adjacency matrix $\mathbf{A}$, size k, loading λ, shrinkage c, budget T.
Form the degree-adaptive weights $d_i = s_i(\beta) - \lambda$ of Proposition 3(iii) and set $\mathbf{x}^0 = (k/n)\mathbf{1}$.
for $t = 0, 1, \ldots, T-1$ **do**
  $\mathbf{b}^t \leftarrow \mathbf{A}\mathbf{x}^t + (c\,\mathbf{d} + \lambda\mathbf{1}) \odot \mathbf{x}^t$
  $\mathbf{x}^{t+1} \leftarrow$ solution of (10) with weights $c\,\mathbf{d}$, obtained from (11)
  if $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$ **then**
    recompute $\mathbf{x}^{t+1}$ from (11) using the unscaled weights $\mathbf{d}$ ▷ safeguard
  end if
end for
Output: the k largest entries of $\mathbf{x}^T$.

3.5 The Minorization Weights Select the Limit Point

Every $\mathbf{d} \in \mathcal{D}$ produces an algorithm with identical guarantees, since Appendix C establishes monotone ascent, square-summable steps and convergence to a Karush–Kuhn–Tucker point of (3) for every admissible weight vector alike, at a cost of $O(m+n)$ per iteration. The theory therefore leaves the members interchangeable, whereas practice separates them sharply. Launched from the same starting point with the same iteration budget, members differing only in the exponent β of Proposition 3(iii) terminate at stationary points of markedly different quality, by a margin large enough to dominate the difference between our method and the published baselines. Neither the shared initialization nor a failure to converge can account for it, so the weight vector acts as a selector among the basins rather than as a preconditioner. Rescaling the coordinates changes the relative speed at which vertices may grow, and those permitted to grow early recruit their neighbours, so that the subgraph nucleates around them.

Because the members that do well depend on the global arrangement of dense pockets in a particular graph, we know of no way to characterize the best one analytically, and none is best everywhere. What can be asked of the family is a robust default, and the intermediate members supply one. We fix $\beta = \frac{1}{2}$ throughout Section 5, not because it dominates but because it is the member one would choose knowing nothing about the instance at hand, and Appendix H reports the study behind these statements.

4 Taking Longer Steps Than the Minorization Bound Permits

The analysis of Appendix C guarantees ascent for every $\mathbf{d} \in \mathcal{D}$, and Proposition 2 enlarges $\mathcal{D}$ as far as it can be while the surrogate remains a minorizer of f. That guarantee is nevertheless a worst-case one, holding simultaneously for every graph, every k and every iterate along the trajectory, so it says nothing about how conservative the bound is on the instances actually encountered. We show here that it is conservative by a wide margin.

Shrinking the weights. Take any admissible $\mathbf{d} \in \mathcal{D}$ and replace it by $c\,\mathbf{d}$ with $c > 0$ in (12). For $c \geq 1$ the weights only grow, and since $\mathcal{D}$ is upward closed the step merely becomes shorter and the guarantee is retained, whereas for $c < 1$ the surrogate may fail to minorize f and the iteration may overshoot. The constant interpolates between two familiar extremes, since as $c \to 0$ the quadratic term of (10) vanishes and the inner problem degenerates into the top-k oracle that drives Frank–Wolfe, whereas $c = 1$ recovers the conservative step.

The bound supplies its own test. The guarantee is not simply abandoned when $c < 1$. A step that fails to ascend is detected exactly by $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$, and evaluating $f(\mathbf{x}^{t+1})$ needs only the product $\mathbf{A}\mathbf{x}^{t+1}$, which the following iteration computes in any case, so the test is free unless it fires. When it does, the step is recomputed with the unscaled $\mathbf{d}$, which lies in $\mathcal{D}$ and therefore ascends by Lemma 8. We call the resulting iteration F^3-DkS and summarize it in Algorithm 1.

Theorem 6 (Soundness at any shrinkage)**.** *Let $\mathbf{d} \in \mathcal{D}$ with $\mathbf{d} > \mathbf{0}$ and let $c > 0$ be arbitrary. The iterates of Algorithm 1 satisfy $\mathbf{x}^t \in \mathcal{C}_k^n$ for all t, and with the safeguard enabled $f(\mathbf{x}^{t+1}) \geq f(\mathbf{x}^t)$ for all t, irrespective of the value of c (for the proof, see Appendix D.6).*

Feasibility holds because the maximizer in (10) returns a point of $\mathcal{C}_k^n$ whatever weights it is given, and monotonicity because every step either ascends already or is replaced by one that does. The minorization inequality therefore does double duty, as the guarantee that makes $c = 1$ safe *a priori* and as a run-time detector that makes any other c safe *a posteriori*, at the cost of one extra sparse matrix–vector product on the steps where it fires and nothing on those where it does not.

Fixing the constant. One number remains to be chosen. We fix it once and for all on synthetic Chung–Lu graphs with planted dense subgraphs [10] that share no vertex with any evaluation graph, and apply it unchanged to every real network, so that no instance-level tuning takes place anywhere. The choice is not delicate, and we take $c = 0.06$ from the middle of a broad flat interval. What the shrinkage buys is a reduction in the number of iterations rather than a change in the fixed point reached. Appendices J, K and L report the grid on which the constant is fixed, the effect of the iteration budget, and the behaviour of the safeguard.

5 Experiments

Our experiments substantiate the claim that drives the paper, namely that it is the choice of minorization weights, and not the iteration budget, that decides the quality of the solution returned, and they place the resulting algorithm against the published state of the art.

Setup. We evaluate on seven undirected SNAP networks [22], the graphs on which this literature is standardly benchmarked, spanning three orders of magnitude in size, from a collaboration graph with roughly ten thousand vertices to soc-pokec with 1.6 million vertices and 22.3 million edges. The subgraph size ranges over $k \in \{10, 20, 30, 50, 70, 100, 200, 300, 500, 700, 1000\}$, giving 77 instances, from sizes at which tight cliques exist to sizes at which none does. We report the density (2) of the returned set, the measure on which the competing solvers are themselves assessed, so that every method is judged by the criterion its own authors use. Every continuous method is rounded identically, by keeping the k largest entries of its score vector, so that the comparison isolates the optimizer rather than the rounding rule, and we also report the spectral upper bound available for this problem [28].

We compare F^3-DkS against six published solvers, namely Greedy [11], TPM [37], Rank-1 [28], FW and AdamW [25], and EP-Prox [24], each implemented from its own paper and run with the settings specified there, so that every competitor is met in the configuration its authors intended. F^3-DkS uses the $\beta = \frac{1}{2}$ member of the degree family, is run from the uniform starting point for 40 iterations, and is never tuned on any evaluation graph. Appendix F gives the protocol.

The starting point. We use the uniform point $\mathbf{x}^0 = (k/n)\mathbf{1}$ throughout, since it costs nothing to compute and is the best of the natural candidates where the choice matters at all. A nearly binary point sits close to a vertex of $\mathcal{C}_k^n$ and leaves the iteration almost no room in which to move, so the better it looks as a subgraph the less it can be improved upon. Appendix I varies the starting point over six candidates with the minorizer held fixed.

5.1 Comparison Against Published Solvers

Table 1 reports the comparison and Figure 1 its dependence on k. Over the 77 instances F^3-DkS attains a mean density of 57.3% and outperforms every published solver, the nearest of them being

Table 1: Mean density (%) over all k, by graph. The last column aggregates every instance. Best per column in bold, second best underlined.

Method	facebook	web-Stanford	as-skitter	soc-pokec	ca-HepTh	ca-CondMat	com-dblp	*all graphs*
Greedy (2001) [11]	38.5	23.2	33.6	17.7	11.7	19.3	25.5	24.2
TPM (2013) [37]	67.3	25.4	6.8	2.8	20.1	33.0	56.2	30.2
Rank-1 (2014) [28]	69.0	29.3	22.0	17.3	36.0	29.4	61.6	37.8
FW (2025) [25]	**70.1**	<u>59.6</u>	**72.3**	33.4	20.0	34.6	63.3	50.5
AdamW (2025) [25]	<u>69.9</u>	48.5	68.9	<u>40.1</u>	**39.2**	**38.2**	<u>64.7</u>	<u>52.8</u>
EP-Prox (2026) [24]	67.6	15.7	13.8	5.6	29.6	25.3	30.7	26.9
F^3-DkS *(ours)*	**70.1**	**65.7**	<u>72.2</u>	**53.3**	<u>38.4</u>	<u>36.6</u>	**64.8**	**57.3**

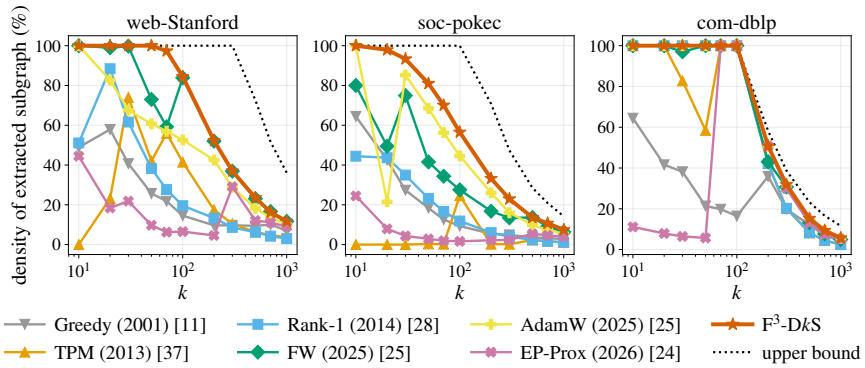

Figure 1: Density against subgraph size k on the three graphs where the methods separate most.

AdamW at 52.8% and Frank–Wolfe at 50.5%, and it is best or tied best on more instances than any other method, in 40 iterations and with no tuning, no eigenvalue computation and no learning. The margins are largest at small k. On web-Stanford at $k = 50$ it returns a clique, where Frank–Wolfe reaches 73.0%, AdamW 60.7% and EP-Prox 9.7%, and Figure 1 shows the advantage sustained out to $k = 70$.

The aggregate is not carried by a single instance. F^3-DkS leads or ties for the lead on most of the graphs and is on a par with the leading method on the rest, matching Frank–Wolfe on as-skitter to within four hundredths of a point (72.22% against 72.26%) in marginally less time, and coming within 0.8 and 1.6 points of AdamW on the two smallest graphs while running some three hundred to five hundred times faster. F^3-DkS gains as much as 13.2 points of density over the strongest competing solver on a graph, and nowhere in the study does any of them finish more than 1.6 points above it. Appendices M and N give the timings and the per-instance results.

6 Limitations

The relaxation is provably tight only while k stays below the clique number; above that range no proof is available, although the iteration behaved well there on every graph we tried.

7 Conclusion

In this paper, we studied the diagonally loaded relaxation of DkS from the standpoint of optimization rather than modelling, and treated it with the minorization–maximization principle. We fixed the loading at the lightest value that keeps the relaxation tight, which is what leaves the graph itself in control of the search direction, and built on it separable minorizers indexed by a weight vector constrained only by positive semidefiniteness, every member of which solves its inner problem in closed form, costs $O(m + n)$ per iteration and ascends provably to a Karush–Kuhn–Tucker point. What we regard as most consequential is that these identical guarantees conceal very different behaviour, the weights selecting the basin and not merely the step length. F^3-DkS exploits this through a local, degree-adaptive member that needs no tuning and no eigenvalue computation, and that takes steps longer than the bound permits, made safe by the test the same bound supplies. On seven real-world networks it outperformed every published solver it was compared against.

AI use statement

AI-assisted editing was used for language editing and proofreading of the manuscript, and for reviewing code written by the authors. Generative AI tools were not used for research ideation, for the minorizers or any other theoretical result, for the proofs of Appendix D, for the algorithms, or for the conduct and reporting of the experiments. We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

Reproducibility statement

We have made every effort to make this work reproducible, and we release the source code of our method together with the data pipeline as a supplementary zip archive submitted with this paper. The archive contains a reference implementation of the minorizers of Section 3, of the closed-form inner solve of Proposition 4 and of Algorithm 1; a download script that fetches each of the seven evaluation graphs from its canonical SNAP source and verifies it against the checksum of the file we used, together with the preprocessing code that reproduces the sizes in Table 3; the generator of the synthetic graphs of Appendix F.4 on which the shrinkage constant is fixed; the result files behind every number we report; and the scripts that regenerate every table and figure of the paper from those files. Correctness tests for the closed-form inner solves and for the monotone ascent of each MM variant are included as well. The algorithms themselves are stated in the paper, with the surrogates and their admissible weights in Section 3, the deployed iteration in Algorithm 1, and the complete experimental protocol – preprocessing, rounding rule, iteration budgets and the time cap on the two largest graphs – in Appendix F. All theoretical claims are stated with their hypotheses and proved in Appendix D, with the convergence analysis collected in Appendix C. Every method is deterministic given the graph and k, the single constant c is fixed once on synthetic graphs that share no vertex with any evaluation graph (Appendix J) and is never tuned on an evaluation graph, and the per-instance results underlying every aggregate are reported in Appendix N. The measured densities behind every row of every table are included in the archive, so that each table and figure of the paper regenerates exactly.

References

[1] Brendan P. Ames. Guaranteed recovery of planted cliques and dense subgraphs by convex relaxation. *Journal of Optimization Theory and Applications*, 167:653–675, 2015.

[2] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB Journal*, 23:175–199, 2014.

[3] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and MapReduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012.

[4] Leonard E. Baum and John A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967.

[5] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. CopyCatch: Stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd International Conference on World Wide Web*, pp. 119–130, 2013.

[6] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pp. 201–210, 2010.

[7] Immanuel M. Bomze. Evolution towards the maximum clique. *Journal of Global Optimization*, 10:143–164, 1997.

[8] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp. 84–95, 2000.

[9] Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering*, 24(7):1216–1230, 2012.

[10] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6(2):125–145, 2002.

[11] Uriel Feige, David Peleg, and Guy Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29 (3):410–421, 2001.

[12] Eugene Fratkin, Brian T. Naughton, Douglas L. Brutlag, and Serafim Batzoglou. MotifCut: Regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14):e150–e157, 2006.

[13] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pp. 721–732, 2005.

[14] Andrew V. Goldberg. Finding a maximum density subgraph. Technical Report UCB/CSD-84-171, University of California, Berkeley, 1984.

[15] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. FRAUDAR: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 895–904, 2016.

[16] David R. Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[17] Subhash Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[18] Aritra Konar and Nicholas D. Sidiropoulos. Exploring the subgraph density-size trade-off via the Lovász extension. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 743–751, 2021.

[19] Sandeep Kumar, Jiaxi Ying, José Vinícius de M. Cardoso, and Daniel P. Palomar. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

[20] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *ACM Computing Surveys*, 56(8):1–40, 2024.

[21] Kenneth Lange, David R. Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

[22] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[23] Ya Liu, Junbin Liu, and Wing-Kin Ma. Cardinality-constrained binary quadratic optimization via extreme point pursuit, with application to the densest k-subgraph problem. In *ICASSP 2024 – IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 9631–9635, 2024.

[24] Ya Liu, Junbin Liu, Wing-Kin Ma, and Aritra Konar. A scalable and exact relaxation for densest k-subgraph via error bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 15360–15368, 2026.

[25] Qiheng Lu, Nicholas D. Sidiropoulos, and Aritra Konar. Densest k-subgraph mining via a provably tight relaxation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 12291–12299, 2025.

[26] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–961, 2017.

[27] Theodore S. Motzkin and Ernst G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

[28] Dimitris Papailiopoulos, Ioannis Mitliagkas, Alexandros Dimakis, and Constantine Caramanis. Finding dense subgraphs via low-rank bilinear optimization. *Proceedings of the 31st International Conference on Machine Learning*, pp. 1890–1898, 2014.

[29] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[30] Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology (RECOMB)*, pp. 456–472, 2010.

[31] Junxiao Song, Prabhu Babu, and Daniel P. Palomar. Optimization methods for designing sequences with low autocorrelation sidelobes. *IEEE Transactions on Signal Processing*, 63(15): 3998–4009, 2015.

[32] Junxiao Song, Prabhu Babu, and Daniel P. Palomar. Sparse generalized eigenvalue problem via smooth optimization. *IEEE Transactions on Signal Processing*, 63(7):1627–1642, 2015.

[33] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 939–948, 2010.

[34] Ying Sun, Prabhu Babu, and Daniel P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017.

[35] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 104–112, 2013.

[36] Weiran Wang and Canyi Lu. Projection onto the capped simplex. *arXiv preprint arXiv:1503.01002*, 2015.

[37] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14:899–925, 2013.

Supplementary Material

A Notation

Table 2 collects the symbols used throughout the paper.

Table 2: Notation used throughout the paper.

Symbol	Meaning
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	undirected, unweighted simple graph
$n,\ m$	number of vertices and of edges of $\mathcal{G}$
k	prescribed size of the subgraph sought
ω	clique number of $\mathcal{G}$
$\deg_i,\ \mathcal{N}(i)$	degree and neighbourhood of vertex i
$\mathbf{A}$	adjacency matrix of $\mathcal{G}$, symmetric with zero diagonal
$\lambda,\ \mathbf{M}$	diagonal loading and the loaded matrix $\mathbf{M} = \mathbf{A} + \lambda\mathbf{I}$
$f(\mathbf{x})$	loaded objective $\mathbf{x}^\top\mathbf{M}\mathbf{x}$ of (3)
$\mathcal{B}_k^n$	binary feasible set $\{\mathbf{x} \in \{0,1\}^n : \sum_i x_i = k\}$
$\mathcal{C}_k^n$	capped simplex $\{\mathbf{x} \in [0,1]^n : \sum_i x_i = k\}$, the convex hull of $\mathcal{B}_k^n$
$\mathbf{x}^t,\ x_i$	iterate at iteration t; i-th entry of $\mathbf{x}$
$\mathbf{d},\ \mathcal{D}$	minorization weights and the admissible set (9)
$g(\mathbf{x} \mid \mathbf{x}^t)$	surrogate minorizing f at $\mathbf{x}^t$
c	shrinkage constant applied to $\mathbf{d}$
θ	multiplier of the equality constraint in the inner problem
$\mathbf{1},\ \mathbf{I},\ \mathbf{0}$	all-ones vector, identity matrix and all-zero vector or matrix
$(\cdot)^\top$	transpose
$\mathbf{A} \succeq \mathbf{0}$	$\mathbf{A}$ is positive semidefinite
$\mathbf{A} \geq \mathbf{0}$	the entries of $\mathbf{A}$ are nonnegative
$\lambda_{\min}(\mathbf{A}),\ \lambda_{\max}(\mathbf{A})$	extreme eigenvalues of a symmetric $\mathbf{A}$
$\mathrm{Diag}(\mathbf{d})$	diagonal matrix carrying the entries of $\mathbf{d}$
$\odot,\ \|\cdot\|$	Hadamard product and Euclidean norm
$[a]_0^1$	projection $\min\{1, \max\{0, a\}\}$ of a scalar onto the unit interval

B An Overview of the Minorization–Maximization Framework

The MM principle replaces a difficult optimization problem by a sequence of simpler ones, and has become a standard workhorse in signal processing, statistics and machine learning [21, 16, 31, 32, 34, 19]. Because our problem is a maximization, the surrogate is built below the objective rather than above it. Given the current iterate $\mathbf{x}^t$, one constructs a surrogate $g(\mathbf{x} \mid \mathbf{x}^t)$ that *minorizes* the objective f of (3) at $\mathbf{x}^t$ in the sense of (5), and maximizes the surrogate in place of the objective, $\mathbf{x}^{t+1} \in \arg\max_{\mathbf{x} \in \mathcal{C}_k^n} g(\mathbf{x} \mid \mathbf{x}^t)$. The two conditions in (5) are all that is needed for the scheme to be well behaved, since together with the optimality of $\mathbf{x}^{t+1}$ they give

$$f(\mathbf{x}^{t+1}) \ \geq\ g(\mathbf{x}^{t+1} \mid \mathbf{x}^t) \ \geq\ g(\mathbf{x}^t \mid \mathbf{x}^t) \ =\ f(\mathbf{x}^t), \tag{14}$$

where the first inequality is minorization and the last equality is tangency. The objective therefore never decreases, whichever surrogate is used, and this holds without any line search or step-size selection. The entire design problem is consequently to find a surrogate that is simultaneously a valid minorizer of f and cheap to maximize over $\mathcal{C}_k^n$.

The minorizers of Section 3 retain a separable *quadratic* term carrying a diagonal weight vector $\mathbf{d}$, and how that differs from the solvers already proposed for (3) is the source of everything else in this paper. Appendix E draws out the comparison.

Remark 7 (Relation to projected gradient ascent)**.** The isotropic case is instructive. When $\mathbf{d} = \mu\mathbf{1}$, the iteration (12) reduces to projected gradient ascent with step size $1/(2\mu)$, and the MM derivation identifies $\mu = -\lambda_{\min}(\mathbf{A}) - \lambda$ as the smallest isotropic weight for which monotone ascent is guaranteed, so that the longest such step is obtained without any line search. For non-isotropic $\mathbf{d}$ the iteration becomes a diagonally preconditioned ascent step whose preconditioner is read off the degree profile of the graph rather than tuned.

C Convergence Analysis and Computational Cost

We now establish the convergence properties of the iteration (12). The argument proceeds in the order in which such arguments are usually organized. We show first that the objective sequence ascends, then that it is bounded and therefore convergent, then that the distance between successive iterates tends to zero, and finally that every limit point of the iterate sequence satisfies the first-order optimality conditions of (3). The first step is sharper than the generic MM inequality (14), because the strong concavity of the quadratic surrogate turns plain monotonicity into a quantitative statement.

Lemma 8 (Quantitative ascent)**.** *Let $\mathbf{d} \in \mathcal{D}$ with $\mathbf{d} > \mathbf{0}$, and let $\{\mathbf{x}^t\}$ be generated by* (12) *from an arbitrary $\mathbf{x}^0 \in \mathcal{C}_k^n$. Then, for every $t \geq 0$,*

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) \;\geq\; \sum_{i=1}^{n} d_i \,(x_i^{t+1} - x_i^t)^2 \;\geq\; d_{\min}\, \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \;\geq\; 0, \tag{15}$$

where $d_{\min} := \min_i d_i > 0$.

Inequality (15) is stronger than plain monotonicity in a way that matters for what follows. The objective increases by at least a $\mathbf{d}$-weighted squared step length, so the iteration can only stop making progress when the iterates themselves stop moving. The two remaining ingredients are boundedness of the objective and compactness of the feasible set, and both are immediate.

Lemma 9 (Boundedness)**.** *The feasible set $\mathcal{C}_k^n$ is a nonempty compact convex polytope, and the objective of* (3) *satisfies $0 \leq f(\mathbf{x}) \leq k^2$ for every $\mathbf{x} \in \mathcal{C}_k^n$ whenever $\lambda \in [0, 1]$.*

Putting the two lemmas together yields the convergence theorem. We state it for the sequences themselves rather than only for the objective values, since it is the behaviour of $\{\mathbf{x}^t\}$ that the algorithm ultimately returns.

Theorem 10 (Convergence of SQM)**.** *Let $\mathbf{d} \in \mathcal{D}$ with $\mathbf{d} > \mathbf{0}$, and let $\{\mathbf{x}^t\} \subset \mathcal{C}_k^n$ be generated by* (12) *from an arbitrary $\mathbf{x}^0 \in \mathcal{C}_k^n$. Then the following hold.*

- *(i)* Monotonicity and convergence of the objective. *The sequence $\{f(\mathbf{x}^t)\}$ is non-decreasing and bounded above by k^2, hence it converges to a limit $f^\star \leq k^2$.*
- *(ii)* Asymptotic regularity. *The step lengths are square summable, $\sum_{t \geq 0} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \leq \big(k^2 - f(\mathbf{x}^0)\big)/d_{\min} < \infty$; consequently $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \to 0$ and $\min_{t<T} \|\mathbf{x}^{t+1} - \mathbf{x}^t\| = O(1/\sqrt{T})$.*
- *(iii)* Existence of limit points. *The sequence $\{\mathbf{x}^t\}$ is bounded and therefore possesses at least one convergent subsequence; every limit point is feasible and attains the objective value $f^\star$.*
- *(iv)* Stationarity. *Every limit point $\bar{\mathbf{x}}$ of $\{\mathbf{x}^t\}$ is a Karush–Kuhn–Tucker point of* (3)*, that is, it satisfies $\nabla f(\bar{\mathbf{x}})^\top (\mathbf{x} - \bar{\mathbf{x}}) \leq 0$ for all $\mathbf{x} \in \mathcal{C}_k^n$.*

The same conclusions hold for SLM on any compact subset of the relative interior of $\mathcal{C}_k^n$, with (15) *replaced by* (14)*.*

It is important to be equally clear about what Theorem 10 does not assert. Nothing in it prevents the limit point from being a stationary point of poor quality, and since the objective is non-convex and its landscape rich in such points, this is not a remote possibility but the typical situation. Section 3.5 shows that it is exactly what happens in practice, and that the choice of $\mathbf{d}$ is what decides the outcome.

Regarding cost, one iteration of (12) requires one sparse matrix–vector product with $\mathbf{A}$ and a constant number of $O(n)$ passes for the root find, so it costs $O(m + n)$ with $O(n)$ memory beyond the graph itself. The weights of Proposition 3(ii) and (iii) cost one further matrix–vector product, computed once and reused for all k, whereas those of Proposition 3(i) require $\lambda_{\min}(\mathbf{A})$, the only quantity anywhere in the method whose cost is not linear in m and a further reason to prefer the local weights at scale.

D Proofs

Throughout this appendix, $\mathbf{M} = \mathbf{A} + \lambda \mathbf{I}$ with $\lambda \in [0, 1]$, the matrix $\mathbf{A}$ is the symmetric adjacency matrix of a simple graph, so that $\mathbf{A} \geq \mathbf{0}$ entrywise and $A_{ii} = 0$, and $\mathcal{C}_k^n = \{\mathbf{x} \in [0, 1]^n : \mathbf{1}^\top \mathbf{x} = k\}$ with $1 \leq k \leq n$.

D.1 Proof of Proposition 1

(i) If $\mathbf{x} \in \mathcal{B}_k^n$ then every coordinate is 0 or 1, so $x_i^2 = x_i$ and $\mathbf{x}^\top \mathbf{x} = \sum_i x_i = k$. Hence $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \lambda k$ for every $\mathbf{x} \in \mathcal{B}_k^n$, and two points of $\mathcal{B}_k^n$ differ in f exactly as they differ in $\mathbf{x}^\top \mathbf{A}\mathbf{x}$.

(ii) Differentiating gives $\nabla f(\mathbf{x}) = 2(\mathbf{A} + \lambda \mathbf{I})\mathbf{x} = 2\mathbf{A}\mathbf{x} + 2\lambda \mathbf{x}$, in which the second term is a positive multiple of $\mathbf{x}$ and does not involve $\mathbf{A}$. At $\mathbf{x}^0 = (k/n)\mathbf{1}$ the i-th entry of $\mathbf{A}\mathbf{x}^0$ is $(k/n)\deg_i$, so that $\|\mathbf{A}\mathbf{x}^0\|_2 = (k/n)\big(\sum_i \deg_i^2\big)^{1/2}$, while $\|\lambda \mathbf{x}^0\|_2 = \lambda(k/n)\sqrt{n}$. Dividing the first by the second and cancelling k/n yields (4), and the ratio exceeds one precisely when $\lambda < \bar{d}$. Finally, $\bar{d}$ is the quadratic mean of the degrees and therefore at least their arithmetic mean $2m/n$, so $\bar{d} > 1$ whenever the average degree exceeds one, and the condition is met at $\lambda = 1$. ■

D.2 Proof of Proposition 2

Write $\mathbf{D} = \mathrm{Diag}(\mathbf{d})$. Starting from the identity $\mathbf{x}^\top \mathbf{M}\mathbf{x} = \mathbf{x}^\top(\mathbf{M} + \mathbf{D})\mathbf{x} - \sum_i d_i x_i^2$ we obtain, for any pair $\mathbf{x}, \mathbf{x}^t$,

$$\begin{aligned} f(\mathbf{x}) - g_{\mathrm{q}}(\mathbf{x} \mid \mathbf{x}^t) &= \Big[\mathbf{x}^\top(\mathbf{M} + \mathbf{D})\mathbf{x} - \sum_i d_i x_i^2\Big] \\ &\quad - \Big[2(\mathbf{x}^t)^\top(\mathbf{M} + \mathbf{D})\mathbf{x} - \sum_i d_i x_i^2 - (\mathbf{x}^t)^\top(\mathbf{M} + \mathbf{D})\mathbf{x}^t\Big] \\ &= \mathbf{x}^\top(\mathbf{M} + \mathbf{D})\mathbf{x} - 2(\mathbf{x}^t)^\top(\mathbf{M} + \mathbf{D})\mathbf{x} + (\mathbf{x}^t)^\top(\mathbf{M} + \mathbf{D})\mathbf{x}^t \\ &= (\mathbf{x} - \mathbf{x}^t)^\top(\mathbf{M} + \mathbf{D})(\mathbf{x} - \mathbf{x}^t), \end{aligned} \tag{16}$$

which is (8). By hypothesis $\mathbf{M} + \mathbf{D} = \mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) \succeq \mathbf{0}$, so the right-hand side of (16) is nonnegative for every $\mathbf{x}$, which is the minorization inequality, and it vanishes at $\mathbf{x} = \mathbf{x}^t$, which is the tangency condition. This proves both claims. We record in addition that differentiating (7) gives $\nabla g_{\mathrm{q}}(\mathbf{x} \mid \mathbf{x}^t) = 2\mathbf{b}^t - 2\mathbf{D}\mathbf{x}$, so that $\nabla g_{\mathrm{q}}(\mathbf{x}^t \mid \mathbf{x}^t) = 2(\mathbf{M} + \mathbf{D})\mathbf{x}^t - 2\mathbf{D}\mathbf{x}^t = 2\mathbf{M}\mathbf{x}^t = \nabla f(\mathbf{x}^t)$. The surrogate is therefore first-order consistent with the objective at the current iterate, a fact used in the proof of Theorem 10. ■

D.3 Proof of Proposition 3

(i) If $d_i = \mu := \max\{-\lambda_{\min}(\mathbf{A}) - \lambda, 0\}$ for all i, then $\mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) = \mathbf{A} + (\mu + \lambda)\mathbf{I}$, whose smallest eigenvalue is $\lambda_{\min}(\mathbf{A}) + \mu + \lambda \geq 0$ by the definition of μ.

(iii) We prove the general degree family first, since (ii) is the special case $\beta = 0$. Let $p_i > 0$ for all i. For any indices i, j and any reals x_i, x_j,

$$0 \leq \left(\sqrt{\tfrac{p_j}{p_i}}\, x_i + \sqrt{\tfrac{p_i}{p_j}}\, x_j\right)^2 = \tfrac{p_j}{p_i} x_i^2 + \tfrac{p_i}{p_j} x_j^2 + 2x_i x_j, \tag{17}$$

that is, $2x_i x_j \geq -\big(\tfrac{p_j}{p_i} x_i^2 + \tfrac{p_i}{p_j} x_j^2\big)$. Summing this over the edge set and using the fact that $\mathbf{A}$ is the adjacency matrix of a simple graph,

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = 2\sum_{(i,j)\in\mathcal{E}} x_i x_j \;\geq\; -\sum_{(i,j)\in\mathcal{E}} \left(\tfrac{p_j}{p_i} x_i^2 + \tfrac{p_i}{p_j} x_j^2\right) \;=\; -\sum_{i=1}^n x_i^2 \frac{1}{p_i} \sum_{j\in\mathcal{N}(i)} p_j \;=\; -\sum_{i=1}^n s_i x_i^2, \tag{18}$$

where $s_i := p_i^{-1} \sum_{j\in\mathcal{N}(i)} p_j$. Hence $\mathbf{x}^\top(\mathbf{A} + \mathrm{Diag}(\mathbf{s}))\mathbf{x} \geq 0$ for every $\mathbf{x} \in \mathbb{R}^n$, that is, $\mathbf{A} + \mathrm{Diag}(\mathbf{s}) \succeq \mathbf{0}$, and the choice $d_i = s_i - \lambda$ gives $\mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda \mathbf{1}) = \mathbf{A} + \mathrm{Diag}(\mathbf{s}) \succeq \mathbf{0}$. Taking $p_i = \deg_i^{\beta/2}$ yields the stated $s_i(\beta)$. Vertices of degree zero contribute an identically zero row and column to $\mathbf{A}$ and may be assigned any $s_i \geq 0$; in the implementation we replace $\deg_i$ by $\max\{\deg_i, 1\}$ in the definition of p_i, which is immaterial for the same reason.

(ii) With $\beta = 0$ we have $p_i \equiv 1$ and therefore $s_i = \deg_i$, so that $\mathbf{A} + \mathrm{Diag}(\deg) = \mathbf{Q}$ is the signless Laplacian and the display above reduces to the familiar identity $\mathbf{x}^\top \mathbf{Q}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} (x_i + x_j)^2 \geq 0$. ■

Remark 11 (Structure of $\mathcal{D}$)**.** The set $\mathcal{D}$ defined in (9) is convex, being the preimage of the positive semidefinite cone under an affine map, and it is upward closed, since $\mathbf{d}' \geq \mathbf{d}$ implies $\mathbf{A} + \mathrm{Diag}(\mathbf{d}' + \lambda\mathbf{1}) = \mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda\mathbf{1}) + \mathrm{Diag}(\mathbf{d}' - \mathbf{d}) \succeq \mathbf{0}$. Consequently any convex combination of the weights of Proposition 3, and any pointwise increase of such a combination, is again an admissible weight vector. This is what makes the shrinkage of Section 4 analyzable, since for $c \geq 1$ the shrunk weights remain in $\mathcal{D}$, and for $c < 1$ the safeguard returns to a vector that does.

D.4 Proof of Proposition 4

Since $\mathbf{d} > \mathbf{0}$, the objective of (10) is strictly concave and the feasible set is a nonempty compact convex polytope, nonempty because $1 \leq k \leq n$, so a unique maximizer exists. The constraints are affine, hence the Karush–Kuhn–Tucker conditions are both necessary and sufficient. Introducing a multiplier $\theta \in \mathbb{R}$ for the equality $\mathbf{1}^\top \mathbf{x} = k$ and multipliers $\boldsymbol{\nu} \geq \mathbf{0}$ and $\boldsymbol{\mu} \geq \mathbf{0}$ for $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{x} \leq \mathbf{1}$, stationarity of the Lagrangian reads

$$2b_i - 2d_i x_i - \theta + \nu_i - \mu_i = 0, \qquad \nu_i x_i = 0, \qquad \mu_i(x_i - 1) = 0. \tag{19}$$

If $0 < x_i < 1$ then $\nu_i = \mu_i = 0$ and (19) gives $x_i = \frac{b_i}{d_i} - \frac{\theta}{2d_i}$. If $x_i = 0$ then $\mu_i = 0$ and $\nu_i = \theta - 2b_i \geq 0$, which is equivalent to $\frac{b_i}{d_i} - \frac{\theta}{2d_i} \leq 0$. If $x_i = 1$ then $\nu_i = 0$ and $\mu_i = 2b_i - 2d_i - \theta \geq 0$, equivalent to $\frac{b_i}{d_i} - \frac{\theta}{2d_i} \geq 1$. The three cases combine into (11).

It remains to characterize θ. Each coordinate map $\theta \mapsto x_i(\theta)$ is continuous, non-increasing because $1/(2d_i) > 0$, and piecewise linear with the two breakpoints $\theta = 2b_i - 2d_i$ and $\theta = 2b_i$, so $\phi(\theta) = \sum_i x_i(\theta) - k$ is continuous, non-increasing and piecewise linear with at most $2n$ breakpoints. Moreover $\phi(\theta) \to n - k \geq 0$ as $\theta \to -\infty$ and $\phi(\theta) \to -k < 0$ as $\theta \to +\infty$, so ϕ has at least one root. The set of roots is a closed interval on which ϕ vanishes identically; since no coordinate can change on such an interval without changing the sum, $\mathbf{x}(\theta)$ is constant there, and any root produces the same, unique, maximizer. Sorting the $2n$ breakpoints and scanning them identifies the bracketing interval in $O(n \log n)$ operations, after which θ follows from one linear equation. ■

Remark 12 (The root find in practice)**.** We do not sort. Because ϕ is monotone and piecewise linear, the Newton step $\theta \leftarrow \theta - \phi(\theta)/\phi'(\theta)$ with $\phi'(\theta) = -\sum_{i \in \mathcal{F}(\theta)} \frac{1}{2d_i}$, where $\mathcal{F}(\theta) = \{i : 0 < x_i(\theta) < 1\}$, either lands on the root or crosses at least one breakpoint, so combining it with bisection on a maintained bracket terminates in finitely many steps. Each step costs one $O(n)$ pass, and warm-starting θ from the previous MM iteration keeps the number of passes small. Over the 36,800 root-find calls made by the SQM, SLM and F^3-DkS runs of Table 9, on all seven evaluation graphs and every k, the average is 2.7 passes per call.

D.5 Proof of Proposition 5

For $u > 0$, concavity of the logarithm at $u = 1$ gives $\log u \leq u - 1$, that is, $u \geq 1 + \log u$, with equality if and only if $u = 1$. Let $\mathbf{x} > \mathbf{0}$ and $\mathbf{x}^t > \mathbf{0}$ and apply this inequality with $u = \frac{x_i x_j}{x_i^t x_j^t}$, which gives

$$x_i x_j \;\geq\; x_i^t x_j^t \Big(1 + \log x_i + \log x_j - \log x_i^t - \log x_j^t\Big). \tag{20}$$

Every entry of $\mathbf{M} = \mathbf{A} + \lambda\mathbf{I}$ is nonnegative, because $\mathbf{A} \geq \mathbf{0}$ and $\lambda \geq 0$, so we may multiply the above by M_{ij} and sum over all i, j without reversing the inequality, and this gives

$$\begin{aligned}\mathbf{x}^\top \mathbf{M}\mathbf{x} &\geq \sum_{i,j} M_{ij} x_i^t x_j^t + \sum_{i,j} M_{ij} x_i^t x_j^t \big(\log x_i - \log x_i^t\big) + \sum_{i,j} M_{ij} x_i^t x_j^t \big(\log x_j - \log x_j^t\big) \\ &= f(\mathbf{x}^t) + 2\sum_{i=1}^{n} x_i^t (\mathbf{M}\mathbf{x}^t)_i \big(\log x_i - \log x_i^t\big),\end{aligned} \tag{21}$$

where the two cross sums were merged using the symmetry $\mathbf{M} = \mathbf{M}^\top$. This is exactly (13), and equality holds at $\mathbf{x} = \mathbf{x}^t$, which is tangency. First-order consistency also holds, since $\partial g_\ell / \partial x_i = 2w_i/x_i$ equals $2(\mathbf{M}\mathbf{x}^t)_i$ at $\mathbf{x} = \mathbf{x}^t$, which is the i-th entry of $\nabla f(\mathbf{x}^t)$.

For the inner problem, $\sum_i w_i \log x_i$ is concave and the feasible set is a polytope, so the Karush–Kuhn–Tucker conditions $w_i/x_i - \theta + \nu_i - \mu_i = 0$, with the same complementary slackness as above,

are necessary and sufficient. A coordinate with $w_i > 0$ cannot vanish, because of the logarithmic barrier; if $0 < x_i < 1$ then $x_i = w_i/\theta$, and if $x_i = 1$ then $\mu_i = w_i - \theta \geq 0$, that is, $w_i/\theta \geq 1$. Hence $x_i = \min\{1, w_i/\theta\}$, and $\theta > 0$ is fixed by $\sum_i \min\{1, w_i/\theta\} = k$, whose left-hand side is continuous and non-increasing in θ, tends to $|\{i : w_i > 0\}| \geq k$ as $\theta \to 0^+$ and is at most $\mathbf{1}^\top \mathbf{w}/\theta \to 0$ as $\theta \to \infty$. ■

D.6 Proof of Theorem 6

Feasibility. Each iteration sets $\mathbf{x}^{t+1}$ to the maximizer of a strictly concave function over $\mathcal{C}_k^n$, and the starting point $(k/n)\mathbf{1}$ is an element of $\mathcal{C}_k^n$. Both are therefore elements of $\mathcal{C}_k^n$ for any value of $c > 0$, so $\mathbf{x}^t \in \mathcal{C}_k^n$ for all t by induction.

Monotonicity for $c \geq 1$. Since $\mathbf{d} \in \mathcal{D}$ and $\mathbf{d} > \mathbf{0}$, we have $c\,\mathbf{d} \geq \mathbf{d}$ pointwise, and $\mathcal{D}$ is upward closed by Remark 11, so $c\,\mathbf{d} \in \mathcal{D}$ and $c\,\mathbf{d} > \mathbf{0}$. The step is then exactly one iteration of (12) with the admissible weights $c\,\mathbf{d}$, and Lemma 8 applies. Note that the ascent argument uses only that the weights employed at iteration t are admissible and positive; a different admissible weight vector may therefore be used at each iteration without affecting the conclusion.

Monotonicity for $c < 1$. In this regime $c\,\mathbf{d}$ may leave $\mathcal{D}$ and the surrogate need no longer minorize f, so the trial point produced by (11) carries no guarantee. The safeguard evaluates $f(\mathbf{x}^{t+1})$ and, whenever $f(\mathbf{x}^{t+1}) < f(\mathbf{x}^t)$, discards the trial point and recomputes the iterate with the unscaled weights $\mathbf{d}$, which lie in $\mathcal{D}$ and are positive and therefore give an increase by the previous paragraph. In either branch $f(\mathbf{x}^{t+1}) \geq f(\mathbf{x}^t)$, which is the claim. ■

Remark 13 (Cost of the safeguard test)**.** Evaluating $f(\mathbf{x}^{t+1}) = (\mathbf{x}^{t+1})^\top \mathbf{A}\mathbf{x}^{t+1} + \lambda\|\mathbf{x}^{t+1}\|^2$ requires the product $\mathbf{A}\mathbf{x}^{t+1}$, which iteration $t+1$ needs in any case. The test therefore costs nothing unless it fires, and when it fires it costs one extra sparse matrix–vector product.

D.7 Proof of Lemma 8

Fix t and abbreviate $g(\cdot) = g_{\mathrm{q}}(\cdot \mid \mathbf{x}^t)$. The function g is separable with Hessian $-2\,\mathrm{Diag}(\mathbf{d})$ and is therefore strongly concave, so that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$g(\mathbf{x}) \leq g(\mathbf{y}) + \nabla g(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) - \sum_i d_i (x_i - y_i)^2. \tag{22}$$

Apply (22) with $\mathbf{y} = \mathbf{x}^{t+1}$, the maximizer of g over the convex set $\mathcal{C}_k^n$, and $\mathbf{x} = \mathbf{x}^t \in \mathcal{C}_k^n$. First-order optimality of $\mathbf{x}^{t+1}$ gives $\nabla g(\mathbf{x}^{t+1})^\top (\mathbf{x}^t - \mathbf{x}^{t+1}) \leq 0$, whence

$$g(\mathbf{x}^t) \leq g(\mathbf{x}^{t+1}) - \sum_i d_i (x_i^t - x_i^{t+1})^2. \tag{23}$$

Combining (23) with the minorization inequality $f(\mathbf{x}^{t+1}) \geq g(\mathbf{x}^{t+1})$ and the tangency identity $f(\mathbf{x}^t) = g(\mathbf{x}^t)$ yields the first inequality of (15). The second follows from $d_i \geq d_{\min} > 0$ for every i, and the third from nonnegativity of the squared norm. ■

D.8 Proof of Lemma 9

The set $\mathcal{C}_k^n$ is the intersection of the box $[0,1]^n$, which is compact and convex, with the hyperplane $\mathbf{1}^\top \mathbf{x} = k$, which is closed and convex, so it is compact and convex; it is nonempty because $1 \leq k \leq n$, the vector $(k/n)\mathbf{1}$ being one of its elements. For the bounds on the objective, note first that $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \lambda\|\mathbf{x}\|^2 \geq 0$ on $\mathcal{C}_k^n$, since $\mathbf{A} \geq \mathbf{0}$, $\lambda \geq 0$ and $\mathbf{x} \geq \mathbf{0}$. For the upper bound, recall from the exact value of the maximum over the scaled simplex [25] that

$$\max_{\mathbf{x} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{x} = k} \mathbf{x}^\top (\mathbf{A} + \lambda \mathbf{I})\mathbf{x} = k^2 \Big(1 + \frac{\lambda - 1}{\omega}\Big), \tag{24}$$

which is at most k^2 whenever $\lambda \leq 1$. Since $\mathcal{C}_k^n$ is contained in the scaled simplex $\{\mathbf{x} \geq \mathbf{0} : \mathbf{1}^\top \mathbf{x} = k\}$, the same bound holds on $\mathcal{C}_k^n$. ■

D.9 Proof of Theorem 10

(i) Monotonicity and convergence of the objective. Lemma 8 gives $f(\mathbf{x}^{t+1}) \geq f(\mathbf{x}^t)$ for every t, so the real sequence $\{f(\mathbf{x}^t)\}$ is non-decreasing, and Lemma 9 bounds it above by k^2. A non-decreasing sequence of reals that is bounded above converges, so $f(\mathbf{x}^t) \to f^\star$ for some $f^\star \leq k^2$.

(ii) Asymptotic regularity. Summing the ascent inequality (15) over $t = 0, \ldots, T-1$ telescopes to

$$d_{\min} \sum_{t=0}^{T-1} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \;\leq\; \sum_{t=0}^{T-1} \sum_{i=1}^{n} d_i (x_i^{t+1} - x_i^t)^2 \;\leq\; f(\mathbf{x}^T) - f(\mathbf{x}^0) \;\leq\; k^2 - f(\mathbf{x}^0), \quad (25)$$

where the last step uses Lemma 9. Letting $T \to \infty$ gives $\sum_{t \geq 0} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \leq (k^2 - f(\mathbf{x}^0))/d_{\min} < \infty$. A convergent series has vanishing terms, so $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \to 0$. Moreover, the smallest of the first T terms is no larger than their average, so $\min_{t<T} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \leq (k^2 - f(\mathbf{x}^0))/(d_{\min} T)$, which is the stated $O(1/\sqrt{T})$ rate.

(iii) Existence of limit points. Every iterate lies in $\mathcal{C}_k^n$, which is compact by Lemma 9, so $\{\mathbf{x}^t\}$ is bounded and the Bolzano–Weierstrass theorem supplies a convergent subsequence. Any limit point $\bar{\mathbf{x}}$ is feasible, because $\mathcal{C}_k^n$ is closed, and satisfies $f(\bar{\mathbf{x}}) = f^\star$ by continuity of f together with part (i).

(iv) Stationarity. Let $\bar{\mathbf{x}}$ be a limit point, say $\mathbf{x}^{t_j} \to \bar{\mathbf{x}}$. By part (ii) the successive iterates become arbitrarily close, so $\mathbf{x}^{t_j+1} \to \bar{\mathbf{x}}$ as well. For every $\mathbf{x} \in \mathcal{C}_k^n$, optimality of $\mathbf{x}^{t_j+1}$ for the surrogate $g_{\mathrm{q}}(\cdot \mid \mathbf{x}^{t_j})$ over the convex set $\mathcal{C}_k^n$ gives $\nabla g_{\mathrm{q}}(\mathbf{x}^{t_j+1} \mid \mathbf{x}^{t_j})^\top (\mathbf{x} - \mathbf{x}^{t_j+1}) \leq 0$. The gradient $\nabla g_{\mathrm{q}}(\mathbf{x} \mid \mathbf{y}) = 2(\mathbf{M} + \mathbf{D})\mathbf{y} - 2\mathbf{D}\mathbf{x}$ is jointly continuous in its two arguments, so passing to the limit along the subsequence yields

$$\big(2(\mathbf{M} + \mathbf{D})\bar{\mathbf{x}} - 2\mathbf{D}\bar{\mathbf{x}}\big)^\top (\mathbf{x} - \bar{\mathbf{x}}) = \nabla f(\bar{\mathbf{x}})^\top (\mathbf{x} - \bar{\mathbf{x}}) \leq 0 \qquad \text{for all } \mathbf{x} \in \mathcal{C}_k^n, \quad (26)$$

where the identity uses the first-order consistency established at the end of the proof of Proposition 2. This is the first-order necessary condition for (3); since the constraints are affine no constraint qualification is required, and $\bar{\mathbf{x}}$ is a Karush–Kuhn–Tucker point. The argument is the standard successive-upper-bound analysis, whose hypotheses — a continuous, first-order consistent surrogate that is strongly concave on a convex compact set — are met here [29].

SLM. The same four steps apply verbatim once the iterates are confined to a compact subset of the relative interior of $\mathcal{C}_k^n$, which is what the weight floor of Section 3.4 enforces. The surrogate g_ℓ is concave and first-order consistent, its maximizer over $\mathcal{C}_k^n$ is unique, and the quantitative bound (15) is replaced by the plain monotonicity (14), which is all that parts (i), (iii) and (iv) use. ■

E Relation to the Existing Solvers

The solvers already proposed for the loaded relaxation and its relatives are close enough to ours in purpose to deserve a careful comparison rather than a single clause, and the comparison turns on one question, namely what is left free once the formulation has been fixed.

Cardinality-constrained binary quadratic optimization has been studied through an extreme-point-pursuit formulation, with DkS as the headline application [23]. The objective is forced to be strictly concave by a spectral shift, so that its optimum is attained at a vertex of the convex hull, and that optimum is then approached along a homotopy path deforming an easy convex problem into the concave one. Every step along the path is a top-k selection, and at the end of the path the iteration coincides with the truncated power method [37], which had been derived from an entirely different starting point. The Frank–Wolfe scheme arrives at the same oracle by another route, its linear maximization subproblem being a top-k selection of the gradient [25], and the exact-penalty method we run as the EP-Prox baseline builds a penalty from the error-bound principle and applies non-convex proximal gradient, with the proximal operator once again a top-k selection [24].

That shared oracle is the right place from which to see the difference. A top-k selection leaves nothing to choose. Once the inner problem is a sort, the iteration map is pinned down by the formulation, and the only quantities the designer still holds act on the trajectory from outside the map. The construction of Section 3.2 keeps a quadratic term $-\sum_i d_i x_i^2$, and the price of keeping it is that the inner problem becomes a weighted projection onto the capped simplex instead of a sort. The return is the weight

Table 3: The evaluation graphs, after the preprocessing described in this appendix. The last column is the clique number ω where it is known.

Graph	n	m	$\bar{\deg}$	$\deg_{\max}$	$\lambda_{\max}(\mathbf{A})$	ω
facebook	4,039	88,234	43.7	1,045	162.4	69
web-Stanford	281,903	1,992,636	14.1	38,625	448.1	61
as-skitter	1,696,415	11,095,298	13.1	35,455	–	67
soc-pokec	1,632,803	22,301,964	27.3	14,854	–	29
ca-HepTh	9,877	25,973	5.3	65	–	–
ca-CondMat	23,133	93,439	8.1	279	–	–
com-dblp	317,080	1,049,866	6.6	343	–	–

vector $\mathbf{d}$, which ranges over the whole set $\mathcal{D}$ and which Appendix H shows to be worth a factor of two in achieved density. The degree-adaptive family $s_i(\beta)$, the observation that the weights select the basin, and everything built on it in Sections 3.5 and 4 have no counterpart in a design whose inner problem is a sort, because there is no free weight there to vary.

Two smaller points of contact are worth recording. The SLM surrogate of Section 3.4 rests on the nonnegativity of $\mathbf{A}$ rather than on the curvature of the objective, and yields a multiplicative water-filling update that has no analogue among the solvers above. And the top-k oracle those solvers share is not foreign to our iteration either, since it is the limit our step approaches as $c \to 0$ in Section 4; that the baselines built on it behave similarly to one another in our experiments, and differently from the SQM family, follows from the oracle rather than from any one formulation.

F Experimental Setup

This appendix records the details needed to reproduce the numbers in the main text. We describe first the data and the preprocessing, then the protocol and the baselines, and finally the synthetic graphs on which the shrinkage constant is fixed.

F.1 Graphs and Preprocessing

The seven evaluation graphs are taken from the SNAP collection [22] in their raw edge-list form. Each file is read once, and the following normalizations are applied before anything else happens. The edge list is symmetrized, so that a directed pair (i, j) contributes both A_{ij} and A_{ji}; self-loops are dropped, which matters because $\mathbf{A}$ is required to have a zero diagonal for the diagonal loading argument of Section 3.1 to be exact; duplicate edges are merged, so that the result is a simple graph; and the vertex identifiers, which in the SNAP files are not contiguous, are relabelled to $\{0, \dots, n-1\}$. Isolated vertices produced by this relabelling are kept, since removing them would change n and hence the meaning of the uniform starting point $(k/n)\mathbf{1}$. The result is stored as a compressed sparse row matrix of float32 values with sorted indices.

Table 3 reports the sizes after this processing, and they reproduce the standard published sizes for these graphs exactly, which is the strongest single check we have that the preprocessing is correct. The degree columns also show how heavy-tailed the graphs are. The largest degree exceeds the mean by three to four orders of magnitude, which is precisely the regime in which an isotropic spectral shift is a poor preconditioner, as Section 3.2 argues and Section 3.5 confirms.

The web-Stanford and soc-pokec graphs are natively directed and are symmetrized as above, so the densities we report refer to the underlying undirected graph. This is the usual convention in the DkS literature and is what makes our numbers comparable with previously published ones.

F.2 Protocol

Every method receives the identical sparse matrix and returns a score vector $\mathbf{x} \in \mathbb{R}^n$, which is rounded by keeping the k largest entries. Ties are broken by index, which affects nothing of substance but makes every run deterministic. The quantity reported throughout is the density (2) of the selected set, computed from its indicator vector exactly as Section 2 defines it. Rounding every method in

the same way is deliberate. Several of the baselines are usually paired with a bespoke rounding rule or a local-search post-processing step, and including those would confound the comparison of the optimizers, which is what we are interested in. A local-search refinement applied uniformly to all methods raises everyone's numbers by a similar margin and does not change the ordering.

All timings are wall-clock, measured on one desktop machine with an AMD Ryzen 5 5600H processor, six cores at $3.3\,\mathrm{GHz}$, and $8\,\mathrm{GB}$ of RAM, single-threaded except where the underlying BLAS chooses otherwise, and no GPU is used at any point. The reported time is the time to produce the score vector; the shared preprocessing, that is, reading the graph and computing the weight vector, is measured separately and reported in Appendix M, since it is amortized over all values of k on a given graph. Every graph is run at every value of k. On the two largest, as-skitter and soc-pokec, we cap every iterative method at $1800\,\mathrm{s}$ of wall-clock time per instance. The cap is generous, being more than seven times the $233\,\mathrm{s}$ that the slowest MM iteration needs at that scale, and it binds on a single method, the AdamW reparameterization, which is by an order of magnitude the most expensive in the comparison. We state it explicitly because it is the one place where the budgets are not identical across graphs. F^3-DkS reaches the densities reported for those two graphs in 11 and 18 seconds, two orders of magnitude inside the cap.

F.3 Baselines

We took considerable care over the baseline implementations, since a comparison against under-tuned classical solvers would be worth nothing. Each baseline was implemented following the description given in its own paper, and each was run with the parameter values and the iteration budget that paper specifies. We do not restate those settings here, because they are the authors' choices rather than ours, and no baseline was run in a configuration of our own devising. The six baselines are the following, each stated together with the weakness that motivates the present work.

- **Greedy (2001)** [11], which takes the $k/2$ highest-degree vertices and then adds the $k/2$ vertices with the most neighbours among them. It never revisits a vertex once chosen, so a high-degree vertex lying outside every dense pocket is kept for good, and the quality of what it returns degrades as k grows.
- **TPM (2013)** [37], the truncated power method, which alternates a power iteration with a hard thresholding to k nonzeros. The thresholding destroys the ascent property of the power step, so the iteration carries no monotonicity guarantee, and its support is driven by the leading eigenvector, which on a heavy-tailed graph concentrates on a few hubs.
- **Rank-1 (2014)** [28], the spectral procedure that reads a k-subset off the leading eigenvector of $\mathbf{A}$. A single eigenvector describes the graph only where the spectrum has a pronounced gap, which real networks do not provide, and an eigendecomposition is needed before any subgraph is produced at all.
- **FW (2025)** [25], the Frank–Wolfe solver for exactly the relaxation (3), whose linear oracle is a top-k selection of the gradient. That oracle leaves the designer nothing to choose, the iteration map being fixed once the formulation is, and the step length is decided by a schedule external to it.
- **AdamW (2025)** [25], the smooth reparameterization in which the constraint is absorbed into a change of variables and the resulting unconstrained problem is optimized by AdamW. The reparameterization introduces a learning rate, a weight decay and a schedule, none of which the formulation determines, and each pass is by an order of magnitude the most expensive in the comparison.
- **EP-Prox (2026)** [24], the exact-penalty proximal-gradient method built from the error-bound principle, whose proximal operator is again a top-k selection. The penalty weight has to be driven along a continuation path, and the guarantee it rests on holds only once that weight is large enough, a threshold not available before the graph is seen.

F.4 The Synthetic Graphs

The shrinkage constant c of Section 4 is fixed on synthetic graphs alone. No evaluation graph, and no graph resembling one, is used for that purpose. An instance is drawn as follows.

Table 4: Effect of the diagonal loading λ on the density (%) reached by SQM from the uniform start. Best per row in bold, second best underlined.

Graph	k	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$	$\lambda = 10$
facebook	20	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
facebook	100	**98.4**	**98.4**	**98.4**	**98.4**	**98.4**	<u>98.3</u>
facebook	500	**24.7**	**24.7**	**24.7**	**24.7**	**24.7**	**24.7**
facebook	1000	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	<u>9.9</u>
web-Stanford	20	87.9	87.9	<u>89.5</u>	<u>89.5</u>	**90.0**	47.4
web-Stanford	100	**53.7**	**53.7**	<u>53.6</u>	53.5	53.5	7.8
web-Stanford	500	**23.5**	**23.5**	**23.5**	**23.5**	**23.5**	<u>2.0</u>
web-Stanford	1000	**11.7**	**11.7**	**11.7**	**11.7**	**11.7**	<u>1.1</u>

A Chung–Lu backbone [10] on n vertices is generated, with n drawn uniformly from $[2000, 12000]$. Each vertex i receives an expected degree w_i sampled from a power law with exponent $\gamma \sim \mathcal{U}[2.1, 3.0]$, rescaled so that the mean expected degree is $\bar{d} \sim \mathcal{U}[4, 25]$, and $\frac{1}{2}\sum_i w_i$ edges are drawn with endpoint probabilities proportional to $\mathbf{w}$. Self-loops and duplicates are removed as for the real graphs. Into this backbone we plant between one and three dense subgraphs, each of a size drawn uniformly from $[20, \min\{400, \max\{25, n/10\}\}]$ and with an internal edge probability drawn uniformly from $[0.35, 1.0]$, so that the planted pockets range from moderately dense to complete cliques. The size k is redrawn for every instance, log-uniformly on $[10, \min\{1000, n/4\}]$, so that the whole range of k used at evaluation time is represented rather than one regime.

This generator is designed to reproduce two properties of real networks, namely a heavy degree tail, which is what makes the isotropic spectral shift a poor preconditioner and therefore what makes the choice of $\mathbf{d}$ matter, and a small number of exceptionally dense pockets, which is what creates the competing basins of attraction. It reproduces neither the community structure nor the clustering coefficient of real graphs, and it is deliberately much smaller than the evaluation graphs, the largest synthetic graph having 1.2×10^4 vertices against 1.7×10^6 for as-skitter.

G Effect of the Diagonal Loading

Section 3.1 fixes $\lambda = 1$ and gives the argument for it in Proposition 1. Table 4 reports what actually happens as λ varies, with everything else held fixed.

The pattern is the one the theory predicts, and it is worth spelling out because it explains why a large loading is not simply the safer choice. At $\lambda = 0$ the relaxation is the plain quadratic form over the capped simplex; it is not tight, and the iteration is free to converge to an interior point that rounds badly. As λ grows the objective becomes more convex and its maximizers are pushed towards the vertices of $\mathcal{C}_k^n$, which is what tightness requires. But convexity here is not free. Once λ is large the weights $\mathbf{d}$ must grow in order to keep $\mathbf{A} + \mathrm{Diag}(\mathbf{d} + \lambda\mathbf{1})$ positive semidefinite, the MM step becomes correspondingly short, and the iteration freezes near wherever it was started; at $\lambda = 10$ the method essentially returns a rounded version of its own starting point. The value $\lambda = 1$ sits at the point where the relaxation is tight and the landscape is still gentle enough to move in, which is what Proposition 1 predicts and which these numbers support.

H The Minorizer Family

Table 5 is the family study on which Section 3.5 rests. Every row is produced by the same solver from the same uniform starting point $\mathbf{x}^0 = (k/n)\mathbf{1}$ and with the same iteration budget, so that the only quantity varying down a block is the minorization weight vector.

Two features of the family deserve to be spelled out before the numbers. The first is what the exponent β controls. Since $s_i(\beta) = \deg_i^{-\beta/2} \sum_{j \in \mathcal{N}(i)} \deg_j^{\beta/2}$ compares the degree of a vertex with a β-weighted average of the degrees of its neighbours, β decides how aggressively the hubs of the graph are discounted relative to their neighbourhoods, and hence which vertices are allowed to grow fastest in the early iterations. The second is where the isotropic spectral shift sits in that trade-off. It is governed by the single worst direction of the spectrum of $\mathbf{A}$, which on a graph with a heavy degree

Table 5: Density (%) reached by different members of the minorizer family from the same uniform starting point and the same iteration budget. Best per column in bold, second best underlined.

Graph	Minorizer	k=20	k=100	k=500	k=1000
facebook	$\beta = -1$	**100.0**	**98.4**	**24.7**	**10.0**
	$\beta = 0$	**100.0**	**98.4**	**24.7**	**10.0**
	$\beta = \frac{1}{2}$	**100.0**	**98.4**	**24.7**	**10.0**
	$\beta = 1$	**100.0**	**98.4**	**24.7**	**10.0**
	$\beta = 2$	**100.0**	**98.4**	**24.7**	**10.0**
	spectral	**100.0**	**98.4**	**24.7**	**10.0**
	SLM	**100.0**	**98.4**	**24.7**	**10.0**
web-Stanford	$\beta = -1$	5.3	1.0	**23.5**	**11.7**
	$\beta = 0$	89.5	<u>53.5</u>	**23.5**	**11.7**
	$\beta = \frac{1}{2}$	**100.0**	**84.5**	**23.5**	**11.7**
	$\beta = 1$	<u>98.9</u>	39.7	9.4	10.4
	$\beta = 2$	88.4	40.1	<u>23.1</u>	**11.7**
	spectral	88.4	42.8	9.4	4.5
	SLM	91.6	41.1	<u>23.1</u>	<u>11.6</u>

tail is dictated by a handful of hub vertices, and that one conservative weight is then imposed on all n coordinates, including the overwhelming majority which are sparsely connected and could safely move much further. The iteration consequently crawls, which is the behaviour the table exhibits.

The size of the spread is the first thing to note. On web-Stanford at $k = 100$ the achieved density varies by more than a factor of two between the best and the worst member of the family, with the isotropic spectral shift and the signless-Laplacian member falling between the two extremes. The effect is not a small-k curiosity either. It persists at the largest subgraph sizes we examine, where the isotropic shift falls to 4.5% at $k = 1000$ while the intermediate members of the degree family hold 11.7%, and it becomes extreme at small k, where the two ends of the family can differ by almost the entire range of the density scale on a single instance. Since every one of these algorithms is launched from the same point and ascends monotonically to a Karush–Kuhn–Tucker point of the same problem, the spread can be attributed neither to the initialization nor to a failure of any member to converge.

Two further points deserve comment. The first concerns the instance dependence of the best member. If one were to pick a single exponent β on the basis of one graph and one value of k and then use it everywhere, the choice would be wrong often enough to matter. The best member changes with k on a fixed graph and changes across graphs at a fixed k; there is no member that is best everywhere, and the members that are worst on some instances are catastrophically rather than merely mildly worse. On web-Stanford the exponent $\beta = -1$ returns 5.3% at $k = 20$, where $\beta = \frac{1}{2}$ returns a clique, and 1.0% at $k = 100$, where $\beta = \frac{1}{2}$ returns 84.5%; at $k = 1000$ the isotropic spectral shift returns 4.5% against 11.7% for the intermediate members. This is what makes the choice of weights a genuine design decision rather than a formality.

The second concerns why β has this effect at all. The weight $s_i(\beta) = \deg_i^{-\beta/2} \sum_{j \in \mathcal{N}(i)} \deg_j^{\beta/2}$ is a ratio between a vertex's own degree and a β-weighted average of its neighbours' degrees, so β controls whether hubs or their peripheries receive the larger minorization weight, and hence which of them is allowed to grow faster in the early iterations. With β large and positive the weights on high-degree vertices become very large, those coordinates are held nearly fixed, and the iteration is driven by the periphery; with β very negative the opposite happens, and a few hubs saturate immediately and drag the iterate into whichever basin they belong to. The intermediate values $\beta \in \{0, \frac{1}{2}\}$ balance the two, which is why they are the most reliable members and why $\beta = \frac{1}{2}$ is our default.

I THE STARTING POINT

Table 6 is the study announced in Section 5, in which the minorizer is fixed at $\beta = \frac{1}{2}$ and only the starting point changes.

Table 6: Effect of the starting point. Each cell reports the density (%) of the rounded starting point and, after the arrow, the density reached from it by the same solver. Best final density per column in bold, second best underlined.

Graph	Starting point	k=20	k=100	k=500	k=1000
facebook	uniform	8.9 → **100.0**	2.3 → **98.4**	2.9 → **24.7**	1.8 → **10.0**
	degree	28.4 → **100.0**	49.6 → **98.4**	21.8 → **24.7**	9.2 → **10.0**
	core number	95.8 → **100.0**	87.9 → **98.4**	24.7 → **24.7**	9.9 → **10.0**
	leading eigenvector	100.0 → **100.0**	97.7 → **98.4**	21.9 → **24.7**	8.2 → **10.0**
	greedy peeling (2001) [11]	34.7 → **100.0**	62.4 → <u>98.3</u>	24.1 → <u>24.6</u>	9.6 → **10.0**
	Frank–Wolfe output (2025) [25]	100.0 → **100.0**	98.4 → **98.4**	24.7 → **24.7**	10.0 → **10.0**
web-Stanford	uniform	0.0 → **100.0**	0.0 → **84.5**	0.0 → **23.5**	0.0 → **11.7**
	degree	18.4 → <u>98.9</u>	6.5 → 44.9	3.6 → 9.3	1.8 → 4.7
	core number	19.5 → 89.5	28.0 → 53.8	23.5 → **23.5**	11.3 → **11.7**
	leading eigenvector	88.4 → 88.4	19.6 → 44.9	6.2 → 9.3	3.0 → 4.7
	greedy peeling (2001) [11]	57.9 → 91.6	14.6 → 40.8	9.4 → 16.6	7.0 → <u>10.4</u>
	Frank–Wolfe output (2025) [25]	98.9 → <u>98.9</u>	83.9 → <u>83.9</u>	23.1 → <u>23.1</u>	11.7 → **11.7**

Table 7: Mean density (%) as the shrinkage constant c varies, at a fixed budget of 40 iterations from the uniform start. The shaded row is the value used throughout the paper. Best per column in bold, second best underlined.

Shrinkage	synthetic graphs	14 evaluation instances
$c = 0.02$	**77.80**	63.66
$c = 0.04$	77.58	<u>64.17</u>
$c = 0.06$	77.54	64.06
$c = 0.083$	<u>77.66</u>	**64.20**
$c = 0.1$	77.62	64.16
$c = 0.15$	77.44	63.40
$c = 0.25$	77.03	63.34
$c = 0.5$	76.40	61.62
$c = 1$ (unshrunk MM step)	74.99	56.36
per-instance oracle over c	–	64.21

The single most instructive cell is on web-Stanford at $k = 100$. The Frank–Wolfe output begins there at a density of 83.9% and does not move, whereas the uniform point begins at 0% and finishes at 84.5%, the best of the six candidates. The quality of a starting point as a subgraph is therefore not what determines the quality of the solution reached from it, and the geometric explanation given in the main text is borne out by the table, since it is precisely the nearly binary starting points, which sit closest to a vertex of $\mathcal{C}_k^n$, that improve the least.

Two remarks are worth adding to the discussion in the main text. First, the rows of Table 6 are not equally priced. The Frank–Wolfe starting point is itself the output of a full solve and the leading eigenvector requires a Lanczos run, so those two rows consume far more work than the uniform start, which costs nothing at all. That the uniform start still finishes ahead of both on web-Stanford is the substance of the result. Second, the rows are not monotone in the initial density, and that is precisely the point. What a good starting point has to supply is not a good subgraph but a good *basin*, and the two are different things.

J Fixing the Shrinkage Constant

Section 4 replaces the admissible weights $\mathbf{d}$ by $c\,\mathbf{d}$ with $c < 1$, which lengthens every step beyond what Proposition 2 permits. One constant therefore has to be chosen, and it must be chosen without looking at the evaluation graphs. We fix it on the synthetic graphs of Section F.4 and apply it unchanged to every real network. Table 7 reports the grid on those graphs, together with the corresponding mean over the evaluation instances so that the reader can see how the two relate.

Table 8: Mean density (%) against the iteration budget T, over the 32 instances of facebook, web-Stanford, as-skitter and soc-pokec, both arms using $\beta = \frac{1}{2}$ and the safeguard. Best per column in bold, second best underlined.

Iteration	$T = 40$	$T = 100$	$T = 200$
shrunk step, $c = 0.06$	**65.50**	**65.52**	**65.53**
unshrunk step, $c = 1$	<u>64.45</u>	<u>64.95</u>	<u>65.14</u>

Three observations come out of it. The first is the magnitude of the effect. The unshrunk step, at $c = 1$, is markedly worse at this budget than any of the shrunk ones, on the synthetic graphs and on the real instances alike. The minorization bound is a worst-case statement over all graphs and all iterates, and on the instances one actually meets it is roughly an order of magnitude more conservative than it needs to be.

The second is that the choice within the shrunk regime is not delicate. Over the whole range $c \in [0.02, 0.1]$ the synthetic means lie within 0.3 of a point of one another and the evaluation means within 0.6, so the grid is flat and any value taken from that plateau behaves like any other. This is why we do not read off the synthetic argmax, which the flatness makes essentially arbitrary, but take $c = 0.06$ from the middle of the plateau and use it throughout. The table also shows that it is not the value a search on the evaluation graphs would have returned, which is what one should expect of a constant fixed elsewhere.

The third is that a per-instance oracle over c, that is, the mean obtained by choosing the best constant separately on every instance in hindsight, improves on the best fixed constant only in the second decimal place of a point. The right step length is therefore not strongly instance dependent, which is what makes a single constant an adequate answer.

K Effect of the Iteration Budget

The main comparison gives the classical solvers the iteration counts their own papers specify and F^3-DkS only 40 iterations. Table 8 records what happens when that budget is varied, with everything else held fixed.

The unshrunk iteration, at $c = 1$, gains 0.69 of a point in mean density on going from $T = 40$ to $T = 200$; the shrunk one gains 0.03, because it has converged by $T = 40$. The two facts combine into the practical statement we would take away from this study. At $T = 40$, $c = 0.06$ reaches 65.50%, which is already above the 65.14% that $c = 1$ reaches at $T = 200$. A fivefold reduction in the number of iterations is available simply by taking the longer step that the minorization bound forbids and that the safeguard of Section 4 makes safe. Since the arithmetic per iteration is identical in the two arms, this translates directly into wall-clock time, as Appendix M confirms.

L Behaviour of the Safeguard

Section 4 runs the iteration with weights outside the region in which Proposition 2 guarantees a minorizer, and restores monotonicity by testing the objective and, when the test fails, recomputing the iterate with the unscaled weights. Since the deployed shrinkage is more than an order of magnitude below the value the bound requires, one would expect the safeguard to be working hard, yet the measurements show the opposite. Over the 55 instances on which we instrumented the iteration, the safeguard was invoked on 14 of 2200 iterations, that is, on 0.6% of them, and on 42 of the 55 instances it was never invoked at all. Because the test requires $\mathbf{A}\mathbf{x}^{t+1}$, which the following iteration computes in any case (Remark 13), a firing costs one extra sparse matrix–vector product and a non-firing costs nothing whatsoever.

The interesting half of the picture is what happens when the safeguard is switched off. On eleven of the thirteen instances on which it fires the answer is nothing at all. The long steps are ascent steps anyway, they are simply not *provably* ascent steps, and disabling the test changes neither the trajectory nor the returned subgraph. On web-Stanford at $k = 200$ and $k = 300$, however, the difference is not marginal. With the test disabled, 19 of the 40 iterations strictly decrease the objective and the

returned density falls from 52.2% to 9.0% and from 37.0% to 6.5% respectively. The long steps genuinely do overshoot on those instances, the unprotected iteration oscillates instead of ascending, and the return to the admissible weights is what rescues it.

We draw two conclusions. First, the low average firing rate should not be read as evidence that the safeguard is unnecessary; it is load-bearing exactly where it fires, and the instances on which it fires are not ones we could have identified in advance. Second, this is a concrete illustration of the point made in Section 4, namely that a convergence proof is useful not only for the guarantee it states, but for the test it supplies, which remains meaningful even when the algorithm is deliberately run outside the region the proof covers.

M Running Time

The headline comparison is between F^3-DkS at 40 iterations and the classical solvers at the budgets their own papers specify. The median wall-clock time per instance is 1.4 s for F^3-DkS against 9.1 s for the unshrunk MM step on web-Stanford, 11.1 s against 42.4 s on as-skitter, and 18.0 s against 93.5 s on soc-pokec, which is close to the factor of five that the iteration count implies, since the arithmetic per iteration is identical and the safeguard fires rarely. F^3-DkS is also faster than Frank–Wolfe, the fastest of the relaxation-based baselines, which needs 2.2, 11.3 and 34.0 s on the same three graphs, and it is one to two orders of magnitude cheaper than the AdamW reparameterization, which needs 176 s on web-Stanford and 1804 s on soc-pokec. Greedy peeling is cheaper again by one to three orders of magnitude and correspondingly worse in density, so it is the right choice only where the quality of the answer does not matter much.

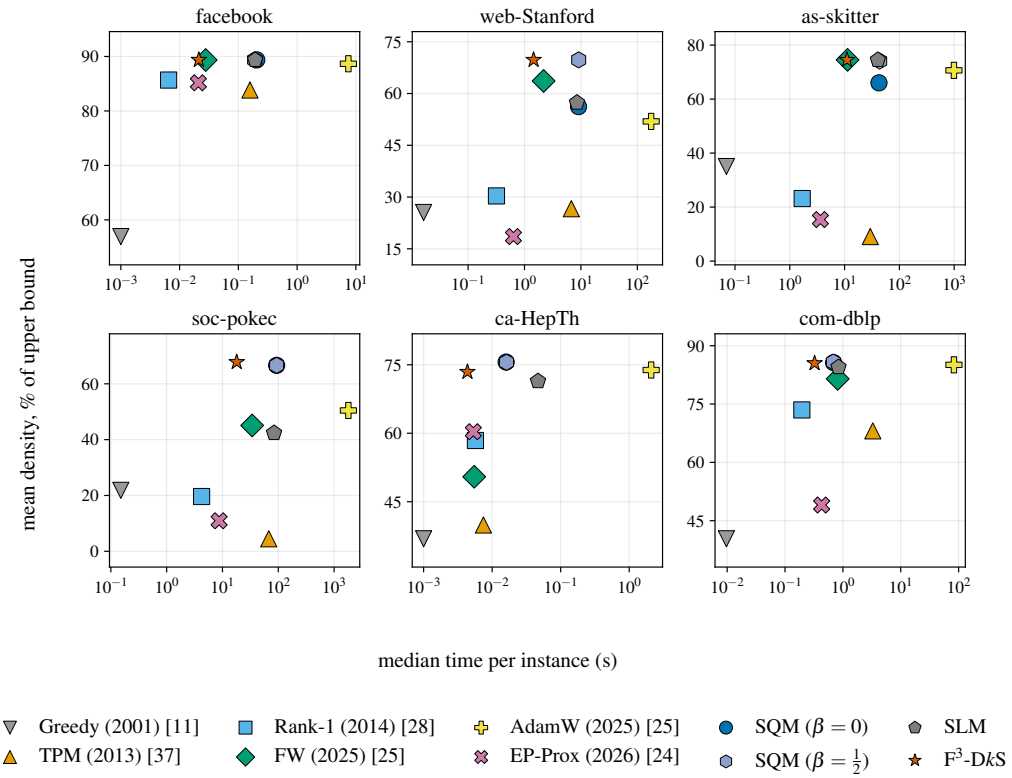

Figure 2: Cost against quality. Each method is summarized by its median wall-clock time and its mean density as a percentage of the upper bound. Up and to the left is better.

Figure 2 places every method on the cost–quality plane, one panel per graph. F^3-DkS is at or within 0.3 of a point of the highest quality on five of the six graphs shown while sitting one to two orders of magnitude to the left of the AdamW reparameterization, which is the slowest method

throughout. On ca-HepTh the same minorizer run to the full budget reaches slightly higher density, at a correspondingly higher cost, which is the trade F^3-DkS is designed to make.

One further cost comparison is worth making explicit. The weights of Proposition 3(ii) and (iii) require one sparse matrix–vector product, computed once per graph and reused for every k, and on web-Stanford this takes a fraction of a second. The isotropic member of the family instead needs $\lambda_{\min}(\mathbf{A})$, which we obtain by a Lanczos iteration on $\sigma\mathbf{I} - \mathbf{A}$. This is the only quantity anywhere in the paper whose cost is not linear in m, and on the larger graphs it accounts for a significant fraction of the total solve time. That the local, degree-adaptive members of the family are both cheaper and, as Table 5 shows, usually better, is one of the more practically useful findings here.

N Complete Results

Tables 9 and 10 give the density reached by every method on every graph at every value of k we ran, of which Table 1 in the main text is a summary. The upper bound row is the spectral bound available for this problem [28].

Reading the two tables together brings out three things. The first is where the advantage of F^3-DkS is concentrated. Below $k = 100$, the range in which a genuinely dense set exists to be found, it improves on the strongest competing solver by three points of density on average, and on individual graphs by very much more, gaining forty-eight points on soc-pokec at $k = 20$, thirty-eight on web-Stanford at $k = 70$ and twenty-seven on web-Stanford at $k = 50$. These are precisely the instances on which the minorization weights decide which basin the iteration enters, and they are what the aggregate rests on.

The second is how narrow the field is everywhere else. Once k passes a hundred the graphs hold no dense set of that size, every method is drawn towards the same value, and the differences shrink accordingly. Away from the two smallest collaboration graphs at small k, no cell of the study separates F^3-DkS from the best of the competing solvers by as much as half a point in either direction, so the comparison there is a tie and is read as one.

The third concerns the bound itself. It is attained at the smallest sizes, where cliques exist, and loosens steadily as k grows, averaging nineteen points at $k = 1000$ against the eight that the best method reaches, so the distance to it is not the distance to the optimum.

Table 9: Density (%) at every k we ran, on facebook, web-Stanford, as-skitter and soc-pokec. Best per column in bold, second best underlined.

Graph	Method	k=10	k=20	k=30	k=50	k=70	k=100	k=200	k=300	k=500	k=700	k=1000
facebook	Greedy (2001) [11]	35.6	34.7	41.8	39.1	55.6	62.4	62.0	41.8	24.1	16.5	9.6
	TPM (2013) [37]	91.1	99.5	97.0	94.1	97.7	98.3	**77.7**	42.1	22.0	12.5	8.2
	Rank-1 (2014) [28]	**100.0**	**100.0**	99.5	99.8	99.3	97.7	**77.7**	42.9	21.9	12.5	8.2
	FW (2025) [25]	**100.0**	**100.0**	**100.0**	**100.0**	99.8	**98.4**	**77.7**	43.5	**24.7**	**16.8**	**10.0**
	AdamW (2025) [25]	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	98.0	77.4	43.2	24.5	15.7	9.7
	EP-Prox (2026) [24]	**100.0**	**100.0**	99.5	99.7	99.5	97.8	60.0	38.3	24.0	15.1	9.4
	SQM (β=0) *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	99.9	**98.4**	**77.7**	**43.6**	**24.7**	**16.8**	**10.0**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**98.4**	**77.7**	**43.6**	**24.7**	**16.8**	**10.0**
	SLM *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	99.9	**98.4**	**77.7**	**43.6**	**24.7**	**16.8**	**10.0**
	F^3-DkS *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**98.4**	**77.7**	**43.6**	**24.7**	**16.8**	**10.0**
	upper bound	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*81.6*	*54.3*	*32.5*	*23.2*	*16.3*
web-Stanford	Greedy (2001) [11]	48.9	57.9	40.7	25.7	21.8	14.6	9.7	9.7	9.4	10.1	7.0
	TPM (2013) [37]	0.0	23.2	74.0	42.0	55.9	41.4	17.4	10.5	6.3	4.7	3.8
	Rank-1 (2014) [28]	51.1	88.4	61.6	38.2	27.7	19.6	13.3	8.6	6.2	4.2	3.0
	FW (2025) [25]	**100.0**	98.9	99.8	73.0	59.3	83.9	52.0	36.9	23.1	**16.6**	**11.7**
	AdamW (2025) [25]	**100.0**	82.6	67.6	60.7	57.0	52.6	42.5	28.9	18.2	13.5	10.1
	EP-Prox (2026) [24]	44.4	18.4	21.8	9.7	6.3	6.5	4.6	29.1	12.1	11.1	9.0
	SQM (β=0) *(ours)*	**100.0**	89.5	77.5	64.3	58.7	53.5	43.5	34.9	**23.5**	16.3	**11.7**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**97.3**	**84.5**	**52.2**	**37.0**	**23.5**	16.3	**11.7**
	SLM *(ours)*	86.7	91.6	89.7	77.1	59.5	41.1	**52.2**	**37.0**	23.1	**16.6**	11.6
	F^3-DkS *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**97.3**	84.4	**52.2**	**37.0**	**23.5**	16.3	**11.7**
	upper bound	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*72.6*	*51.6*	*36.2*
as-skitter	Greedy (2001) [11]	62.2	56.3	59.1	44.3	41.2	34.5	19.3	15.0	14.7	13.5	10.0
	TPM (2013) [37]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	35.6	24.2	15.5
	Rank-1 (2014) [28]	11.1	23.7	21.4	24.2	26.1	27.2	36.5	32.3	18.2	12.5	8.5
	FW (2025) [25]	**100.0**	**100.0**	**100.0**	99.9	**98.9**	**92.8**	**71.6**	**55.9**	**35.7**	**24.4**	15.6
	AdamW (2025) [25]	**100.0**	**100.0**	**100.0**	99.7	96.9	90.1	70.9	46.4	25.3	16.9	12.2
	EP-Prox (2026) [24]	15.6	27.4	23.0	17.1	12.2	10.0	7.8	6.5	5.9	15.4	11.3
	SQM (β=0) *(ours)*	**100.0**	**100.0**	**100.0**	97.4	87.2	72.8	45.7	45.0	27.4	18.7	12.4
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	**100.0**	99.8	98.0	96.9	92.6	71.3	**55.9**	**35.7**	**24.4**	15.6
	SLM *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	98.8	92.7	71.4	**55.9**	**35.7**	**24.4**	15.6
	F^3-DkS *(ours)*	**100.0**	**100.0**	**100.0**	99.6	98.8	92.7	71.5	**55.9**	**35.7**	**24.4**	**15.8**
	upper bound	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*72.0*	*50.5*
soc-pokec	Greedy (2001) [11]	64.4	42.6	27.4	18.4	12.6	9.3	5.4	4.9	3.6	3.2	2.8
	TPM (2013) [37]	0.0	0.0	0.0	0.3	0.2	24.6	0.2	0.2	2.3	1.7	1.1
	Rank-1 (2014) [28]	44.4	43.7	34.9	23.2	16.6	11.8	6.0	4.0	2.4	1.7	1.1
	FW (2025) [25]	80.0	49.5	74.9	41.6	34.4	27.5	16.8	13.4	13.6	9.1	6.4
	AdamW (2025) [25]	**100.0**	21.6	85.3	68.5	56.3	44.6	26.0	16.0	9.9	7.4	5.4
	EP-Prox (2026) [24]	24.4	7.9	4.4	2.9	1.9	1.6	2.2	2.3	5.4	4.7	4.0
	SQM (β=0) *(ours)*	**100.0**	95.3	88.7	74.3	63.5	51.7	**33.4**	24.9	**15.7**	**10.7**	**7.6**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	95.8	88.3	74.0	63.3	51.5	**33.4**	**25.0**	**15.7**	**10.7**	**7.6**
	SLM *(ours)*	86.7	53.7	39.5	30.9	32.4	22.4	17.1	22.8	12.1	8.6	6.3
	F^3-DkS *(ours)*	**100.0**	**97.9**	**93.3**	**81.0**	**70.1**	**56.5**	33.2	23.0	13.6	**10.7**	7.5
	upper bound	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*70.8*	*47.4*	*28.5*	*20.9*	*14.2*

Table 10: Density (%) at every k we ran, on ca-HepTh, ca-CondMat and com-dblp. Best per column in bold, second best underlined.

Graph	Method	k=10	k=20	k=30	k=50	k=70	k=100	k=200	k=300	k=500	k=700	k=1000
ca-HepTh	Greedy (2001) [11]	26.7	23.2	21.6	17.6	13.4	9.3	6.8	4.4	2.6	1.7	1.2
	TPM (2013) [37]	64.4	52.1	38.2	24.2	17.1	13.6	4.7	2.8	1.6	1.2	0.8
	Rank-1 (2014) [28]	**100.0**	**100.0**	**100.0**	45.4	24.8	13.7	4.7	2.9	1.8	1.2	0.8
	FW (2025) [25]	55.6	52.1	34.3	23.3	17.6	18.3	7.7	5.3	2.9	1.9	**1.3**
	AdamW (2025) [25]	**100.0**	**100.0**	**100.0**	**53.0**	35.4	23.4	8.6	5.2	2.8	1.8	1.2
	EP-Prox (2026) [24]	17.8	**100.0**	**100.0**	43.8	26.5	18.7	8.3	4.9	2.6	1.7	1.1
	SQM (β=0) *(ours)*	**100.0**	**100.0**	**100.0**	**53.0**	**35.7**	**23.5**	**9.1**	**5.5**	**3.0**	**2.0**	**1.3**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	**100.0**	**100.0**	**53.0**	**35.7**	23.4	**9.1**	**5.5**	**3.0**	**2.0**	**1.3**
	SLM *(ours)*	**100.0**	**100.0**	**100.0**	47.5	29.1	18.3	8.4	5.4	**3.0**	**2.0**	**1.3**
	F^3-DkS *(ours)*	**100.0**	**100.0**	**100.0**	47.5	34.4	20.5	8.7	**5.5**	**3.0**	**2.0**	**1.3**
	upper bound	*100.0*	*100.0*	*100.0*	*63.3*	*45.0*	*31.3*	*14.3*	*8.9*	*5.0*	*3.5*	*2.4*
ca-CondMat	Greedy (2001) [11]	48.9	47.9	32.9	22.4	20.0	15.6	9.4	6.5	4.0	2.9	2.0
	TPM (2013) [37]	93.3	77.9	59.5	47.4	34.6	22.7	10.7	7.2	4.3	2.9	2.0
	Rank-1 (2014) [28]	88.9	66.3	52.4	37.5	28.4	21.4	11.6	7.5	4.4	3.1	2.1
	FW (2025) [25]	**100.0**	75.8	72.2	48.0	31.9	22.9	11.7	**8.3**	**4.8**	**3.3**	**2.2**
	AdamW (2025) [25]	**100.0**	**100.0**	88.0	**48.4**	32.8	23.0	11.3	7.7	4.4	3.0	2.0
	EP-Prox (2026) [24]	**100.0**	56.3	34.3	25.0	21.4	16.3	9.8	6.8	4.0	2.7	1.9
	SQM (β=0) *(ours)*	**100.0**	**100.0**	**92.2**	47.4	**36.0**	**25.5**	**12.7**	8.2	**4.8**	**3.3**	**2.2**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	94.2	72.4	47.4	**36.0**	**25.5**	**12.7**	8.2	**4.8**	**3.3**	**2.2**
	SLM *(ours)*	**100.0**	94.2	72.4	48.2	33.7	22.9	12.2	**8.3**	**4.8**	**3.3**	**2.2**
	F^3-DkS *(ours)*	**100.0**	94.2	72.4	48.2	33.7	22.9	12.5	**8.3**	**4.8**	**3.3**	**2.2**
	upper bound	*100.0*	*100.0*	*100.0*	*77.3*	*54.9*	*38.3*	*19.0*	*12.7*	*7.6*	*5.4*	*3.8*
com-dblp	Greedy (2001) [11]	64.4	41.6	38.2	21.4	19.8	16.4	35.9	19.8	12.1	7.0	4.1
	TPM (2013) [37]	**100.0**	**100.0**	82.8	58.4	**100.0**	**100.0**	42.2	20.2	8.2	4.5	2.4
	Rank-1 (2014) [28]	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	42.2	20.2	8.2	4.5	2.4
	FW (2025) [25]	**100.0**	**100.0**	97.0	**100.0**	**100.0**	**100.0**	43.1	30.7	13.2	7.7	5.0
	AdamW (2025) [25]	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**50.7**	31.3	14.9	9.4	5.5
	EP-Prox (2026) [24]	11.1	7.9	6.4	5.7	**100.0**	**100.0**	**50.7**	29.9	13.0	7.4	5.1
	SQM (β=0) *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**50.7**	**31.9**	15.1	**9.6**	**5.8**
	SQM (β=$\frac{1}{2}$) *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**50.7**	**31.9**	**15.4**	**9.6**	**5.8**
	SLM *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**50.7**	30.7	14.5	9.2	5.2
	F^3-DkS *(ours)*	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**50.7**	**31.9**	15.3	9.4	5.7
	upper bound	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*58.2*	*38.7*	*23.2*	*16.6*	*11.6*