

Graph Coarsening Enables Escaping Local Optima in Compute-Constrained Nonconvex MAX-CUT

M. Harper Langston
 Pierre-David Letourneau
 Dalton Jones
 Harris Teague
 Peter Scott
 Mingu Lee
 Richard A. Lethin
 Christopher Lott
 Roland Memisevic
 Fatih Porikli
Qualcomm AI Research

HLANGSTO@QTI.QUALCOMM.COM
 PLETOURN@QTI.QUALCOMM.COM
 DAL TJONE@QTI.QUALCOMM.COM
 HTEAGUE@QTI.QUALCOMM.COM
 PETER.SCOTT@NYU.EDU
 MINGUL@QTI.QUALCOMM.COM
 RLETHIN@QTI.QUALCOMM.COM
 CLOTT@QTI.QUALCOMM.COM
 RMEMISEV@QTI.QUALCOMM.COM
 FPORIKLI@QTI.QUALCOMM.COM

Abstract

Because its binary optimization landscape contains exponentially many local optima, MAX-CUT is highly sensitive to initialization. We study Particle Polynomial Relaxation (PPR), a moment-based nonconvex continuous reformulation possessing an advantageous optimization landscape. We warm-start PPR with Combinatorial Multigrid (CMG), a spectral graph-coarsening method that produces structure-aware initial solutions in near-linear time without training data. On **t2g20** ($n = 400$), a benchmark instance with known exact optimum, the combined CMG+PPR pipeline reaches 97.5% of optimum on average and 98.4% on the best run. An extended variant with discrete local-search refinement reaches the published optimum on all sixty Big Mac instances evaluated by a recent neural branch-and-bound method. Using a sparse CSR representation, the same approach scales to G-set instances with up to 20,000 vertices, reducing memory requirements by up to 3333 $\times$ while maintaining a consistent improvement over the CMG warm start. Gains are largest on sparse weighted graphs; on dense graphs we complement the pipeline with spectral sparsification.

1. Introduction

MAX-CUT asks for a partition of a graph’s vertices that maximizes the total weight of crossing edges (Karp, 1972; Garey and Johnson, 1979). With $x_i \in \{-1, +1\}$, it can be written as the nonconvex quadratic program

$$\min_{x \in \{-1, +1\}^n} -\frac{1}{4} \sum_{(i,j) \in E} w_{ij}(1 - x_i x_j), \quad (1.1)$$

equivalently as a ground-state search for the associated Ising energy (Barahona, 1982; Lucas, 2014). We first note that the flip neighborhood on $\{-1, +1\}^n$ may contain exponentially many local optima; so, a local method’s terminal cut can be determined more by the initial point than by the amount of search performed afterward. Second, the classical Goemans-Williamson 0.878 approximation ratio (Goemans and Williamson, 1995), tight under the

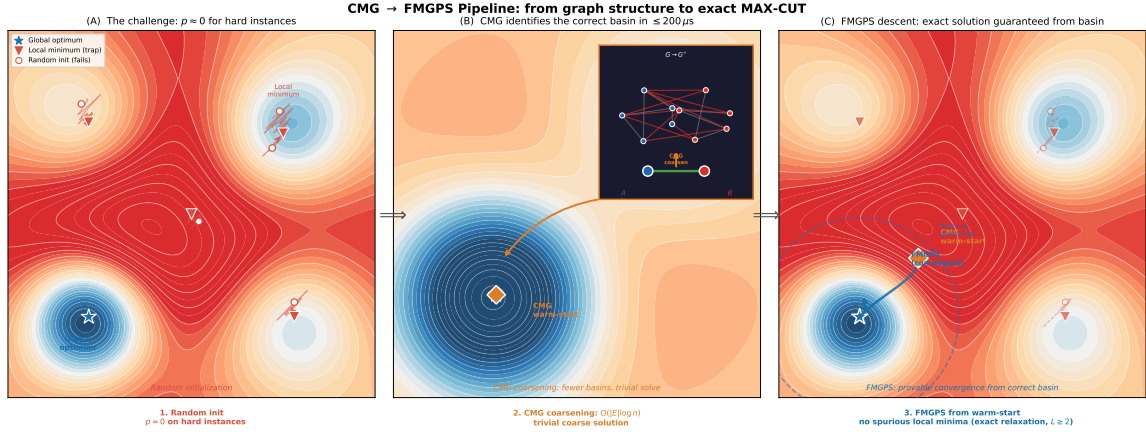


Figure 1: Schematic comparison of random and structure-aware initialization. **Left:** random restarts are likely to terminate in different local optima of the flip landscape. **Center:** CMG coarsening summarizes the graph into larger structural units and returns a warm start. **Right:** PPR is then initialized in that basin and can make multi-node moves that are inaccessible to one-bit local search.

Unique Games Conjecture in the worst case (Khot et al., 2007), does not describe how close a nonconvex solver can get on structured instances when initialization already captures useful graph information.

We investigate whether two complementary ideas can be combined effectively. *First:* a continuous moment-based reformulation which possesses a highly-advantageous optimization landscape, namely, every suboptimal point has zero escape barrier when its mixing weight is free to move (Appendix A.1 and (Letourneau et al., 2023)). *Second:* a graph coarsening warm start that places the finite-particle solver in a useful basin. We find that this combination is effective on sparse and weighted structured instances and that it scales to $n = 20,000$ when the coupling matrix is stored sparsely. The same experiments also identify a clear boundary: on dense or nearly homogeneous graphs, CMG often has little structure to exploit, and the pipeline provides little or no improvement over the warm start. Appendix B gives additional background, mechanical details, and a small end-to-end walk-through. The main text focuses on the solver, initialization, and empirical behavior of the composed pipeline.

Relation to learned combinatorial solvers. Recent learned MAX-CUT methods supply structural information through training. OptGNN (Yau et al., 2024) learns an over-parameterized Burer-Monteiro (Burer and Monteiro, 2003) update, while Chen et al. (Chen et al., 2026) train a GNN to predict SDP certificates used inside branch-and-bound. Our approach uses a different source of structure: spectral graph coarsening on the input graph itself. This requires no labels, no training distribution, and no neural inference hardware. We do not claim uniform superiority, but on the six Biq Mac families evaluated by (Chen et al., 2026), our full heuristic pipeline reaches the same published optima on all 60 instances without their training setup (Table 2).

2. Method: moment relaxation with a graph-coarsening warm start

2.1. Continuous Reformulation through Discrete/Particle Product Measures

Our proposed PPR method is inspired by the work of Lasserre (Lasserre, 2001) and Letourneau et al. (Letourneau et al., 2023) which both address the problem of global polynomial optimization through a moment-based reformulation. In this case, consider a *level- l discrete product measure* to be defined as,

$$\mu^{(l)}(\cdot) = \bigotimes_{i=1}^D \left(\sum_{r=1}^R w_i^{(l,r)} \delta_{\bar{x}_i^{(r)}} \right), \quad w_i^{(l,\cdot)} \in \Delta^{R-1}, \quad (2.1)$$

where $\Delta^{R-1} := \{w \in \mathbb{R}_{\geq 0}^R : \sum_r w_r = 1\}$ is the probability simplex; R is thus the number of particles (point masses, or atoms) used to represent each coordinate's marginal, and $l = 1, \dots, L$ indexes one of L independently optimized such mixtures (*levels*), combined below through a mixing weight β . Its degree- n marginal moment in coordinate i is

$$a_{n,i}^{(l)} := \int T_n(x_i) d\mu^{(l)}(x) = \sum_{r=1}^R w_i^{(l,r)} T_n(\bar{x}_i^{(r)}), \quad (2.2)$$

where $\{T_n(x)\}$ is a one-dimensional graded polynomial basis. Given a polynomial $p(x) = \sum_{\mathbf{n} \in \text{supp}(p)} p_{\mathbf{n}} T_{\mathbf{n}}(x)$,¹ define the *single-level functional*

$$g(w) := \mathbb{E}_{\mu(w)}[p(x)] = \sum_{\mathbf{n} \in \text{supp}(p)} p_{\mathbf{n}} \prod_{i=1}^D a_{n_i,i}(w_i), \quad w \in (\Delta^{R-1})^D, \quad (2.3)$$

and, for $L \geq 1$ levels with a *free* mixing weight $\beta \in \Delta^{L-1}$, the objective

$$F(w, \beta) := \sum_{l=1}^L \beta_l g(w^{(l)}) \quad \text{on} \quad \mathcal{F} := (\Delta^{R-1})^{D \times L} \times \Delta^{L-1}. \quad (2.4)$$

Call level l *present* if $\beta_l > 0$, and write $V_l := g(w^{(l)})$ for its value. We prove in Appendix A.1 that the continuous optimization problem,²

$$\min_{(w, \beta) \in \mathcal{F}} F(w, \beta) \quad (2.5)$$

is equivalent to the original MAX-CUT problem (Eq. 1.1); despite being nonconvex, every suboptimal point has zero escape barrier when β is free to move³. In this case, the objective polynomial $p(x)$ is the max-cut objective of Eq. 1.1, and the particles' support is $\{-1, 1\}$. Each $\mu_i^{(l)}$ is represented by R atoms with softmax weights and tanh-parameterized locations, making moment-matrix positivity guaranteed and leaving an unconstrained problem

-
1. $\mathbf{n} \in \mathbb{N}$ is a *multi-index*. $\text{supp}(p)$ is the set of nonzero coefficients of $p(\cdot)$ (in this specific basis) and $T_{\mathbf{n}}(x) := \prod_{i=1}^D T_{n_i}(x)$ is a multi-dimensional polynomial.
 2. Our implementation parametrizes w through softmax to enforce its simplex constraint with no fundamental differences to conclusions; β is not parametrized or optimized at all, held fixed at the uniform average.
 3. Can be strictly improved through a feasible path originating at this point

with $O(LnR)$ variables (linear in n for fixed L, R , versus $O(n^2)$ for the GW SDP (Appendix B.5.2)). We solve this problem over w with L-BFGS (Liu and Nocedal, 1989) across a penalty-continuation schedule, holding β fixed at the uniform average $\beta_l = 1/L$ rather than optimizing it.⁴

2.2. Graph-coarsening initialization

At larger scales, finite-particle relaxation benefits from a strong initialization; we use Combinatorial Multigrid (CMG) (Langston et al., 2019; Koutis et al., 2011) to construct a *structure-aware warm start*. CMG builds a coarsen→solve→uncoarsen graph hierarchy: at each level, vertex i joins the cluster of its strongest neighbor $B[i] = \arg \max_{j \sim i} w_{ij}$ when $M[i]/D[i] \geq 1/8$, where $M[i] = \max_{j \sim i} w_{ij}$ and $D[i] = \sum_{j \sim i} w_{ij}$; otherwise it remains a singleton. We directly solve the coarsest graph, and the solution is projected back to the original graph with local-flip refinement at each level (total cost $O(|E| \log n)$).

The resulting initialization is entirely graph-driven and requires no training data. Its value comes from exposing large-scale structure before continuous refinement. When edge weights are nearly uniform, however, little structure is available to exploit, the hierarchy compresses poorly, and the warm-start advantage diminishes. This behavior appears in the experiments and is consistent with the mechanism of the method.

2.3. The composed pipeline

CMG returns a binary warm start x_{cmg} that is already refined by single-bit flips. From there, PPR searches over a continuous particle representation that can make simultaneous multi-node changes. In the stronger configuration, the best binary solution from pass p is used to initialize pass $p + 1$. A single 2-opt (Croes, 1958) sweep is applied after the final PPR pass (applying 2-opt after every pass tends to lock the solution into a two-bit local basin too early (Table 7)). For dense graphs with more than $3n$ edges, we first apply the Spielman-Srivastava (S-S) effective-resistance sparsification (Spielman and Srivastava, 2011); this step is a no-op for the already-sparse structured instances that dominate our results.

This is the pipeline we evaluate in Section 3: Spielman-Srivastava spectral sparsification (Spielman and Srivastava, 2011) (when needed) feeds a CMG warm start, which PPR refines on the finest graph across one or more passes, closed by a single 2-opt sweep; an optional discrete tail (Kernighan-Lin, 3-opt, restarts, and a memetic GA) can be appended afterward. Figure 2 (Appendix B.4) draws this end to end, and Appendix B.3 summarizes how the three main stages complement each other.

3. Experiments

We evaluate our pipeline (Section 2.3 with details in Appendix B) on structured toroidal Ising instances from the Biq Mac library (Rendl et al., 2010) (branch-and-cut exact optima as ground truth) and on real G-set instances up to $n = 20,000$; ratios are cut/optimum. We report 0.878 as the poly.-time *worst-case* guarantee (Goemans and Williamson, 1995) (not GW’s typical achieved cut, which is far higher on structured instances); where we do contrast against an achieved GW cut we use a measured SDP-plus-rounding solve.

4. App. A, Prop. 4 gives the weaker barrier bound this restriction leaves intact (details of the w -only solve in Alg. 1)

Structural warm start dominates raw search on the hard small instance. The $n=15$ instance shows that CMG finds the optimum in $30\mu s$, five orders of magnitude less time than the 21 random PPR restarts consume, and the best of those restarts still reaches a cut of only 37 (one short of the optimal 38); none of the 21 restarts finds the exact optimum. No additional restart budget closes the gap. The instance is small, but it isolates the failure mode of random initialization: once the solver enters the wrong basin, further local optimization does not recover the optimum.

Rank $R > 1$ is necessary for the composed pipeline to improve on the warm start. Table 1 reports the pipeline on two toroidal families with published exact optima (of six tested; the other four follow the same pattern), together with a dense expander contrast. The $R = 1$ configuration barely moves off CMG; $R \geq 2$ lifts every structured instance 4-15 points past CMG alone, well past the 0.878 worst-case line. On the expander g05_60, by contrast, CMG is already at 0.96 and *no* configuration improves it, as the dense-regime failure mode described above predicts.

Table 1: S-S→CMG→PPR, mean cut/OPT ratio over three seeds vs. known optima (Rendl et al., 2010); g05_60 here is a 5-instance subset of the family, vs. the full 10-instance mean in Table 2. P-32,8 denotes $L = 32, R = 8$.

Family	n	CMG	P-16, $R = 1$	P-32,8	Notes
t3g5	125	0.828	0.883	0.978	3-D toroidal
t2g20	400	0.830	0.832	0.907	2-D toroidal
<i>Reference:</i>				0.878	GW guarantee
g05_60	60	0.963	0.963	0.963	Expander (5/10); CMG already strong

Pushed further, on the largest exact-optimum instance in this comparison the pipeline reaches 97.5% mean and 98.4% best. t2g20 ($n = 400$) is the largest instance in Table 1 with a published exact optimum, and the weakest single-shot result above. Chaining five PPR passes lifts CMG’s 0.832 to 0.952; a single closing 2-opt sweep lifts this to **0.975** mean, **0.984** best seed (Table 7 and Figure 5, both in Appendix B.5.6). Interleaving 2-opt after *every* pass instead locks the solution into a 2-opt basin after pass 1, finishing at only 0.957, so the ordering (search, then polish once) matters. Restart count saturates quickly ($k=40 \rightarrow 80 \rightarrow 160$ buys +0.003 then +0.001), confirming that the quality of the warm-start basin, not restart breadth, is the bottleneck (Figure 6).

With a discrete polish tail, our pipeline reaches the published optimum on all sixty Biq Mac instances. Chen et al. (Chen et al., 2026) train a GNN to accelerate B&B to a *proven* optimum on 6 Biq Mac families (60 instances). Our pipeline with a discrete tail (multi-pass PPR $\rightarrow$ 2-opt (Croes, 1958) $\rightarrow$ Kernighan-Lin (Kernighan and Lin, 1970) $\rightarrow$ 3-opt (Lin and Kernighan, 1973) $\rightarrow$ 200-restart $\rightarrow$ memetic GA (Moscato and Cotta, 2003; Chu and Beasley, 1998)) matches the published optimum on *every one of the* 60 (Table 2, Figure 9) in 8.5-130.5s per instance, against B&B times up to 40,930s (Mosek SDP bound) or 4,509s (learned proxy). While they prove optimality, we report a heuristic cut that matches a published one. However, it answers the comparison the learned solvers set up, with no training data or inference hardware. The g05 expanders also confirm

Table 1’s dense-regime pattern: warm start alone already sits at 0.97-0.98, so light PPR closes most of the gap.

Table 2: 10 instances chosen from the 6 Biq Mac families (Chen et al., 2026); full pipeline matches all 60 published optimum. Mean single-core per-instance wall clock used for timing comparisons; B&B times from (Chen et al., 2026), Tab. 3.

Family	n	Ours (heuristic)			(Chen et al., 2026) B&B (s)	
		CMG	Pipeline	time (s)	Vanilla (s)	Neural (s)
g05_60	60	0.975	1.000	23.9	53	9.5
g05_80	80	0.977	1.000	63.8	1170	156
g05_100	100	0.982	1.000	130.5	40930	4509
pm1s_80	80	0.817	1.000	8.5	610	81
pm1s_100	100	0.818	1.000	13.7	10382	1058
w01_100	100	0.881	1.000	12.7	11349	1074

At the continuous phase, L-BFGS beats Adam; but, the discrete tail sets the final cut, not the optimizer. PPR solves each penalty-continuation stage with a single L-BFGS call; we swapped that inner solve for Adam (Kingma and Ba, 2015), full-batch gradient descent (GD), and a level-subsampled stochastic variant based on stochastic approximation (Robbins and Monro, 1951) on eight instances, with the warm start, restarts, continuation schedule, and downstream discrete tail held fixed (full per-instance table in Appendix B.5.12). Pre-tail, the four disagree by up to 7.4%⁵, but the identical discrete tail collapses that spread to $\leq 0.7\%$ on every instance: L-BFGS just gets there the cheapest.

The warm start’s advantage persists to $n=20,000$. A dense coupling matrix requires $8n^2$ bytes (3.2 GB at $n = 20,000$), impractical for sparse-signed graphs ($|E| \approx 2n$). CSR representations storing only $2|E|$ nonzeros resolve this. Running CMG $\rightarrow$ light PPR $\rightarrow$ 2-opt on 6 real G-set instances, $n = 5K-20K$ (Table 3), reduces memory by $833-3333\times$ with linear scaling in n . Gains over CMG warm start through the continuous-relaxation path alone, $\Delta_{2\text{opt}}$, remain in a tight $1.19-1.21\times$ band, matching the behavior at $n = 125-1000$ (Figure 7). Adding a discrete tail⁶ raises the final gain to $\Delta_{\text{final}} = 1.32-1.33\times$, again with no observable decay. For G77 and G81, the full pipeline reaches **98.7%** and **99.0%** (the continuous-relaxation path alone reaches 90.8% and 89.8%) of the externally verified reported optima (Cosm’s (Zick et al., 2026) 9940 and 14060). A tabu-diversified tail (Glover, 1986; Glover and Laguna, 1997) reaches 13962 (99.3% of 14060) on G81 in ~ 213 s and provides the study’s only internal optimality certificate, verified in a further ~ 2.4 s across 40 connected cores (Appendix B.5.13).

4. Discussion: comparison and limitations

Comparable assumptions to bespoke heuristics. Cosm (Zick et al., 2026) reports optimality on G77 and G81 using an external exact solver rather than an internal proof. Our ratio-to-optimum results rely on the same type of reference. We do not aim to match Cosm’s runtime performance; our contribution is a general framework that com-

5. L-BFGS’s curvature edge leaves Adam up to 6.2% behind, and GD, with neither momentum nor curvature, stalls at the CMG warm start on G1 and is $11-30\times$ slower per instance.

6. Kernighan-Lin, 3-opt, a memetic GA, and an iterated local search: the retuned configuration of Table 14.

Table 3: Sparse CSR path, four of six G-set instances, single core. “mem”: dense/CSR storage ratio. “2opt”: cut after CMG→PPR→2-opt only, with its own gain over CMG, $\Delta_{2\text{opt}}$ (this is the configuration the pipeline name describes). “final”: cut after also appending the retuned discrete tail of Table 14 (KL/3-opt/GA/ILS), with Δ_{final} . ratio-to-OPT given at both stages, only where an externally verified optimum exists.

Instance	n	mem	cmg	2opt	$\Delta_{2\text{opt}}$	final	Δ_{final}	ratio-to-OPT (2opt/final)
G57	5000	833.3×	2600	3142	1.208	3460	1.331	—
G67	10000	1666.7×	5198	6258	1.204	6872	1.322	—
G77	14000	2333.3×	7454	9022	1.211	9810	1.316	0.908 / 0.9869
G81	20000	3333.3×	10574	12626	1.194	13922	1.317	0.898 / 0.9902

bines reusable structure-aware components (nonconvex relaxation and spectral coarsening) with a broader class of constraints rather than a solver tailored to MAX-CUT.

When the method provides little benefit. On dense, well-conditioned graphs, CMG coarsening succeeds almost everywhere, so the warm start is already near the local-flip limit and PPR yields no measurable improvement (Table 1). On uniform-weight Erdős-Rényi graphs, the CMG grouping criterion $M[i]/D[i] \geq 1/8$ rarely holds once $n > 16$, preventing meaningful coarsening. In both cases, the pipeline exploits weight heterogeneity, so these graph families provide little structure to exploit. This behavior is consistent with our design goal of a *structure-aware method*.

Open questions. A discrete polishing stage (Kernighan–Lin, 3-opt, or a memetic GA) closes most of the remaining gap on the hardest instances. However, a random-start baseline with the same polishing budget matches or exceeds the CMG-seeded pipeline on roughly half of these cases. Once sufficient restarts are available, final solution quality is often determined more by the polishing stage than by the warm start (Table 13; see also Table 12). Understanding this trade-off and extending the approach beyond $n = 20,000$ under fixed compute remain promising directions.

5. Conclusion

A structure-aware CMG warm start combined with a particle-based continuous relaxation achieves near-optimal MAX-CUT solutions across a wide range of problem sizes. The approach reaches 97.5% mean (best-seed 98.4%) of the exact optimum on **t2g20** ($n = 400$), and 98.7–99.0% of the exact optimum on the two largest G-set instances with externally verified optima. These gains persist up to 20,000 vertices without observable degradation, although they naturally diminish on graph families with little exploitable structure. Appendix A.1 discusses the theoretical motivation that continuous relaxation eliminates escape barriers under free mixing, while the CMG warm start escapes the limitations of purely local search. In compute-constrained regimes, this combination provides a robust advantage; however, when large discrete-search budgets become available, the polishing stage increasingly determines final solution quality. Consistent with the tail-ablation study (Table 13), a random start carried through the same high-budget discrete tail can match or exceed the CMG-seeded pipeline on roughly half of the hardest instances tested.

References

- Francisco Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.
- Nicolas Boumal, Vladislav Voroninski, and Afonso S. Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.
- Samuel Burer and Renato D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.
- Hao Chen, Chendi Qian, Christopher Morris, Andrea Lodi, and Can Li. Solving Max-Cut to global optimality via feasibility-preserving graph neural networks, 2026. arXiv:2605.07113.
- Paul C. Chu and John E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.
- G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.
- Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.
- Yinhao Dong, Pan Peng, and Ali Vakilian. Learning-augmented streaming algorithms for approximating MAX-CUT. In *Innovations in Theoretical Computer Science (ITCS)*, 2025. arXiv:2412.09773.
- Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.
- Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.
- Qi Huangfu and J. A. Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. HiGHS solver, <https://highs.dev/>.
- Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

- George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
- Brian W. Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.
- Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.
- Ioannis Koutis, Gary L. Miller, and David Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115(12):1638–1646, 2011.
- Dieter Kraft. A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, DFVLR, German Aerospace Center, 1988.
- M. Harper Langston, Mitchell Tong Harris, Pierre-David Letourneau, Richard Lethin, and James Ezick. Combinatorial multigrid: Advanced preconditioners for ill-conditioned linear systems. In *IEEE High Performance Extreme Computing Conference (HPEC)*, 2019.
- Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
- Pierre-David Letourneau, Dalton Jones, Matthew Morse, and M. Harper Langston. An efficient framework for global non-convex polynomial optimization over the hypercube, 2023. arXiv:2308.16731.
- Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.
- Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.
- Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:5, 2014.
- Pablo Moscato and Carlos Cotta. A gentle introduction to memetic algorithms. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, 2003.
- Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010. Biq Mac Library, <https://biqmac.aau.at/>.
- Herbert Robbins and Sutton Monroe. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

- Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.
- Po-Wei Wang, Wei-Cheng Chang, and J. Zico Kolter. The mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints. In *arXiv preprint arXiv:1706.00476*, 2017.
- Morris Yau, Nikolaos Karalias, Eric Lu, Jessica Xu, and Stefanie Jegelka. Are graph neural networks optimal approximation algorithms? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2310.00526.
- Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.
- Kenneth M. Zick, Nikhil Shukla, and Alexander Marakov. Cosm: Collective switched motion for fast and accurate sparse Ising optimization, 2026. arXiv:2605.30355.

Appendix A. Particle Polynomial Relaxation (PPR) for MAX-CUT: full algorithm

Algorithm 1 specializes the particle-parameterization solver of [Letourneau et al. \(2023\)](#) to the MAX-CUT setting. The only constraint is the binary/box penalty $\frac{\rho}{2} \|\text{mean}_l(a2_l) - \mathbf{1}\|^2$ appearing in the objective below, which drives each marginal’s second moment to one and is therefore zero exactly when every marginal is supported on $\{-1, +1\}$. There is no equality-constraint dual loop; instead, K independent restarts are run through a fixed penalty-continuation schedule $\rho_1 < \dots < \rho_S$. Pseudocode for the rest of the pipeline is presented in Appendix C.

Algorithm 1: PPR for MAX-CUT

Input: J ($n \times n$, normalized); warm start x_0 ; L, R, k ; sat; penalty schedule $\rho_1 < \dots < \rho_S$
Output: best ± 1 cut found
 best $\leftarrow -\infty$
for $\text{restart} = 1, \dots, k$ **do**
 $u_{l,0,i} \leftarrow \text{atanh}(\text{sat} \cdot x_{0,i}) + \mathcal{N}(0, 0.02)$ for all l // warm-started slot
 all other $u_{l,r,i} \sim \mathcal{N}(0, 0.02)$ // unwarmed particles
 for $\rho = \rho_1, \dots, \rho_S$ **do**
 form marginals $a_l = \sum_r w_r \tanh(u_{l,r,\cdot})$, $a2_l = \sum_r w_r \tanh(u_{l,r,\cdot})^2$
 $F(u) \leftarrow -\frac{1}{2L} \sum_l a_l^\top J a_l + \frac{\rho}{2} \|\text{mean}_l(a2_l) - \mathbf{1}\|^2$
 run L-BFGS on $F(u)$ to convergence
 end
 for each (l, r) **do**
 $\hat{x}_i \leftarrow \text{sign}(\tanh(u_{l,r,i}))$ for all i
 local-flip refine $\hat{x}$ on G
 keep $\hat{x}$ if $\text{Cut}(\hat{x}) > \text{best}$
 end
end
return best $\hat{x}$ found across all restarts and (l, r)

A.1. Theory and Proofs

In this appendix, we demonstrate that the proposed continuous reformulation for the MAX-CUT problem possesses an advantageous optimization landscape. Namely, we show that for every non-optimal feasible point, there exists a path of descent (not necessarily strict). In other words, the reformulated objective does not possess any strict local minima other than global minima (which may or may not be strict either). The arguments are processed without the softmax parametrization for simplicity. It can be shown that the results still hold following such parametrization.

Weights, levels, and the objective Recall that *level- l discrete product measure* is

$$\mu^{(l)}(\cdot) = \bigotimes_{i=1}^D \left(\sum_{r=1}^R w_i^{(l,r)} \delta_{\hat{x}_i^{(r)}} \right), \quad w_i^{(l,\cdot)} \in \Delta^{R-1}, \quad (\text{A.1})$$

where $\Delta^{R-1} := \{w \in \mathbb{R}_{\geq 0}^R : \sum_r w_r = 1\}$ is the probability simplex. Its degree- n marginal moment in coordinate i is

$$a_{n,i}^{(l)} := \sum_{r=1}^R w_i^{(l,r)} T_n(\bar{x}_i^{(r)}), \quad (\text{A.2})$$

so $a_{0,i}^{(l)} \equiv 1$ is pinned by the simplex constraint. Define the *single-level functional*

$$g(w) := \mathbb{E}_{\mu(w)}[p(x)] = \sum_{\mathbf{n} \in \text{supp}(p)} p_{\mathbf{n}} \prod_{i=1}^D a_{n_i,i}(w_i), \quad w \in (\Delta^{R-1})^D, \quad (\text{A.3})$$

and, for $L \geq 1$ levels with a *free* mixing weight $\beta \in \Delta^{L-1}$, the objective

$$F(w, \beta) := \sum_{l=1}^L \beta_l g(w^{(l)}) \quad \text{on} \quad \mathcal{F} := (\Delta^{R-1})^{D \times L} \times \Delta^{L-1}. \quad (\text{A.4})$$

Call level l *present* if $\beta_l > 0$, and write $V_l := g(w^{(l)})$ for its value.

Oscillation Let

$$C := \max_{\bar{x} \in \bar{X}} p(\bar{x}) - \min_{\bar{x} \in \bar{X}} p(\bar{x}) = \max_{\bar{x} \in \bar{X}} p(\bar{x}) - p^* \quad (\text{A.5})$$

be the (finite) oscillation of p over the grid. Since $g(w)$ in (A.3) is an expectation of p under a probability measure supported on $\bar{X}$, every feasible w satisfies $p^* \leq g(w) \leq p^* + C$, hence

$$|g(w) - g(w')| \leq C \quad \text{for all feasible } w, w'. \quad (\text{A.6})$$

Definition 1 (Escape barrier) For $L \geq 1$ let F and $\mathcal{F}$ be as in (A.4). The escape barrier at a feasible point (w, β) is

$$B(w, \beta) := \inf_{\pi} \left[\max_{t \in [0,1]} F(\pi(t)) - F(\pi(0)) \right], \quad (\text{A.7})$$

the infimum taken over all continuous feasible paths $\pi : [0, 1] \rightarrow \mathcal{F}$ with $\pi(0) = (w, \beta)$ and $F(\pi(1)) < F(\pi(0))$ (so $B := +\infty$ when no such path exists). Note $B \geq 0$ always, since $t = 0$ is admissible in the maximum.

The barrier dichotomy needs one elementary fact in both of its directions, so we record it first.

Theorem 2 (Value equivalence) $\pi^* := \min_{(w, \beta) \in \mathcal{F}} F(w, \beta) = p^*$.

Proof Step 1 ($\pi^* \leq p^*$). Let $\bar{x}^* \in \bar{X}$ be a global minimizer of p over the grid (which exists since $\bar{X}$ is finite). For each i let r_i^* be the index with $\bar{x}_i^{(r_i^*)} = \bar{x}_i^*$, and set $w_i^{(1,r)} := \mathbb{I}[r = r_i^*]$ (one-hot), with $\beta_1 = 1$. This is feasible, and F evaluates exactly to $p(\bar{x}^*) = p^*$ term by term. Hence $\pi^* \leq p^*$.

Step 2 ($\pi^* \geq p^*$). For *any* feasible w , each level's contribution $\mathbb{E}_{\mu^{(l)}(w)}[p]$ is, by definition (A.1), the expectation of p under a genuine probability measure supported on $\bar{X}$, a

convex combination of point masses at grid points, since $\mu^{(l)}$ is a product of finitely supported marginals. Every such expectation is $\geq \min_{\bar{x} \in \bar{X}} p(\bar{x}) = p^*$. Any convex combination of these expectations⁷ preserves the bound, so $F(w, \beta) \geq p^*$ for every feasible (w, β) , giving $\pi^* \geq p^*$.

Combining, $\pi^* = p^*$. ■

Theorem 3 (Lateral-escape: zero barrier at every suboptimal point) *Let $L \geq 2$ and let $(w, \beta) \in \mathcal{F}$ satisfy $F(w, \beta) > p^*$. Then for every $\delta \in (0, 1]$ there is a continuous feasible path $\pi : [0, 1] \rightarrow \mathcal{F}$ with $\pi(0) = (w, \beta)$ such that*

$$F(\pi(1)) = p^* \quad \text{and} \quad \max_{t \in [0, 1]} F(\pi(t)) \leq F(w, \beta) + \delta C. \quad (\text{A.8})$$

Consequently $B(w, \beta) = 0$: the net descent $F(w, \beta) - p^* > 0$ is a fixed positive quantity independent of δ , while the barrier that must be crossed to realize it is at most δC and hence arbitrarily small. Conversely, if $F(w, \beta) = p^*$ then $F \geq p^*$ everywhere on $\mathcal{F}$ (Step 2 of Theorem 2) rules out any feasible path reaching a lower value, so $B(w, \beta) = +\infty$. Together the two directions give the full dichotomy: $B(w, \beta) = 0$ iff $F(w, \beta) > p^*$, and $B(w, \beta) = +\infty$ if and only if $F(w, \beta) = p^*$, with no intermediate case (no suboptimal point is protected by a barrier of positive height).

Proof Let $\bar{x}^* \in \bar{X}$ be a global minimizer of p over the grid and let $e(\bar{x}^*) \in (\Delta^{R-1})^D$ be the corresponding one-hot weight vector, so $g(e(\bar{x}^*)) = p(\bar{x}^*) = p^*$ (this is the only place a specific weight vector is needed, and it requires nothing of R beyond $\bar{x}^* \in \bar{X}$; no Vandermonde or rank hypothesis enters). Write $V_l := g(w^{(l)})$ and call level l *present* if $\beta_l > 0$. Two cases.

Case 1: some level l_1 is absent ($\beta_{l_1} = 0$). *Phase A (exactly flat).* Holding β and every other level fixed, move $w^{(l_1)}$ to $e(\bar{x}^*)$ along the straight line $(1-t)w^{(l_1)} + te(\bar{x}^*)$, feasible for all t by convexity of each Δ^{R-1} . Since F depends on $w^{(l_1)}$ only through the product $\beta_{l_1}g(w^{(l_1)})$ and $\beta_{l_1} = 0$, F is *constant* along the whole of Phase A: the rise is exactly 0. *Phase B (exactly affine, strictly decreasing).* Now move β_{l_1} from 0 to 1, taking the removed mass from the other levels proportionally, i.e. $\beta_{l_1}(t) = t$ and $\beta_l(t) = (1-t)\beta_l$ for $l \neq l_1$. This stays in Δ^{L-1} , and since every V_l is now fixed, $F(t) = (1-t)F(w, \beta) + tp^*$ is exactly affine with constant slope $p^* - F(w, \beta) < 0$: strictly decreasing, ending at p^* . The path never rises at all, so (A.8) holds with room to spare, for every δ .

Case 2: every level is present ($\beta_l > 0$ for all l). Fix $\delta \in (0, 1]$, choose $l_1 \in \arg \max_l V_l$ (so $V_{l_1} \geq \sum_l \beta_l V_l = F(w, \beta)$), and set $\delta' := \min(\delta, \beta_{l_1}) > 0$.

Phase A (lateral: non-increasing, and exactly flat at a critical point). Holding all $w^{(l)}$ fixed, decrease β_{l_1} from β_{l_1} to δ' , redistributing the removed mass $\beta_{l_1} - \delta' \geq 0$ proportionally among the other levels: $\beta_l(t)$ interpolates affinely, remaining in Δ^{L-1} throughout. Mass is thereby moved *out of* a level of maximal value V_{l_1} and *into* a convex combination of levels each of value $\leq V_{l_1}$, so F is non-increasing along Phase A; the rise is ≤ 0 .⁸

7. In particular, a free $\beta \in \Delta^{L-1}$, not merely a uniform average

8. When all present levels share a common value, exactly the situation forced at any critical point by Step 2 of Theorem 2, Phase A is *identically flat*: it is a genuine lateral move, neither ascending nor descending, and it is available regardless of how deeply each individual level is trapped.

Phase B (the only ascent, bounded by $\delta' C$). With β now fixed at $\beta_{l_1} = \delta'$, move $w^{(l_1)}$ to $e(\bar{x}^*)$ along the straight line as in Case 1. Only the term $\beta_{l_1} g(w^{(l_1)}) = \delta' g(w^{(l_1)})$ varies, so for every t

$$F(\pi(t)) - F(\pi(t_A)) = \delta' [g(w^{(l_1)}(t)) - V_{l_1}] \leq \delta' C \leq \delta C, \quad (\text{A.9})$$

where t_A is the end of Phase A and the bound uses $|g(w) - g(w')| \leq C$ from (A.6). Since Phase A did not raise F , the running maximum over Phases A-B is at most $F(w, \beta) + \delta C$.

Phase C (exactly affine, strictly decreasing). Now $V_{l_1} = p^*$ and all V_l are again fixed; move β_{l_1} from δ' to 1, taking mass from the others proportionally as in Case 1's Phase B. F is exactly affine in the interpolation parameter with endpoint value p^* , hence non-increasing (strictly decreasing unless the starting value already equals p^* , in which case the endpoint is reached immediately), and $F(\pi(1)) = p^*$. Phase C therefore adds no new maximum, and (A.8) holds.

In both cases the path satisfies (A.8), and $F(\pi(1)) = p^* < F(w, \beta)$ makes π admissible in Definition 1; letting $\delta \downarrow 0$ gives $B(w, \beta) = 0$. The converse direction stated above needs no further argument beyond $F \geq p^*$ everywhere. $\blacksquare$

The zero-barrier statement is deliberately *weaker* than “local minimum $\Rightarrow$ global minimum,” and the reason is not an artifact of the fixed grid. The stronger statement would require a path whose derivative is strictly negative *immediately* at $t = 0$, hence a mechanism for spawning fresh mass at $\bar{x}^*$ inside a level that already carries mass without first emptying it. Here the level to be repositioned is first emptied to mass δ' (otherwise repositioning it would cost $O(1)$ rather than $O(\delta)$), and the emptying is lateral, not descending. The result is a path that descends overall but not from the outset, which is exactly the distinction between $B = 0$ (proved here) and “not a local minimum” (false here). The operative consequence survives: p^* is reachable from anywhere at arbitrarily small cost, so a solver willing to accept a transient increase of size δC is never blocked by a genuine energy barrier, only by the difficulty of finding the lateral direction.

What this buys the deployed algorithm. Theorem 3's mechanism needs β to move: Case 1 needs an *absent* level ($\beta_{l_1} = 0$) and Case 2 needs β_{l_1} *shrunk* to a small δ' before the repositioning move. Algorithm 1, and every solver run in this paper, never varies β at all: it always averages all L levels uniformly, $\beta_l = 1/L$. Eq. (A.4) is stated for a general, possibly free $\beta \in \Delta^{L-1}$ because the construction and Theorem 2 hold at that generality (Step 2's bound is proved for *any* convex combination, uniform included), but Theorem 3's zero barrier is specific to a β that is free to move, and so does not, by itself, describe what Algorithm 1 is guaranteed. The following proposition states what survives when β is pinned at the uniform average.

Proposition 4 (Barrier bound under the fixed uniform weighting) *Let $\beta_l = 1/L$ for every l , held fixed, and let w be feasible with $F(w, \frac{1}{L}\mathbf{1}) > p^*$. Define the fixed- β escape barrier $B_{\text{fix}}(w)$ exactly as in Definition 1, but with the infimum restricted to continuous paths that vary only w (as Algorithm 1 does; β is not a free coordinate of its search). Then*

$$B_{\text{fix}}(w) \leq C/L.$$

Proof Write $V_l := g(w^{(l)})$ as before and choose $l_1 \in \arg \max_l V_l$, so $V_{l_1} \geq \sum_l \frac{1}{L} V_l = F(w, \frac{1}{L} \mathbf{1})$, exactly as in Theorem 3’s Case 2. Move $w^{(l_1)}$ along the straight line $(1-t)w^{(l_1)} + t e(\bar{x}^*)$ to the global minimizer’s one-hot weight, holding β (hence $\beta_{l_1} = 1/L$) and every other level fixed throughout; there is no Phase A here, since redistributing β is unavailable. Only the term $\frac{1}{L}g(w^{(l_1)})$ varies, so for every t

$$F(\pi(t)) - F(w, \frac{1}{L} \mathbf{1}) = \frac{1}{L} [g(w^{(l_1)}(t)) - V_{l_1}] \leq \frac{1}{L} C,$$

using $g(w^{(l_1)}(t)) \leq p^* + C$ and $V_{l_1} \geq p^*$ (Eq. (A.6)), so the running maximum over the whole path is at most $F(w, \frac{1}{L} \mathbf{1}) + C/L$. At $t = 1$, $F(\pi(1)) - F(w, \frac{1}{L} \mathbf{1}) = \frac{1}{L}(p^* - V_{l_1}) < 0$, a strict decrease since $V_{l_1} \geq F(w, \frac{1}{L} \mathbf{1}) > p^*$. Hence π is admissible in the (restricted) Definition 1, giving $B_{\text{fix}}(w) \leq C/L$. $\blacksquare$

As $L \rightarrow \infty$ this recovers Theorem 3’s $B = 0$: it is exactly the freedom to shrink $\beta_{l_1} \rightarrow 0$ in Cases 1-2 that pinning β at $1/L$ forecloses, leaving $\delta' = 1/L$ (not driveable $\rightarrow 0$ at fixed L) as the best available discount on the oscillation bound. Algorithm 1 is therefore guaranteed only the weaker C/L barrier of Proposition 4, not the zero barrier of Theorem 3; realizing the stronger guarantee would require implementing a solver that actually moves β , which this paper does not do.

Appendix B. Extended background, related work, pipeline mechanics, further results and contributions

This appendix records additional material compressed in the main text: related approaches, the need for structured initialization, the roles of the three pipeline stages, the binary-constraint derivation, and the mechanics of coarsening and refinement. It is included for completeness and does not change any claims in the main text.

B.1. Existing approaches

MAX-CUT algorithms fall into several broad categories. Exact methods such as branch-and-bound (B&B) and dynamic programming provide certificates but scale exponentially in the worst case. The Biq Mac library (Rendl et al., 2010) uses exact methods to provide optima for the structured instances used as ground truth here. The Lasserre hierarchy (Lasserre, 2001) gives a sequence of increasingly tight semidefinite relaxations, but its moment matrices grow quickly with n . Goemans-Williamson (Goemans and Williamson, 1995) solves an SDP relaxation and rounds by random hyperplanes, giving the classical 0.878 worst-case approximation ratio. Learned solvers form a more recent category: OptGNN (Yau et al., 2024) learns updates in a Burer-Monteiro relaxation, Chen et al. (Chen et al., 2026) learn SDP certificates for branch-and-bound, and Dong et al. (Dong et al., 2025) study streaming algorithms with an oracle-like learned labeling. Our method supplies the structural prior through graph coarsening instead of learning.

B.2. Why structured initialization is necessary

The finite-particle PPR solver is still an iterative nonconvex method. The exact moment formulation gives a useful theoretical target, but it does not ensure that a finite number of particles and restarts will find the right basin. The hard $n = 15$ instance illustrates the issue: random initialization misses the exact optimum in all 21 restarts, while CMG finds it directly (Table 4). This motivates treating initialization as a central part of the solver rather than as an implementation detail.

Table 4: Hard instance $n = 15$, seed = 44, known optimum 38. “CMG” means CMG followed by local-flip refinement.

Strategy	Cut	Ratio	Time
Random-init PPR, best of 21	37	0.974	$21 \times \sim 7s$
Random-init PPR, exact hits	0/21	–	–
CMG+flip	38	1.000	$30 \mu s$
CMG→PPR warm start	38	1.000	$\sim 12s$

B.3. Three complementary tools

The pipeline combines three stages. Spielman-Srivastava sparsification (Spielman and Srivastava, 2011) reduces dense graphs to sparse spectral approximations when appropriate. CMG (Langston et al., 2019; Koutis et al., 2011) builds a hierarchy from the graph’s own edge weights and returns a binary warm start. PPR then uses that warm start inside a continuous particle representation that can make multi-node changes. The composition is

$$\underbrace{G \xrightarrow{\text{S-S}} G'}_{\text{sparsify}} \rightarrow \underbrace{G' \xrightarrow{\text{CMG}} x_{\text{cmg}}}_{\text{warm start}} \rightarrow \underbrace{x_{\text{cmg}} \xrightarrow{\text{PPR}} x'}_{\text{fine-level polish}} \rightarrow \underbrace{x' \xrightarrow{2\text{-opt}} x^*}_{\text{refine}},$$

matching Figure 2. In our experiments, the strongest performance is obtained when the stages are composed in this order; removing individual stages generally reduces solution quality on the hardest instances; however, on graphs that are already sparse, the first stage is typically unnecessary.

B.4. The pipeline end to end

Figure 2 draws the composed architecture: S-S sparsification (a no-op on already-sparse graphs) feeds CMG coarsening to a directly-solvable coarsest level, with the SQP sweet-spot solve substituted in at any level with $n_\ell \leq 20$; the coarsest solution is projected back up through local-flip refinement; PPR then runs once, warm-started from that projection, on the finest graph; and multi-pass restarts followed by a single 2-opt sweep close the gap to the exact optimum. This is the pipeline used for every headline result in Section 3, traced end to end on a real graph in Appendix B.5.11.

B.5. Further Contributions and Additional Results

The main contributions are: (1) applying the particle moment framework to the MAX-CUT binary constraint; (2) showing that CMG and PPR play complementary roles in the compute-constrained regime, with the heavy-tail ablation (Table 13) showing this advantage narrows, and a random start can match or exceed it on about half the instances tested, once a large discrete-tail budget is affordable; (3) demonstrating that PPR can improve beyond CMG’s one-flip local optimum on structured instances; (4) reporting the **t2g20** result of 0.975 mean and 0.984 best cut/OPT; (5) identifying the dense homogeneous regime where the method adds little; (6) evaluating the effect of a stronger discrete tail; and (7) scaling the sparse CSR implementation to $n = 20,000$ with no observed decay in the gain over CMG across the tested size range. We follow below with more details on the problem formulation, pipeline details, and additional results to strengthen and highlight our findings.

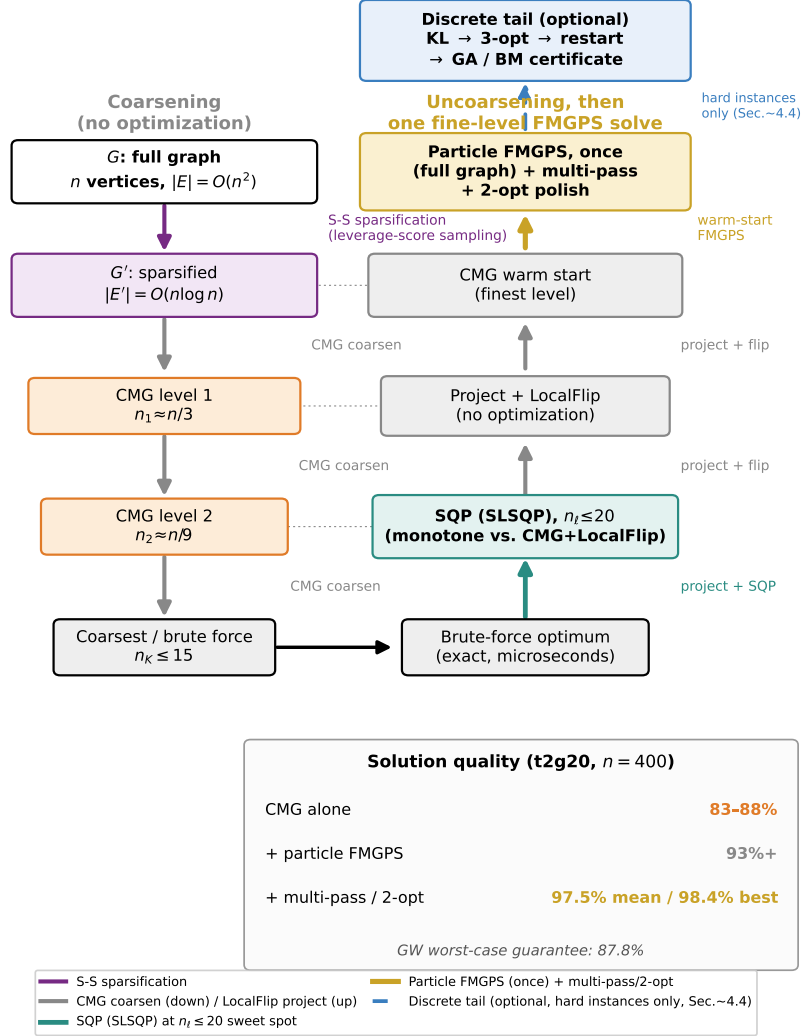


Figure 2: The composed pipeline: S-S sparsification (no-op if already sparse) feeds CMG coarsening to a directly-solvable coarsest level; the coarsest solution is projected back up with local-flip refinement; PPR then runs, warm-started from that projection, on the finest graph, followed by multi-pass restarts and a single closing 2-opt sweep.

B.5.1. THE BINARY-CONSTRAINT DERIVATION

With $x_i \in \{-1, +1\}$, MAX-CUT is

$$\max_x \frac{1}{4} \sum_{(i,j) \in E} w_{ij}(1 - x_i x_j),$$

or equivalently the minimization of its negative over the algebraic variety $\mathcal{V} = \{x \in [-1, 1]^n : x_i^2 = 1 \ \forall i\}$.

B.5.2. PROBLEM DIMENSIONS AND SCALING

With degree-SOS = 2, L levels, and R particles per factor, the particle solver uses $O(LnR)$ parameters. For fixed L and R , this is linear in n , whereas the GW SDP uses $O(n^2)$ variables and Lasserre relaxations grow much faster.

Table 5: Variable-count scaling for PPR-MAX-CUT and standard relaxations.

Component	Count	Notes
Particle locations and weights	$O(LnR)$	linear in n for fixed L, R
GW SDP variables	$O(n^2)$	quadratic in n
Lasserre level k SDP	$O\left(\binom{n+2k}{n}^2\right)$	grows rapidly with n

B.5.3. SPARSIFICATION AND LEVERAGE SCORES

Effective-resistance sparsification samples edges according to leverage scores $\ell_e = w_e R_e$, where $R_e = L_{ii}^\dagger + L_{jj}^\dagger - 2L_{ij}^\dagger$ is the effective resistance of edge $e = (i, j)$. High-leverage edges are structurally important, while low-leverage edges can often be removed while approximately preserving cuts ([Spielman and Srivastava, 2011](#)). Dense uniform graphs have nearly flat leverage scores, so sparsification gives little benefit. Weighted structured graphs can have highly skewed leverage distributions, allowing a much smaller graph to preserve the relevant cut structure.

B.5.4. CMG COARSENING

One CMG coarsening level consists of profiling the graph, splitting long or poorly connected forest paths, isolating weakly attached vertices, and contracting the resulting components. Algorithm 2 gives one level of this procedure.

Solving the coarsest graph directly and projecting the solution back down with local-flip refinement gives the warm start x_{cmg} . Unlike balanced multilevel partitioners such as METIS ([Karypis and Kumar, 1998](#)) and Graclus ([Dhillon et al., 2007](#)), CMG does not force balanced partitions. This is useful for weighted Ising graphs whose cut structure is not balanced.

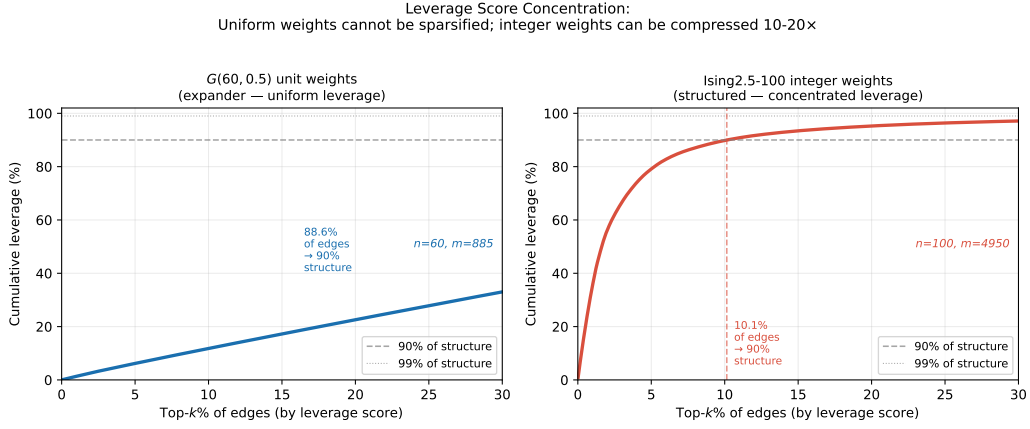


Figure 3: Leverage-score distributions. Uniform-weight graphs have nearly flat leverage scores, while weighted structured graphs can have skewed scores that make sparsification useful.

Algorithm 2: CMG graph coarsening, one level

Input: $G = (V, E, w)$

Output: coarsened G' and cluster map B

GraphProfile: $B[i] \leftarrow \arg \max_{j \sim i} w_{ij}$, $M[i] \leftarrow \max_{j \sim i} w_{ij}$, $D[i] \leftarrow \sum_{j \sim i} w_{ij}$

SplitForest: cut edges in B where path length is too large or conductance is low

UpdateGroups: if $M[i]/D[i] < 1/8$, set $B[i] \leftarrow i$

ForestComponents: label components of B as clusters $\{C_1, \dots, C_{n_c}\}$

Coarsen: $w'_{c_1 c_2} \leftarrow \sum_{(i,j) \in E, i \in C_{c_1}, j \in C_{c_2}} w_{ij}$

return coarsened G' and cluster map B

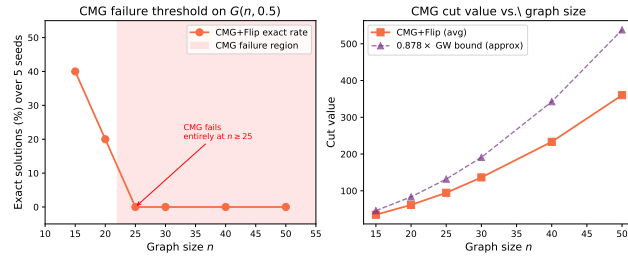


Figure 4: CMG coarsening effectiveness versus graph density. Sparse structured graphs contract aggressively, while dense uniform graphs contract weakly.

B.5.5. REFINEMENT BRANCHES AND MULTI-PASS POLISHING

At very small coarsened levels ($n_\ell \leq 20$), we can solve a rank-2, degree-2-SOS objective with SLSQP (Kraft, 1988) and then apply local-flip refinement. This branch is useful only because it runs on a small coarse graph. The headline Biq Mac and G-set results use the other branch: PPR runs directly on the finest graph, initialized from the CMG projection. Multiple PPR passes are then chained by feeding the best binary solution from one pass into the next, and the final result is polished by a single 2-opt sweep on the original graph.

Table 6: Coarse sweet-spot ($n_\ell \leq 20$) solver comparison, three seeds per size. “Exact” marks a cut matching the known or brute-force optimum.

Graph	Seed	OPT	CMG+Flip	Exact	ADMM	Exact	SQP	Exact
random-15	42	39	37	N	36	N	39	Y
random-15	44	38	38	Y	37	N	38	Y
random-20	43	62	62	Y	61	N	62	Y
random-20	44	70	65	N	65	N	69	N
planted-30-200	42-44	(exact)	exact	Y	exact	Y	exact	Y
Typical wall time			< 1 ms		5-39 s		2-19 s	

B.5.6. ADDITIONAL REFINEMENT AND SCALING RESULTS

Additional experimental results complement the summary statistics reported in Section 3. Table 7 and Figures 5- 7 illustrate how solution quality evolves across refinement stages on **t2g20**, how quickly the multi-pass schedule saturates, and how the sparse implementation scales on large G-set instances.

Table 7: **t2g20** ($n = 400$) configuration ladder, mean cut/OPT over three seeds. The first, second, and fourth rows are the cumulative ladder plotted in Figure 5; the third is the competing *2-opt after every pass* configuration, which branches off pass 1 rather than extending the ladder and finishes 1.8 points behind.

Configuration	Mean	Best seed
CMG baseline	0.832	–
5-pass PPR	0.952	0.968
2-opt after every pass	0.957	0.970
5-pass + single closing 2-opt	0.975	0.984
GW guarantee	0.878	–

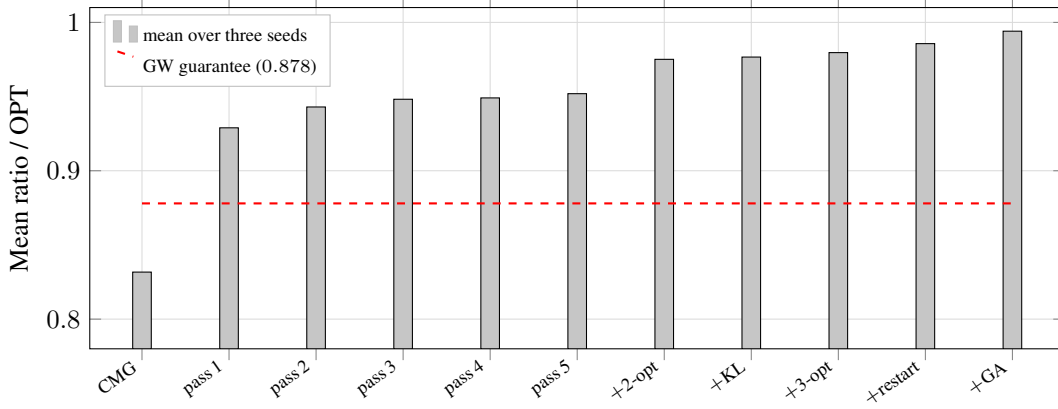


Figure 5: Progressive improvement on $t2g20$ ($n = 400$) across the *cumulative* stage ladder, mean over three seeds. The CMG warm start opens at 0.832; five PPR passes carry it to 0.952; a single closing 2-opt sweep gives the 0.975 headline of Table 7; and the discrete tail (KL, 3-opt, 200-restart, memetic GA) closes to 0.994. Every stage shown here is applied on top of the previous one. The competing *2-opt after every pass* configuration is deliberately absent: it branches off pass 1 rather than extending the ladder, and is reported in Table 7 above and per-instance in Table 10.

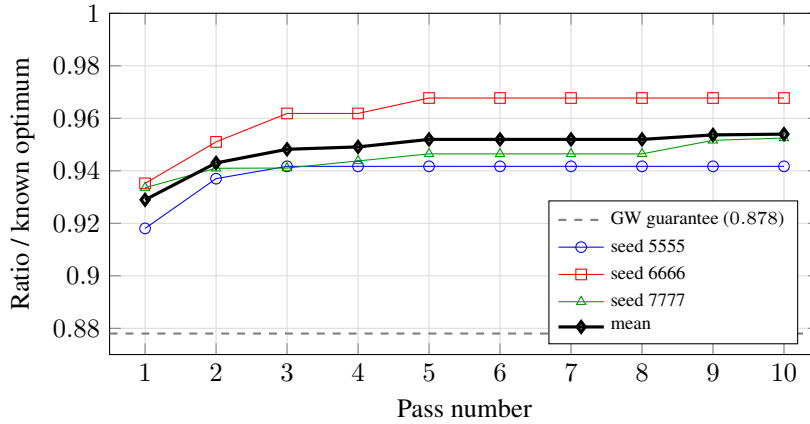
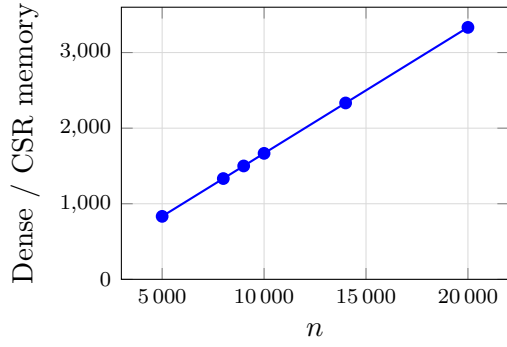
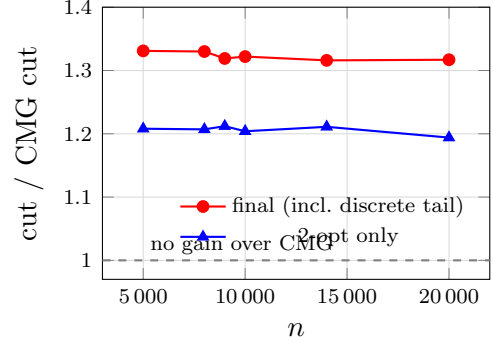


Figure 6: Ratio to known optimum versus pass number on $t2g20$ ($n = 400$), $L=128$, $R=32$, $k=40$. Pass 2 yields the largest single gain (+0.014); the curve is essentially flat from pass 5 onward. Read against Table 9, where *quadrupling* the restart count buys only +0.004 in total: it is the multi-pass schedule, not raw restart breadth, that escapes the warm-start basin.



(a) Dense/CSR memory ratio; grows linearly in n .



(b) Final/CMG and 2-opt-only/CMG; both flat in n .

Figure 7: Sparse scaling on six G-set instances, $n = 5000$ -20,000 (a superset of the four shown in Table 3, which omits G65 and G66). Memory savings grow linearly with n , exceeding $3300\times$ at $n = 20,000$. The right-hand panel shows the gain over the CMG warm start at both pipeline stages: the continuous-relaxation path alone (CMG $\rightarrow$ PPR $\rightarrow$ 2-opt, blue) holds a tight 1.194 - $1.212\times$ band, and appending the discrete tail (KL/3-opt/GA/ILS, red) carries this to 1.316 - $1.331\times$; neither band decays at the largest instance. Both panels are plotted against a common n axis; the right-hand panel's vertical axis is anchored at 1.0 , the no-gain line, so that the flatness of the bands is not exaggerated by a magnified scale.

B.5.7. PASS AND RESTART SATURATION

Tables 8 and 9 below provide the evidence that the quality of the warm-start basin, not the breadth of random restarts, is what limits the pipeline on **t2g20**. Table 8 sweeps the pass count at fixed restart budget: pass 2 delivers the largest single gain (+0.014), the curve then flattens, and five passes capture essentially all achievable gain (0.952), with only a small late bump at pass 9. Table 9 holds the pass count at one and instead *quadruples* the restart count: going $k=40 \rightarrow 80 \rightarrow 160$ buys +0.003 then +0.001, so a single additional pass is worth roughly five times a doubling of k .

Table 8: Pass-count saturation on **t2g20** ($n = 400$). $L=128$, $R=32$, $k=40$, EXTENDED penalty, $s=0.5$. Δ is the incremental mean gain over the previous pass.

Pass	seed 5555	seed 6666	seed 7777	Mean	Δ
1	0.9180	0.9352	0.9336	0.9289	–
2	0.9370	0.9510	0.9410	0.9430	+0.0140
3	0.9417	0.9618	0.9410	0.9482	+0.0052
4	0.9417	0.9618	0.9438	0.9491	+0.0009
5	0.9417	0.9677	0.9464	0.9520	+0.0029
6	0.9417	0.9677	0.9464	0.9520	+0.0000
7	0.9417	0.9677	0.9464	0.9520	+0.0000
8	0.9417	0.9677	0.9464	0.9520	+0.0000
9	0.9417	0.9677	0.9516	0.9537	+0.0017
10	0.9417	0.9677	0.9525	0.9540	+0.0003

Table 9: Single-pass high- k restart sweep on **t2g20**. $L=128$, $R=32$, EXTENDED penalty, $s=0.5$. Quadrupling k gains less than one extra pass does in Table 8.

k	seed 5555	seed 6666	seed 7777	Mean	Δ
40	0.9180	0.9352	0.9336	0.9289	–
80	0.9175	0.9475	0.9309	0.9320	+0.003
160	0.9249	0.9424	0.9323	0.9332	+0.001

 B.5.8. THE FULL **t2g20** STAGE LADDER AND 2-OPT ORDERING

Table 10 isolates the 2-opt sweep itself. Applied once after five passes (variant A) it raises the mean from 0.952 to **0.975** on only 12-19 accepted flips, and the best single seed reaches **0.984** of the exact optimum. Interleaving it after every pass (variant B) instead finds *zero* improving flips: the solution is already 2-opt-optimal from pass 1 onward, locked into a basin later passes cannot escape, and it finishes at 0.957. Variant C shows how much a single 2-opt sweep is worth: 1-pass + 2-opt (0.962) outperforms 5-pass alone (0.952), so one 2-bit sweep recovers more than four extra particle passes. Table 11 orders every configuration by mean ratio, and Table 12 extends the winning one with the discrete tail.

Table 10: 2-opt polish on **t2g20** ($n = 400$). “Before”/“After” are ratio/OPT around the 2-opt sweep; “Flips” counts accepted 2-opt moves.

Experiment	Seed	Before	After	Δ	Flips
A: 5-pass + 2-opt	5555	0.9417	0.9664	+0.025	12
	6666	0.9677	0.9843	+0.017	19
	7777	0.9464	0.9746	+0.028	18
	mean	0.9520	0.9751	+0.023	16
B: 2-opt each pass	5555	0.9548	0.9548	0.000	0
	6666	0.9700	0.9700	0.000	0
	7777	0.9461	0.9461	0.000	0
	mean	0.9570	0.9570	0.000	0
C: 1-pass + 2-opt	5555	0.9217	0.9637	+0.042	23
	6666	0.9498	0.9653	+0.015	13
	7777	0.9276	0.9574	+0.030	24
	mean	0.9330	0.9621	+0.029	20
GW guarantee			0.878		

Table 11: All **t2g20** ($n = 400$) configurations, ordered by mean ratio/OPT. “1-pass only” is the single-pass configuration with 2-opt disabled; Table 8’s pass-1 row (0.9289) is the same nominal configuration measured in a separate run and differs by 0.0041; this is a run-to-run variation, not a discrepancy in the underlying method.

Configuration	Mean	Best seed
CMG baseline	0.832	–
1-pass only	0.933	0.950
5-pass only	0.952	0.968
B: 2-opt after each pass	0.957	0.970
C: 1-pass + 2-opt	0.962	0.965
A: 5-pass + 2-opt	0.975	0.984
GW guarantee	0.878	–

Table 12: Discrete tail appended to the Experiment A (5-pass PPR + 2-opt) partition on **t2g20**, all three seeds, FULL config ($L=128, R=32, k=40$, RHO_EXTENDED, $s=0.5$). The +2-opt row reproduces Table 10’s Experiment A exactly (bit-identical RNG stream and draw order), confirming the tail is layered on the genuine headline partition, not an approximation of it. **ga_only** is the uncontaminated baseline (independent RNG, no CMG or PPR) at the same GA budget; note that it *exceeds* the full pipeline here, the same finding the heavy-set ablation of Table 13 reports.

Stage	5555	6666	7777	Mean	vs. headline
CMG	0.7915	0.8630	0.8406	0.8317	–
5-pass PPR	0.9417	0.9677	0.9464	0.9520	=Table 11
+2-opt (headline)	0.9664	0.9843	0.9746	0.9751	=Table 10
+KL	0.9710	0.9843	0.9746	0.9766	+0.0015
+3-opt	0.9723	0.9888	0.9778	0.9796	+0.0046
+restart (200)	0.9846	0.9888	0.9836	0.9857	+0.0106
+GA	0.9876	0.9959	0.9988	0.9941	+0.0190
ga_only (uncontaminated)	0.9971	0.9973	0.9986	0.9977	–

B.5.9. TAIL ABLATION ON THE HEAVY SET, AND WHERE THE **G81** BUDGET GOES

This block supports the open question: “Once a discrete tail is budgeted, does the warm start still decide the answer?” Table 13 ablates the tail stage by stage across eight instances, and Figure 8 plots it against the **ga_only** control: a pure-random-seeded GA at the same budget, with no CMG or PPR. On three of eight instances the control *beats* the full pipeline and on a fourth it ties, which is the isolation finding the body cites. Table 13 reports absolute cuts, so nothing in it depends on an external reference value. Figure 8 normalizes the same measurements by a best-known cut drawn from the literature. Table 15 adds an *empirical* SDP-relaxation estimate (not a certified upper bound) for the same eight instances.

Table 14 asks where the wall clock actually goes on the largest instance. The marginal efficiency column spans *five orders of magnitude*: the first five stages consume 75.7s, or 1.53% of the original budget, and deliver most of the total gain, while the 200-restart stage spends 2782s to buy 30 cut units: 0.011 cut/s, the least productive stage in the study by three orders of magnitude. Removing it and retuning the remaining stages is what reduces the budget from 4951s to 261s while *improving* the cut.

Finally, every stage above chases a bigger *cut*, and none of them says how close that cut is to optimal. Table 15 closes the ablation with a stage that answers the other question: solve the rank- p SDP relaxation $\max \frac{1}{2} \sum w_{ij}(1 - \langle v_i, v_j \rangle)$, $\|v_i\| = 1$, by the Mixing Method (Wang et al., 2017) (coordinate update $v_i \leftarrow -g_i/\|g_i\|$), round via 200 GW random hyperplanes, and locally optimize each rounding. At $p = \lceil \sqrt{2n} \rceil$ (Boumal et al., 2016) the rank-restricted relaxation is *generically* free of spurious local minima, but the Mixing Method’s convergence check does not itself verify that condition, so the converged value is an *empirical estimate*, not a certified upper bound on the true MAX-CUT value (Table 15 and Appendix B.5.11 say more). The resulting gaps, 10-12% on the sparse-signed instances and 4-6% on the dense contrast pair, are orthogonal to the cut-improvement stages: Burer-Monteiro never beats the GA on cut value, and is not intended to.

Table 13: Discrete-tail ablation across the heavy set (single core, one shared seed so every row is directly comparable). The base of each row is the CMG warm start $\rightarrow$ light multi-pass PPR ($L=32, R=8, k=20$, five passes) $\rightarrow$ 2-opt; subsequent columns append KL, then 200-restart local search, then the memetic GA, reporting the cut after each. 3-opt runs between KL and the restart stage but is omitted as a column: it finds *zero* improving moves on seven of the eight instances, and +2 units on G22. **ga_only** is the control: the same memetic GA at the same budget from a pure random seed, with no CMG and no PPR. All cuts are absolute, so the pipeline-versus-control comparison needs no external reference value; **bold** marks the larger of +GA and **ga_only** on each row. t_F and t_{tail} are the PPR and total-tail wall-clock in seconds. The top block is the sparse-signed toroidal target regime; the bottom block is dense uniform-weight contrast.

Instance	n	$ E $	CMG	PPR+2-opt	+KL	+rst	+GA	ga_only	t_F/t_{tail}
<i>Sparse-signed toroidal (target regime)</i>									
G11	800	1600	418	532	548	560	560	562	133/25
G12	800	1600	414	528	546	550	554	556	163/25
G13	800	1600	416	546	562	578	580	580	165/26
G32	2000	4000	1070	1306	1356	1382	1402	1396	550/128
G33	2000	4000	1012	1298	1328	1358	1368	1362	580/130
G34	2000	4000	1016	1300	1344	1366	1372	1376	631/132
<i>Dense uniform-weight (contrast)</i>									
G1	800	19176	11300	11541	11541	11571	11621	11617	235/513
G22	2000	19990	12806	13113	13126	13228	13355	13332	1068/305

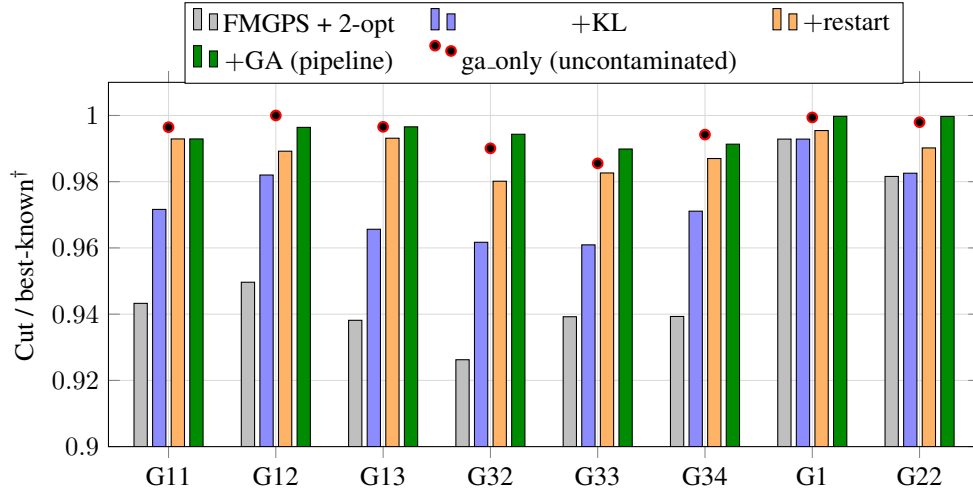


Figure 8: Cut/best-known[†] after each tail stage (Table 13), with the uncontaminated **ga_only** baseline overlaid as red points. Where the red point sits *above* the green +GA bar (G11, G12, G34; equal on G13), a pure-random-seeded GA with no CMG or PPR warm start reaches a better final cut than the full pipeline; the visual form of the isolation finding: once the discrete tail is budgeted, the particle solver does not uniformly improve the reported answer. [†]The denominator here is an external best-known reference, not a proven optimum; its provenance is flagged in the text preceding Table 13. It cancels from the red-versus-green comparison this figure is drawn to make, since both are our own measurements on the same instance.

Table 14: Per-stage anatomy of **G81** ($n = 20,000$, $|E| = 40,000$, optimum 14060 (Zick et al., 2026)) before and after retuning. “ Δcut ” is the gain over the preceding stage; “ t ” is that stage’s own wall time (obtained by differencing the harness’s cumulative timestamps); “ cut/s ” is $\Delta\text{cut}/t$, the stage’s marginal efficiency. The original column is an earlier, unretuned configuration: dense all-pairs 2-opt, linear-scan Kernighan-Lin, 200 random restarts, memetic GA at 150 generations. The retuned column, CSR-adjacent 2-opt, heap-based KL, no restart stage, GA at 800 generations, and a perturbation-based iterated local search, is the configuration behind Table 3; “–” marks a stage that configuration does not run, and the retuned 2-opt’s efficiency is omitted because its wall time is at timer resolution.

Stage	Original (4951 s)			Retuned (261.2 s)			% opt
	cut	t (s)	cut/s	cut	t (s)	cut/s	
CMG warm start	10574	0.004	–	10574	0.007	–	75.21%
Light PPR (3 passes)	12466	64.37	29.4	12466	64.07	29.5	88.66%
2-opt sweep	12628	4.47	36.3	12626	0.000	–	89.80%
Kernighan-Lin	13674	6.19	169.0	13674	0.08	13789.5	97.25%
3-opt + KL/3-opt loop	13676	0.70	2.9	13676	0.04	48.8	97.27%
Multi-restart (200)	13706	2781.86	0.011	–	–	–	–
Memetic GA	13876	1880.56	0.090	13912	57.50	4.10	98.95%
Iterated local search	–	–	–	13922	136.11	0.073	99.02%
Burer-Monteiro round	13876	213.10	0.000	13922	3.39	0.000	99.02%

Table 15: Burer-Monteiro relax-and-reround estimate (rank $p = \lceil \sqrt{2n} \rceil$, 200 GW hyperplanes, single core). “bound” is the converged value of the Mixing Method’s coordinate updates on the (nonconvex, sphere-constrained) SDP relaxation; since the Mixing Method is not guaranteed to reach the relaxation’s global optimum, this is an *empirical estimate*, not a certified upper bound on the true MAX-CUT value (Appendix B.5.11’s Karate Club walkthrough reports a small negative gap, which a genuine upper bound cannot produce). gap_{BM} is the empirical gap between that estimate and Burer-Monteiro’s own rounded cut; gap_{GA} is the empirical gap between the estimate and the pipeline’s best reported cut (the +GA column of Table 13), which is always at least as tight since the GA cut is $\geq$ BM’s cut on every instance. Both gap columns are measured against an estimate this pipeline computes itself, so like Table 13 they involve no external reference value.

Instance	bound	BM cut	GA cut	gap_{BM} (%)	gap_{GA} (%)	t_{BM} (s)
G11	627.7	558	560	11.10	10.78	1.9
G12	622.3	554	554	10.97	10.97	1.9
G13	645.1	578	580	10.40	10.09	1.9
G32	1563.1	1386	1402	11.33	10.31	12.0
G33	1539.5	1360	1368	11.66	11.14	12.1
G34	1541.7	1366	1372	11.39	11.00	11.9
G1	12077.6	11591	11621	4.03	3.78	108.8
G22	14125.8	13283	13355	5.97	5.46	83.1

B.5.10. BIQ MAC FAMILIES: CUT QUALITY AGAINST WALL TIME

Figure 9 is the visual form of body Table 2.

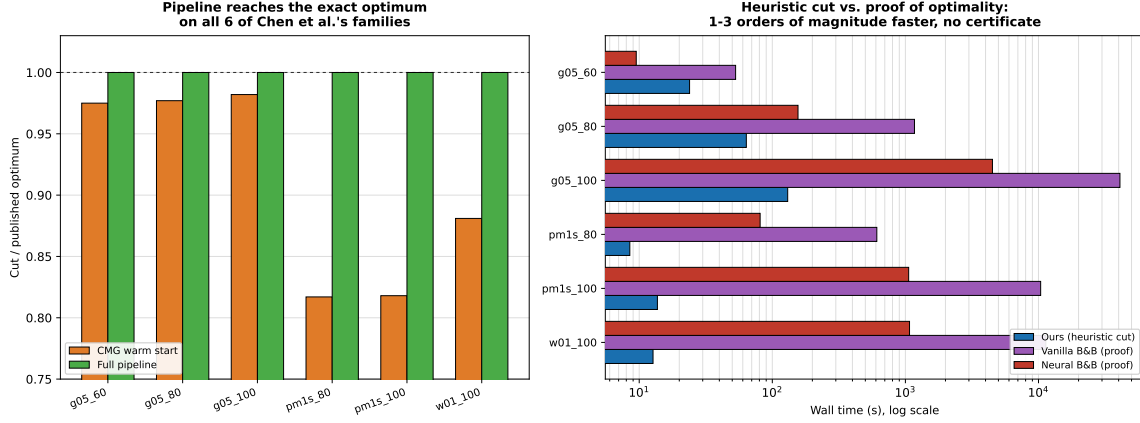


Figure 9: The six Biq Mac families of Chen et al. (Chen et al., 2026), ten instances each. **Left:** cut divided by published optimum, CMG warm start versus the full pipeline; every family reaches 1.000. **Right:** wall time on a log axis, ours against their vanilla and neural branch-and-bound. The two axes measure different deliverables: our bar is a heuristic cut that matches a published optimum, theirs is a *proof* of optimality. The time comparison is therefore not a like-for-like ranking of solvers, and is shown because it is the comparison the learned-solver literature sets up.

B.5.11. PIPELINE WALKTHROUGH EXAMPLE

Figure 10 shows the pipeline on Zachary’s Karate Club (Zachary, 1977). The example is illustrative rather than a benchmark: it uses degree-weighted edges to expose structure and does not report a ratio to a published optimum. It shows the mechanical role of each stage: sparsify, coarsen, project a warm start, and run PPR on the finest graph.

On this small instance, the CMG warm start already reaches the best cut found by the later stages. A Burer-Monteiro (Burer and Monteiro, 2003) relax-and-reround pass gives a converged SDP relaxation *estimate* of approximately 2587.18, matching the best rounded cut to a gap of about -7.7×10^{-6} , a negative gap, which demonstrates directly that this estimate is not a certified upper bound (a genuine upper bound cannot fall below a feasible cut); see Table 15. This example is therefore useful mainly as a transparent mechanical walkthrough, not as evidence that every stage adds value on every graph.

B.5.12. THE CONTINUOUS-PHASE OPTIMIZER SWAP IN FULL

Table 16 gives the per-instance breakdown for the optimizer swap summarized in Section 3. L-BFGS, Adam, GD, and a level-stochastic variant use the same CMG warm start, restart budget, continuation schedule, and downstream tail.

B.5.13. INTERNAL OPTIMALITY CERTIFICATE FOR G81

The tabu-diversified (Glover, 1986; Glover and Laguna, 1997) tail reaches a cut of 13962 on G81, which is 99.3% of the externally reported value 14060 (Zick et al., 2026). As an

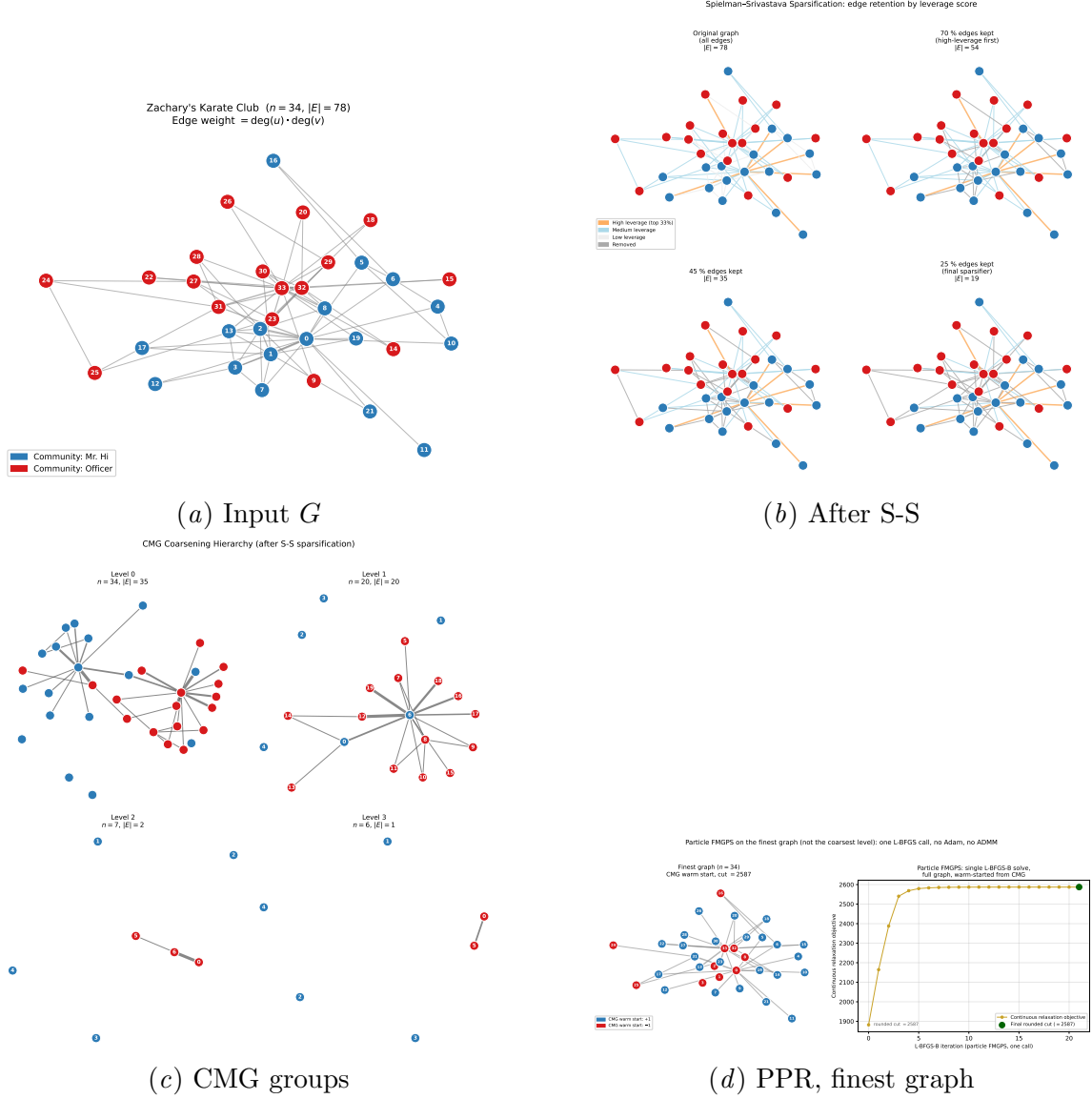


Figure 10: Pipeline walkthrough on Zachary's Karate Club. The four panels show the input graph, sparsified graph, CMG grouping, and a PPR solve on the finest graph initialized from the CMG projection.

Table 16: Continuous-phase optimizer swap: light PPR solved by L-BFGS (production), Adam, GD, and level-stochastic SGD, each feeding the identical tail (2-opt→KL→3-opt→200-restart→memetic GA), shared CMG warm start, independently seeded per arm. Columns are cut after the continuous phase alone (pre-tail) and the final cut (post-tail), as a ratio to L-BFGS’s own cut on that instance; last column is GD’s continuous-phase wall clock relative to L-BFGS’s. Two representative rows per regime shown; the max spread row summarizes all eight.

Instance	n	Pre-tail (/L-BFGS)			Post-tail (/L-BFGS)			$t_{\text{GD}}/t_{\text{LB}}$
		Adam	GD	SGD	Adam	GD	SGD	
<i>Sparse-signed toroidal</i>								
G33	2000	0.938	0.944	1.000	1.003	1.001	1.000	27.5×
G34	2000	0.946	0.942	0.995	0.997	1.000	1.000	29.0×
<i>Dense uniform-weight</i>								
G1	800	0.987	0.979	0.979	1.000	1.000	1.000	11.8×
G22	2000	0.985	0.981	0.983	0.997	0.997	0.999	15.1×
max spread (8 instances)		up to 7.4%			$\leq 0.7\%$			11–30×

internal check, we solved mixed-integer repair problems with HiGHS (Huangfu and Hall, 2018) on every connected residual core of size up to 200 vertices, covering 40 cores in total. No improving repair was found. This certifies the reported cut on those residual cores, but it should not be read as an independent proof that the global G81 optimum is 14060.

Appendix C. Pseudocode for every pipeline component

Appendix B.4 names the components of the pipeline and shows how they compose. This appendix gives pseudocode for each component in isolation, transcribed from the reference implementation that produced every numerical result reported here, so that the pseudocode and the measured behaviour agree exactly. Two components already have boxes and are not repeated: the continuous solver itself is Algorithm 1 (Appendix A), and CMG coarsening is Algorithm 2 (Appendix B.5.4).

C.1. Stage 1: Spielman-Srivastava sparsification

Algorithm 3 realizes the Spielman-Srivastava leverage-score sampler (Spielman and Srivastava, 2011) of Appendix B.5.3. For $n \leq 500$ it computes exact effective resistances via the Laplacian pseudoinverse; for larger n it falls back to the degree-proxy approximation $R_{ij} \approx 1/d_i + 1/d_j$. It is invoked only when $|E| > 3n$; on the already-sparse toroidal instances that dominate our benchmarks it is a no-op.

Algorithm 3: Spielman-Srivastava effective-resistance sparsification

Input: $G = (V, E, w)$; target edge count t

Output: sparsified $G' = (V, E', w')$

Procedure SSSparsify(G, t):

```

    if  $|E| \leq t$  then return  $G$                                 // no-op

    if  $n \leq 500$  then
        build Laplacian  $L$  from  $|w|$ ; eigendecompose  $L = Q\Lambda Q^\top$ 
         $R_{ij} \leftarrow \sum_{k:\lambda_k > \text{tol}} (Q_{ik} - Q_{jk})^2 / \lambda_k$  for every edge  $(i, j)$     // exact resistances
    else
         $d_i \leftarrow \sum_{j \sim i} |w_{ij}|$  for every vertex
         $R_{ij} \leftarrow 1/d_i + 1/d_j$  for every edge  $(i, j)$     // degree-proxy fallback
    end
     $\ell_e \leftarrow |w_e| R_e$  for every edge  $e$     // leverage score
    foreach edge  $e = (i, j)$  do
         $p_e \leftarrow \min(1, \ell_e t / \sum_{e'} \ell_{e'})$ 
        with probability  $p_e$ : keep  $e$  with rescaled weight  $\text{sign}(w_e) |w_e| / p_e$ 
    end
    return sparsified  $G' = (V, E', w')$ 

```

C.2. Stage 3: coarse sweet-spot SQP solve

Algorithm 4 is the Burer-Monteiro (Burer and Monteiro, 2003) moment-matrix solve applied at the bottom of the CMG hierarchy, and is used only at a level with $n_\ell \leq 20$ vertices, where a small dense SLSQP (Kraft, 1988) solve is affordable. Note that here the binary constraint enters as an explicit equality on the moments of Eq. (A.2) ($\mu^{(4)} - 2\mu^{(2)} + \mu^{(0)} = 0$, i.e. $\mathbb{E}[(x^2 - 1)^2] = 0$, which forces each marginal onto $\{\pm 1\}$), rather than through the penalty schedule the larger levels use. Throughout, $\mu_i^{(k),l}$ denotes the order- k moment entry read off the Gram matrix $R_i^{(l)\top} R_i^{(l)}$ of the rank-restricted factor.

C.3. Stage 4: one particle pass, multi-pass, and 2-opt

Algorithm 5 is Algorithm 1 restricted to a fixed L, R, k and packaged for repeated warm-started invocation; it is what Algorithm 6 chains across passes. We keep it as a separate box

Algorithm 4: Coarse sweet-spot SQP solve ($n_\ell \leq 20$)

Input: coarse graph G_ℓ ; CMG projection x_0
Output: best ± 1 partition on G_ℓ
Procedure SQPCoarseSolve(G_ℓ, x_0):

```

    best  $\leftarrow -\infty$ 
    for  $x_{\text{init}} \in \{x_0, \text{one random } \pm 1 \text{ draw}\}$  do
        initialize BM factors  $R_i^{(l)} \in \mathbb{R}^{\text{rank} \times 3}$  at the point mass  $x_{\text{init}}$ 
        solve via SLSQP:  $\min -\frac{1}{4} \sum_{(i,j) \in E} w_{ij} (1 - \sum_l \mu_i^{(1),l} \mu_j^{(1),l})$ 
        s.t.  $\sum_l \mu_i^{(0),l} = 1, \sum_l (\mu_i^{(4),l} - 2\mu_i^{(2),l} + \mu_i^{(0),l}) = 0 \quad \forall i$ 
         $\hat{x}_i \leftarrow \text{sign}(\sum_l \mu_i^{(1),l})$  for all  $i$ 
        LocalFlip( $\hat{x}, G_\ell$ )
        keep  $\hat{x}$  if  $\text{Cut}(\hat{x}) > \text{best}$ 
    end
    return the best  $\hat{x}$  found across the two restarts

```

because the multi-pass driver of Appendix B.5.5 calls it directly rather than through the full solver. Algorithm 7 is the 2-opt (Croes, 1958) closing polish whose *ordering* matters: applied once at the end it is worth +0.023 in mean ratio, while applying it after every pass is worse (Appendix B.5.8).

Algorithm 5: One particle pass

Input: J ; warm start x_{warm} ; L, R, k ; sat; $\rho_1 < \dots < \rho_S$
Output: best cut found and the corresponding partition $\hat{x}$
Procedure OnePass($J, x_{\text{warm}}, L, R, k, \text{sat}, \rho_1 < \dots < \rho_S$):

```

    (identical body to Algorithm 1's restart loop, warm-started from  $x_{\text{warm}}$ )
    return best cut found, and the corresponding partition  $\hat{x}$ 

```

Algorithm 6: Multi-pass restarts

Input: J ; initial partition x_0 ; edges; weights; number of passes

Output: best cut and its partition

Procedure MultiPass($J, x_0, \text{edges}, \text{weights}, \text{passes}$):

```

     $x_{\text{warm}} \leftarrow x_0$ ; best  $\leftarrow -\infty$ 
    for  $p = 1, \dots, \text{passes}$  do
         $(c, \hat{x}) \leftarrow \text{OnePass}(J, x_{\text{warm}}, \dots)$ 
        if  $c > \text{best}$  then
            | best  $\leftarrow c$ ;  $x_{\text{warm}} \leftarrow \hat{x}$  // chain the best binary forward
        end
    end
    return best,  $x_{\text{warm}}$ 

```

C.4. Stage 5 (optional discrete tail): Kernighan-Lin and 3-opt

Algorithms 8 (Kernighan-Lin (Kernighan and Lin, 1970)) and 9 (3-opt (Lin and Kernighan, 1973)) are the first two tail stages ablated in Appendix B.5.9; both are monotone (neither

Algorithm 7: 2-opt sweep

Input: partition x ; $G = (V, E, w)$
Output: number of accepted double-flips

Procedure TwoOptSweep(x, G):

```

 $g_i \leftarrow \sum_{j \sim i} w_{ij} x_j$  for all  $i$ 
repeat
    improved  $\leftarrow$  false
    for  $i = 1, \dots, n$  do
        for  $j = i + 1, \dots, n$  do
             $\Delta \leftarrow 2x_i g_i + 2x_j g_j - 4w_{ij} x_i x_j$ 
            if  $\Delta > 10^{-10}$  then
                update  $g$  for neighbors of  $i$  and of  $j$ ; flip  $x_i \leftarrow -x_i, x_j \leftarrow -x_j$ 
                improved  $\leftarrow$  true
            end
        end
    end
until not improved
return number of accepted double-flips
    
```

ever decreases the cut) and neither is part of the headline light pipeline. KL is the stage that does the bulk of the tail's work, which is why the endpoint is set by the search operator rather than by the starting partition.

Algorithm 8: Kernighan-Lin variable-depth exchange

Input: partition x ; G
Output: total cut improvement

Procedure KLRefine(x, G):

```

total  $\leftarrow$  0
repeat
     $g_i \leftarrow \sum_{j \sim i} w_{ij} x_j$  for all  $i$ ; unlock all vertices
    cum  $\leftarrow$  0; best_cum  $\leftarrow$  0; best_k  $\leftarrow$  0; moves  $\leftarrow$  []
    for  $step = 1, \dots, n$  do
         $v \leftarrow \arg \max_{v \text{ unlocked}} x_v g_v$ ; break if none unlocked
        cum  $\leftarrow$  cum +  $x_v g_v$ ; update  $g$  for neighbors of  $v$  under  $x_v \rightarrow -x_v$ 
        flip  $x_v \leftarrow -x_v$ ; lock  $v$ ; append  $v$  to moves
        if cum > best_cum then
            best_cum  $\leftarrow$  cum; best_k  $\leftarrow$  |moves|
        end
    end
    undo every move in moves after index best_k // keep only the best prefix
    if best_cum >  $10^{-9}$  then total  $\leftarrow$  total + best_cum
until best_cum  $\leq 10^{-9}$ 
return total
    
```

Algorithm 9: Connected-triple 3-opt

Input: partition x ; G
Output: total cut improvement

Procedure ThreeOptSweep(x, G):

```

     $g_i \leftarrow \sum_{j \sim i} w_{ij} x_j$  for all  $i$ ; total  $\leftarrow 0$ 
    repeat
        improved  $\leftarrow$  false
        foreach edge  $(a, b)$ ,  $a < b$  do
             $\Delta_{ab} \leftarrow x_a g_a + x_b g_b - 2w_{ab} x_a x_b$  // pair-flip  $\Delta$ Cut
            if  $\Delta_{ab} > 10^{-9}$  then
                flip  $\{a, b\}$ , update  $g$ ; total  $\leftarrow$  total +  $\Delta_{ab}$ ; improved  $\leftarrow$  true; continue
            end
            foreach  $c$  adjacent to  $a$  or  $b$ ,  $c \notin \{a, b\}$  do
                 $S \leftarrow \{a, b, c\}$ ;  $\Delta_S \leftarrow \sum_{v \in S} x_v g_v - 2 \sum_{(u,v) \in E \cap S} w_{uv} x_u x_v$ 
                if  $\Delta_S > 10^{-9}$  then
                    flip  $S$ , update  $g$ ; total  $\leftarrow$  total +  $\Delta_S$ ; improved  $\leftarrow$  true; break
                end
            end
        end
    until not improved
    return total

```

C.5. Stage 5 (optional discrete tail): local optimum, memetic GA, and Burer-Monteiro

Algorithm 10 drives a partition to a joint 2-opt (Croes, 1958)/Kernighan-Lin (Kernighan and Lin, 1970)/3-opt (Lin and Kernighan, 1973) local optimum; it is the polish step used both by the multi-restart diagnostic and, inside Algorithm 11, to evaluate every population member and every child. Algorithm 11 is the memetic genetic algorithm (Moscato and Cotta, 2003), using the Chu-Beasley (Chu and Beasley, 1998) steady-state replacement scheme, whose `ga_only` control appears in Appendix B.5.9, and Algorithm 12 is the Burer-Monteiro (Burer and Monteiro, 2003) relax-and-reround pass, computed via the Mixing Method (Wang et al., 2017), that produces the empirical SDP-relaxation estimate of Table 15. Its default rank $\lceil \sqrt{2n} \rceil$ is the threshold above which the rank-restricted problem is *generically* free of spurious local minima (Boumal et al., 2016), but the Mixing Method’s convergence check (a tolerance on successive objective values) does not itself verify that condition, so the resulting estimate is not a certified bound; Appendix B.5.11’s Karate Club walkthrough gives a direct, concrete negative-gap counterexample.

Algorithm 10: Joint 2-opt/KL/3-opt local optimum

Input: partition x ; G

Output: x at a joint 2-opt/KL/3-opt local optimum (modified in place)

Procedure LocalOptimize(x, G):

$$\text{TwoOptSweep}(x, G)$$
for it = 1, ..., 50 **do**
$$a \leftarrow \text{KLRefine}(x, G); b \leftarrow \text{ThreeOptSweep}(x, G)$$
if $a + b < 10^{-9}$ then break

end

Algorithm 11: Memetic GA (Chu-Beasley steady-state, agreement crossover)

Input: G ; population size pop; generations gens

Output: fittest population member and its cut

Procedure MemeticGA(G , pop, gens):

for $i = 1, \dots, \text{pop}$ **do**
$$P_i \leftarrow \text{random } \pm 1; \text{LocalOptimize}(P_i, G); \text{fit}_i \leftarrow \text{Cut}(P_i)$$

end

for gen = 1, ..., gens **do**

$p_1, p_2 \leftarrow$ two distinct binary-tournament winners from P

align P_{p_2} to P_{p_1} by global sign (maximize agreement)

child_{*i*} $\leftarrow P_{p_1, i}$ where they agree; a random ± 1 coin flip where they disagree

$$\text{LocalOptimize}(\text{child}, G); c_f \leftarrow \text{Cut}(\text{child})$$
if no population member has $|\text{fit} - c_f| < 0.5$ then

```
// reject duplicate-cut children
```

$$w \leftarrow \arg \min_i \text{fit}_i$$
if $c_f > \text{fit}_w$ **then**
$$P_w \leftarrow \text{child}; \text{fit}_w \leftarrow c_f$$

end

end

end

return $\arg \max_i \text{fit}_i$ and its cut

```
// steady-state replace-worst
```

Algorithm 12: Burer-Monteiro relax-and-reround (Mixing Method)

Input: G ; rank p ; number of hyperplanes h

Output: (bound, best), whose difference is an empirical, not certified, optimality gap

Procedure BurerMonteiro(G, p, h):

```

    if  $p \leq 0$  then  $p \leftarrow \lceil \sqrt{2n} \rceil$  // default rank

     $V_i \leftarrow$  random unit vector in  $\mathbb{R}^p$  for all  $i$ 
    repeat
        for  $i = 1, \dots, n$  do
             $g_i \leftarrow \sum_{j \sim i} w_{ij} V_j$ ;  $V_i \leftarrow -g_i / \|g_i\|$  // anti-align (coordinate update)
        end
         $\text{obj} \leftarrow \frac{1}{2} (W - \sum_{(i,j) \in E} w_{ij} \langle V_i, V_j \rangle)$ 
    until  $|\text{obj}_{\text{prev}} - \text{obj}| < \text{tol} \cdot (1 + |\text{obj}|)$ 
    bound  $\leftarrow$  obj; best  $\leftarrow -\infty$ 
    for  $h' = 1, \dots, h$  do
        draw random hyperplane  $r \sim \mathcal{N}(0, I_p)$ ;  $x_i \leftarrow \text{sign}(\langle V_i, r \rangle)$  for all  $i$ 
        LocalOptimize( $x, G$ ); keep  $x$  if  $\text{Cut}(x) > \text{best}$ 
    end
    return (bound, best) // bound – best is an empirical, not certified, gap

```
