Test Case	EIG1		Paraboli		MELO		Runtimes(s)		improv % vs EIG1	improv % vs Paraboli
	cuts	RC	cuts	RC	cuts	RC	$d = 2$	$d = 10$		
19ks	179	8.92			119	5.89	40	79	+34.0	
bm1	75	38.9			48	30.0	4	9	+22.9	
prim1	75	43.6	53	30.6	64	36.9	3	8	+15.4	-17.1
prim2	254	11.3	146	6.47	169	7.48	26	89	+33.9	-13.6
test02	196	28.4			106	15.4	9	29	+46.6	
test03	85	13.2			60	9.29	9	27	+29.7	
test04	207	36.2			61	10.6	8	24	+70.8	
test05	167	9.93			102	6.07	20	67	+39.0	
test06	295	38.5			90	11.7	10	31	+69.7	
balu	110	69.2	41	25.8	28	17.6	3	7	+74.6	+31.8
struct	49	5.16	40	4.20	38	4.00	12	38	+22.5	+4.77
biomed	286	2.69	135	1.28	115	1.08	132	496	+59.9	+15.7
s9234	166	1.95	74	0.86	79	0.92	108	516	+52.8	-6.53
s13207	110	0.57	91	0.48	104	0.54	186	710	+5.27	-12.1
s15850	125	0.46	91	0.33	52	0.19	308	1197	+58.7	+42.4
industry2	525	1.32	193	0.49	319	0.81	478	1855	+38.6	-39.5

Table 5: Min cut and ratio cut ($\times 10^{-5}$) comparisons for MELO derived 2-way partitionings versus Paraboli [15] and EIG1 [11]. Paraboli results are quoted from [15]. All three algorithms require each cluster to contain at least 45% of the total modules. MELO runtimes are given for a Sun Sparc 10 and reported in seconds.

Test Case	Scheme	Number of Clusters - k									Avg % improv
		10	9	8	7	6	5	4	3	2	
19ks	MELO	9.85	9.10	8.31	7.87	6.84	6.14	5.32	4.99	4.79	+0.00
	EIG1	13.9	11.6	9.15	8.74	8.87	7.00	6.51	6.45	6.35	+18.9
	KP	9.09	9.37	10.7	9.52	9.00	9.00	6.95	6.58	6.20	+17.8
	SFC	15.1	14.3	13.8	13.2	12.2	11.1	8.37	7.48	5.44	+35.7
bm1	MELO	25.5	23.4	21.8	18.6	16.0	12.5	8.63	6.61	5.53	+0.00
	EIG1	33.3	31.1	26.9	22.7	17.0	11.3	8.63	6.61	5.53	+8.94
	KP	27.5	23.1	18.9	18.2	12.5	10.7	8.67	6.61	5.53	-4.97
	SFC	24.8	22.8	20.7	18.0	14.4	11.5	8.89	6.61	5.53	-3.17
prim1	MELO	44.6	41.9	39.7	37.0	34.0	29.4	22.5	17.1	13.4	+0.00
	EIG1	53.1	44.6	40.5	38.1	32.4	28.1	23.9	17.0	13.4	+2.51
	KP	44.7	41.3	32.3	33.2	31.3	29.9	21.2	14.7	13.5	-6.16
	SFC	38.9	36.7	35.2	31.7	28.8	26.0	21.8	14.6	13.4	-10.6
prim2	MELO	13.7	12.7	12.0	11.2	10.1	9.18	7.95	6.76	4.71	+0.00
	EIG1	11.2	10.9	10.1	9.73	9.33	8.33	7.85	7.69	5.55	-5.33
	KP	15.0	15.2	13.5	11.0	10.5	10.1	9.23	7.25	4.64	+7.38
	SFC	13.7	13.3	12.8	12.1	11.0	9.43	7.95	6.86	5.05	+4.14
test02	MELO	21.1	19.9	18.5	17.0	15.4	13.9	12.4	10.7	8.07	+0.00
	EIG1	31.3	31.4	20.2	29.8	28.6	28.8	18.4	18.2	12.4	+36.4
	KP	24.4	22.6	22.1	19.1	19.3	18.7	17.4	12.0	9.26	+16.8
	SFC	25.5	24.1	22.8	20.9	18.5	16.1	13.4	10.9	8.07	+12.4
test03	MELO	19.0	17.6	16.7	15.3	14.6	13.7	12.5	11.6	9.29	+0.00
	EIG1	20.3	19.9	17.7	17.0	16.7	17.6	16.0	14.3	11.9	+14.6
	KP	20.3	22.4	19.1	17.3	18.4	22.8	19.9	14.7	9.45	+19.2
	SFC	22.6	21.1	19.2	17.1	16.2	15.2	14.3	13.0	10.2	+12.0
test04	MELO	13.2	12.3	11.5	10.8	9.97	9.32	8.21	6.83	5.78	+0.00
	EIG1	14.3	12.5	11.8	11.1	10.2	10.3	9.08	8.65	5.85	+6.46
	KP	18.3	16.3	15.0	12.2	13.7	12.5	8.98	12.0	6.74	+22.9
	SFC	22.2	19.9	17.8	17.6	16.5	15.1	11.6	8.19	5.78	+30.7
test05	MELO	7.42	7.03	6.53	6.11	5.79	5.50	4.85	4.35	3.09	+0.00
	EIG1	7.50	6.82	6.70	6.12	5.38	4.97	4.26	4.06	3.09	-3.85
	KP	10.8	10.6	9.28	6.80	6.60	6.69	7.81	7.34	4.52	+27.2
	SFC	9.88	8.66	8.06	7.84	7.32	6.56	5.49	4.90	3.09	+16.1
test06	MELO	21.3	20.2	18.5	16.7	14.7	13.5	11.3	9.54	8.80	+0.00
	EIG1	17.8	17.3	16.0	15.6	16.2	17.3	19.9	15.6	14.3	+10.3
	KP	21.0	20.9	19.7	18.5	19.1	19.1	18.0	18.2	28.6	+24.9
	SFC	27.1	25.1	23.7	20.2	18.4	16.5	13.7	11.3	9.21	+17.3
balu	MELO	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.4	17.6	+0.00
	EIG1	72.2	76.6	81.8	91.3	84.6	74.5	57.8	55.4	47.2	+46.9
	KP	45.7	58.6	56.8	47.9	45.9	35.9	33.2	32.9	48.7	+14.2
	SFC	82.0	79.1	74.1	70.3	64.9	62.2	49.4	47.3	17.6	+34.3
struct	MELO	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	+0.00
	EIG1	9.52	9.15	8.72	7.96	8.03	6.60	6.15	5.68	4.85	-10.7
	KP	14.9	15.7	13.8	14.4	11.4	9.32	7.91	7.51	6.60	+23.8
	SFC	12.1	11.2	10.5	9.41	8.65	7.93	7.05	6.42	4.85	+2.04
biomed	MELO	1.87	1.73	1.62	1.49	1.34	1.23	1.11	0.89	0.61	+0.00
	EIG1	1.68	1.67	1.60	1.58	1.60	1.20	1.27	1.28	0.85	+8.29
	KP	3.22	3.03	2.65	1.75	1.63	1.36	1.83	1.16	0.84	+24.3
	SFC	1.84	1.69	1.59	1.47	1.51	1.48	1.25	1.15	0.85	+9.22
Average	EIG1	+4.39	+5.47	+2.64	+8.30	+11.8	+9.60	+13.9	+19.7	+20.0	+10.6
	KP	+10.2	+16.0	+13.4	+9.35	+12.8	+15.8	+20.1	+21.1	+23.2	+15.8
	SFC	+13.6	+13.5	+14.0	+14.0	+15.2	+15.4	+14.0	+13.2	+6.06	+13.2

Table 4: Scaled cost ($\times 10^{-5}$) comparisons for partitionings derived from MELO orderings versus EIG1 [11], KP [7], and SFC [2] algorithms.

[2] C. J. Alpert and A. B. Kahng, "Multi-way Partitioning Via Spacefilling Curves and Dynamic Programming," *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 652-657.

[3] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering," to appear in *IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1994.

[4] E. R. Barnes, "An Algorithm for Partitioning the Nodes of a Graph," *Siam J. Alg. Disc. Methods* Vol. 3, No. 4, 1992, pp. 541-549.

[5] J. P. Blanks, "Near-Optimal Placement Using a Quadratic Objective Function," *Proc. ACM/IEEE Design Automation Conf.*, 1985, pp. 609-615.

[6] T. N. Bui and B. R. Moon, "A Fast and Stable Hybrid Genetic Algorithm for the Ratio-Cut Partitioning Problem on Hypergraphs," *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 664-669.

[7] P. K. Chan, M. D. F. Schlag and J. Zien, "Spectral K-Way Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD*, 1994, pp. 1088-1096. (also see J. Zien, "Spectral K-Way Ratio Cut Graph Partitioning", M. S. Thesis, Computer Engineering Dept., UC Santa Cruz, March 1993).

[8] J. Frankle and R. M. Karp, "Circuit Placements and Cost Bounds by Eigenvector Decomposition," *IEEE Conf. Computer Aided Design*, 1986, pp. 414-417 (also see Jon Frankle, "Circuit Placement Methods Using Multiple Eigenvectors and Linear Probe Techniques", Ph.D. Thesis, College of Engineering, UC Berkeley, 1987).

[9] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1990, pp. 520-523.

[10] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1992, pp. 422-427.

[11] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD* 11(9), Sept. 1992, pp. 1074-1085.

[12] S. W. Hadley, B. L. Mark, and A. Vannelli, "An Efficient Eigenvector Approach for Finding Netlist Partitions", *IEEE Trans. on CAD* 11(7), July 1992, pp. 885-892.

[13] K. M. Hall, "An r-dimensional Quadratic Placement Algorithm", *Manag. Sci.*, 17(1970), pp. 219-229.

[14] B. Mohar, "The Laplacian Spectrum of Graphs", *Proc. 6th Quadrennial Intl. Conf. on Theory and Applications of Graphs*, 1988, pp. 871-898.

[15] B. M. Riess, K. Doll, and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 646-651.

[16] L. A. Sanchis, "Multiple-way Network Partitioning", *IEEE Trans. on Computers*, 38, 1989, pp. 62-81.

[17] Y. C. Wei and C. K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning", *IEEE Conf. on Computer-Aided Design*, 1989, pp. 298-301.

partitioning instance, and have found that MELO orderings lead to high-quality two-way and multi-way partitionings. We note many possible directions for future research:

- Modify MELO to run in sub-$O(n^2)$ time. During each iteration, MELO considers adding every possible vector to S. Alternatively, after the first vector is chosen, we could rank all the other vectors by their magnitude when added to the first vector. Then, instead of considering all possible vectors during an iteration, consider only vectors in a candidate set T of fixed size (e.g., 50), where T is constructed from the highest ranked vectors. Each time a vector chosen from T is added to S, the next highly ranked vector not in S or T is added to T. The remaining vectors are reranked periodically (e.g., every 100 iterations) and T is updated. We believe that this speedup should not adversely affect MELO's performance.

- Apply vector partitioning to clustering. A subset of vectors of relatively large magnitude that point in the same direction are guaranteed to correspond to a cluster of vertices with small cut (because of the Lemma). Thus, it should be possible to identify such subsets of vectors and thereby construct high-quality clusterings.

- Find and apply lower bounds. Graph spectra have been very useful for constructing lower bounds on partitioning, e.g., λ_2 is a lower bound on the cut of any 2-way partitioning. We would like to answer such questions as, "Assume that we have constructed a d-dimensional vector partitioning instance, and that we have an optimal solution for this instance. Can we guarantee the quality of the corresponding graph partitioning solution in terms of d?" The answer should help us understand how large d needs to be, i.e., for a given graph partitioning instance is there some d such that the vector partitioning instance is "close enough" to be sufficient?

- Finally, and most importantly, we would like to explore possible vector partitioning heuristics. The promising performance of MELO suggests that other vector partitioning heuristics will be successful. For example, an FM-type of local improvement method could be applied to 2-way vector partitioning. Also, algorithms and experiments designed for more than 10 eigenvectors hold promise.

Acknowledgements

We thank Pak Chan, Martine Schlag and Jason Zien for giving us the LASO, KP, and EIG1 code along with their entire set of experimental data. We also thank Jon Frankle for his useful discussions and insight.

References

[1] C. J. Alpert and A. B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning," *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 743-748.

Table 4 reports the results for our third experiment which compares MELO with the multi-way partitioning algorithms EIG1 [11], KP [7], and SFC [2]. The codes for EIG1 and KP were obtained from Dr. Pak Chan; however, note that the results reported in Table 4 are considerably better than the values reported in [7].[4] EIG1 constructs a 2-way ratio-cut partitionings by trying all splits of the $\vec{\mu_2}$ ordering, and the algorithm is iteratively applied to the largest remaining cluster. The SFC and DP-RP codes were acquired from their authors [2]; experiments for SFC were done using the partitioning-specific net model. The results for all four algorithms are given in Table 4. MELO averaged 10.6%, 15.8% and 13.2% improvement over EIG1, KP, and SFC respectively. MELO performed very consistently, outperforming the other algorithms in 32 of 39 head-to-head benchmark comparisons (see the Table's rightmost column) and over the entire range of k.

In our final set of experiments, we used MELO orderings to construct 2-way partitionings by choosing the one with lowest ratio cut from all possible splits of the ordering, with the constraint that each cluster contains at least 45% of the modules. We quote the Paraboli results of [15] as a source of comparison, and additionally compare against EIG1.[5] The MELO results reported are the best observed from splitting each of the ten orderings constructed for schemes #2, #3, and #4; we use three schemes since the best scheme for this application remains unclear. In fact, frequently one scheme performed best for one benchmark and worst for another. MELO ratio-cuts average 42.0% and -2.93% improvement over EIG1 and Paraboli respectively (though Paraboli averages -6.23% improvement over MELO). Table 5 also reports the runtimes required for MELO to construct and split orderings using two and ten eigenvectors, after the eigenvectors have been computed. Despite MELO's $O(dn^2)$ complexity, the runtimes are quite reasonable because the algorithm is so simple (see [1] for detailed runtimes for eigenvector computations). The results from Table 5 do not suggest MELO is a superior two-way partitioner; rather, that multiple eigenvectors can be used to yield high-quality balanced 2-way partitionings. We hypothesize that directly solving the two-way vector partitioning problem will improve these results even further.

6 Conclusion and Future Work

We have proposed a new *vector partitioning* formulation and shown how to transform a graph partitioning instance into a vector partitioning instance. When $d = n$, our formulation is exactly equivalent to graph partitioning. We have also proposed MELO, a simple linear ordering algorithm based on the vector

[4] For the experiments performed in [7], nets with more than 99 pins were removed before the eigenvectors were computed using the Frankle net model. For some of the netlists (test03, test04, and test06), removing large nets disconnected the graph, forcing $\lambda_2 = 0$. Since EIG1 uses $\vec{\mu_2}$ to determine its initial ordering, and since eigenvectors with 0 as an eigenvalue are degenerate, the EIG1 results were much worse than they would have been had the nets not been removed. In addition, KP's results also suffered by using $\vec{\mu_2}$. Hence, we ran both EIG1 and KP using the Frankle and partitioning-specific net models for the seven benchmarks used in [7] (prim1-2, test02-6). We observed that: (i) for EIG1, the partitioning-specific net model averaged 0.19% and 11.1% respective improvement over the Frankle net model and the results from [7]; (ii) for KP, the Frankle net model averaged 2.97% and 2.96% respective improvement over the partitioning-specific net model and the results from [7]. Hence in Table 4, results are given for the partitioning-specific net model for EIG1 and the Frankle net model for KP. We note that for MELO, only experiments with the partitioning-specific net model were performed.

[5] The circuits s9234, s13207, and s15850 contain multiple connected components, so we ran MELO and EIG1 on the largest component (between 97-99% of the modules) and added the remaining components to the smaller cluster, while ensuring that the 45% constraint was satisfied. Thus, the EIG1 results differ from those reported in [15] for this reason and since [15] uses the standard and we use the partitioning-specific net model.

ordering that uses ten eigenvectors ($d = 10$), and the best combined of the ten orderings ($d = 1 - 10$) for Scheme #2. We observe that the $d = 10$ orderings significantly outperform the single eigenvector ordering. Overall, the $d = 1 - 10$ orderings averaged 32.6% improvement over the $d = 1$ orderings but only 6.43% improvement over the $d = 10$ orderings. In addition, the last two rows of Table 3 show that most of the improvement occurs for the smaller k values, suggesting that the $d = 10$ orderings are of high-quality from end to end. In other words, for $k = 2$, a good ordering is not necessarily required as long as the ordering contains a single good splitting point; however, for larger k, $k - 1$ good splits are required, making it more difficult to derive a good partitioning from a poor ordering.

Test Case	Scheme	Number of Clusters - k									Avg % improv
		10	9	8	7	6	5	4	3	2	
19ks	$d = 1 - 10$	9.85	9.10	8.31	7.87	6.84	6.14	5.32	4.99	4.79	+0.00
	$d = 1$	15.1	14.3	13.8	13.2	12.2	11.1	9.51	7.49	6.25	+37.9
	$d = 10$	9.85	9.23	8.45	8.25	7.33	6.82	6.14	5.86	5.20	+6.71
bm1	$d = 1 - 10$	25.5	23.4	21.8	18.6	16.0	12.5	8.63	6.61	5.53	+0.00
	$d = 1$	29.6	26.2	23.9	20.8	16.7	12.5	9.02	6.61	5.53	+5.82
	$d = 10$	27.6	25.2	22.7	19.4	16.0	12.7	8.86	6.90	6.10	+4.51
prim1	$d = 1 - 10$	44.6	41.9	39.7	37.0	34.0	29.4	22.5	17.1	13.4	+0.00
	$d = 1$	80.4	75.4	68.7	61.6	54.9	47.1	38.4	31.7	13.4	+37.1
	$d = 10$	44.6	41.9	39.7	37.1	34.0	30.3	24.3	22.2	20.5	+7.58
prim2	$d = 1 - 10$	13.7	12.7	12.0	11.2	10.1	9.18	7.95	6.76	4.71	+0.00
	$d = 1$	19.6	18.2	16.8	15.4	13.5	11.2	9.45	7.18	5.55	+21.8
	$d = 10$	13.7	13.1	12.6	11.9	11.0	10.2	8.87	7.23	4.71	+5.42
test02	$d = 1 - 10$	21.1	19.9	18.5	17.0	15.4	13.9	12.4	10.7	8.07	+0.00
	$d = 1$	24.4	31.9	29.6	27.8	25.5	22.4	19.7	16.0	12.1	+34.3
	$d = 10$	21.2	20.0	18.7	17.2	15.5	13.9	12.6	11.8	9.48	+3.29
test03	$d = 1 - 10$	19.0	17.6	16.7	15.3	14.6	13.7	12.5	11.6	9.29	+0.00
	$d = 1$	30.7	28.3	25.4	23.1	21.5	19.3	16.5	14.9	11.8	+30.3
	$d = 10$	19.0	17.6	16.9	15.3	14.6	13.7	12.5	11.6	9.45	+0.32
test04	$d = 1 - 10$	13.2	12.3	11.5	10.8	9.97	9.32	8.21	6.83	5.78	+0.00
	$d = 1$	37.9	34.3	30.2	25.1	19.6	15.4	11.6	8.45	5.78	+42.8
	$d = 10$	13.2	12.3	11.5	10.9	10.2	9.37	8.21	6.83	5.93	+0.69
test05	$d = 1 - 10$	7.42	7.03	6.53	6.11	5.79	5.50	4.85	4.35	3.09	+0.00
	$d = 1$	11.7	10.8	9.60	8.78	7.96	7.04	6.32	4.95	3.09	+24.3
	$d = 10$	8.30	7.84	7.48	7.02	6.46	5.82	5.54	4.95	4.04	+12.3
test06	$d = 1 - 10$	21.3	20.2	18.5	16.7	14.7	13.5	11.3	9.54	8.80	+0.00
	$d = 1$	31.7	29.5	27.0	24.2	22.9	21.5	19.2	15.6	14.3	+35.4
	$d = 10$	32.0	28.9	25.0	22.2	21.1	18.7	16.9	14.9	13.8	+30.9
balu	$d = 1 - 10$	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.4	17.6	+0.00
	$d = 1$	95.3	91.3	86.1	79.3	70.2	62.2	60.9	55.7	47.8	+47.8
	$d = 10$	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.9	17.8	+0.35
struct	$d = 1 - 10$	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	+0.00
	$d = 1$	18.8	17.0	15.3	13.8	12.2	10.7	8.72	6.74	4.85	+26.0
	$d = 10$	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	+0.00
biomed	$d = 1 - 10$	1.87	1.73	1.62	1.49	1.34	1.23	1.11	0.89	0.61	+0.00
	$d = 1$	4.82	4.43	4.02	3.52	2.96	2.33	1.54	1.30	0.85	+47.7
	$d = 10$	2.07	1.91	1.73	1.57	1.42	1.31	1.16	0.89	0.61	+5.18
Average vs	$d = 1$	+37.1	+38.6	+37.6	+36.8	+35.3	+32.0	+30.0	+26.4	+19.6	+32.6
$d = 1 - 10$	$d = 10$	+5.15	+5.16	+4.81	+4.98	+5.34	+5.37	+7.10	+9.00	+11.0	+6.43

Table 3: Scaled cost values ($\times 10^{-5}$) for MELO orderings: $d = 1$ uses the ordering of $\vec{\mu_2}$; $d = 10$ uses $\vec{\mu_2}, \vec{\mu_2}, \ldots \vec{\mu_{11}}$; and $d = 1 - 10$ reports the best values from using 1-10 eigenvectors.

Test Case	Scheme	Number of Clusters - k									Avg %
		10	9	8	7	6	5	4	3	2	improv
19ks	#1	9.18	8.65	8.05	7.57	6.65	6.01	5.41	5.02	4.79	-2.37
	#2	9.85	9.10	8.31	7.87	6.84	6.14	5.32	4.99	4.79	+0.00
	#3	9.62	8.99	8.20	7.82	7.83	6.16	5.28	4.97	4.79	+0.69
	#4	9.17	8.62	8.01	7.44	6.52	5.87	5.40	5.02	4.79	-3.14
bm1	#1	25.9	23.8	21.4	17.9	14.1	11.6	9.01	6.61	5.53	-1.89
	#2	25.5	23.4	21.8	18.6	16.0	12.5	8.63	6.61	5.53	+0.00
	#3	25.2	22.8	21.0	18.2	15.4	11.0	9.02	6.61	5.53	-2.31
	#4	26.2	24.1	21.8	18.3	14.6	12.5	9.01	6.61	5.53	-0.02
prim1	#1	46.1	44.0	42.2	38.6	35.8	31.8	27.4	20.5	13.4	+7.24
	#2	44.6	41.9	39.7	37.0	34.0	29.4	22.5	17.1	13.4	+0.00
	#3	41.2	38.6	36.2	33.4	29.8	25.9	22.5	17.1	13.4	-6.48
	#4	46.4	43.7	41.5	38.6	34.6	27.9	26.2	18.5	13.4	+3.84
prim2	#1	12.6	12.7	11.4	10.5	9.66	8.94	7.76	6.75	4.81	-2.96
	#2	13.7	12.7	12.0	11.2	10.1	9.18	7.95	6.76	4.71	-3.54
	#3	12.7	12.1	11.5	10.7	10.3	9.21	8.31	6.82	4.72	-1.41
	#4	12.4	11.9	11.2	10.3	9.39	8.75	7.76	6.75	4.81	-4.74
test02	#1	21.4	20.1	18.8	17.6	16.7	15.5	14.1	11.2	8.23	+4.89
	#2	21.1	19.9	18.5	17.0	15.4	13.9	12.4	10.7	8.07	+0.00
	#3	21.8	21.0	19.9	18.6	16.7	15.2	12.3	10.8	8.07	+4.50
	#4	21.4	20.3	19.0	17.2	16.3	15.4	13.6	11.2	8.23	+4.18
test03	#1	19.6	19.9	18.6	17.1	14.9	14.0	12.8	11.5	9.14	+4.37
	#2	19.0	17.6	16.7	15.3	14.6	13.7	12.5	11.6	9.29	+0.00
	#3	18.3	17.3	16.9	15.2	14.4	13.5	12.2	11.2	9.29	-1.50
	#4	19.9	18.6	16.9	16.1	14.7	13.8	12.7	11.6	9.29	+2.11
test04	#1	13.6	12.6	11.9	11.2	10.3	9.59	8.89	7.57	5.78	+3.97
	#2	13.2	12.3	11.5	10.8	9.97	9.32	8.21	6.83	5.78	+0.00
	#3	13.4	12.4	11.8	11.0	10.2	9.24	8.43	6.83	5.78	+1.18
	#4	13.6	12.7	11.9	11.2	10.3	9.59	8.89	7.56	5.78	+4.04
test05	#1	7.18	6.94	6.66	6.38	5.96	5.58	5.13	4.77	3.09	+2.23
	#2	7.42	7.03	6.53	6.11	5.79	5.50	4.85	4.35	3.09	+0.00
	#3	7.45	7.15	6.68	6.28	5.91	5.62	5.14	4.77	3.09	+2.85
	#4	7.77	7.00	6.74	6.39	5.91	5.60	4.85	4.35	3.09	+1.71
test06	#1	24.4	19.3	17.9	16.3	14.6	12.9	11.6	9.96	8.80	+0.41
	#2	21.3	20.2	18.5	16.7	14.7	13.5	11.3	9.54	8.80	+0.00
	#3	20.7	19.7	18.4	17.0	14.8	13.0	11.3	9.47	8.80	-0.87
	#4	22.2	20.9	19.3	17.7	15.9	14.5	12.5	10.3	8.80	+5.40
balu	#2	54.0	50.1	46.5	43.2	40.0	36.7	32.3	24.4	17.6	+0.00
	#3	53.2	49.1	44.2	42.4	40.0	36.7	32.3	24.8	17.6	-0.96
struct	#2	12.9	12.0	10.9	9.82	8.46	7.56	6.53	5.54	4.25	+0.00
	#3	12.9	12.0	10.9	9.87	8.94	7.86	6.90	5.83	4.62	+3.12
biomed	#2	1.87	1.73	1.62	1.49	1.34	1.23	1.11	0.89	0.61	+0.00
	#3	1.97	1.86	1.72	1.53	1.39	1.26	1.09	0.88	0.61	+2.61
Average	#1	+0.62	+1.14	+1.04	+0.99	-0.08	+0.78	+5.71	+4.83	+0.26	+1.70
#2 vs	#3	-1.32	-0.58	-0.30	-0.09	+1.35	-1.08	+1.37	+0.95	+0.69	+0.01
	#4	+0.68	+0.91	+0.88	+0.86	-0.15	+0.79	+5.00	+3.28	+0.45	+1.41

Table 2: Scaled Cost ($\times 10^{-5}$) values for MELO orderings for four different weighting schemes. Scheme #1 uses $H = \infty$; #2 uses $H = \lambda_2 + \lambda_d$; #3 scales each eigenvector by dividing by its eigenvalue; and #4 adjusts H dynamically so that the sum of the unused eigenvector contributions is zero. The rightmost column gives the percentage improvements of #2 versus the other schemes for each benchmark, and the bottom rows give the percent improvement as a function of k.

In our second set of experiments, we explore the relationship between the quality of MELO partitionings and the choice of d. Table 3 gives Scaled Cost values for the single eigenvector ordering ($d = 1$), the MELO

Test Case	# Modules	# Nets	# Pins
19ks	2844	3282	10547
bm1	882	903	2910
prim1	833	902	2908
prim2	3014	3029	11219
test02	1663	1720	6134
test03	1607	1618	5807
test04	1515	1658	5975
test05	2595	2750	10076
test06	1752	1541	6638
balu	801	735	2697
struct	1952	1920	5471
biomed	6514	5742	21040
s9234	5866	5844	14065
s13207	8772	8651	20606
s15850	10470	10383	24712
industry2	12637	13419	48404

Table 1: Benchmark circuit characteristics.

it provides a single quantity that measures the quality of a linear ordering, and (iii) it permits comparisons to previous algorithms. With the Scaled Cost objective and no cluster size bounds, DP-RP has $O(kn^2)$ time complexity.

Our first experiments compare MELO orderings using the four proposed weighting schemes. For each scheme, we generated ten linear orderings for each benchmark by using the d eigenvectors with smallest eigenvalue (disregarding $\vec{\mu_1}$) for $1 \leq d \leq 10$. These experiments were not performed for the larger benchmarks because DP-RP's $O(n^2)$ space requirement makes it infeasible for netlists with more than 7-8000 modules. DP-RP was applied to each of the ten orderings, and the best Scaled Cost values that were observed are reported in Table 2. Surprisingly, the Scaled Costs did not vary much for the different schemes, though the second and third schemes performed slightly better.[3] The rightmost column of Table 2 reports the average percentage improvement of Scheme #2 versus the other schemes. Positive percent improvement is indicated by a +; if the percent improvement was negative, we report the percent improvement of the superior scheme versus Scheme #2 and indicate this with a −. The percentage improvements of Scheme #2 as a function of k are given in the last rows of the Table. From the bottom right of the Table, we see that #2 averages 1.70%, 0.01%, and 1.41% improvement over Schemes #1, #3, and #4 respectively. The results for the second and third schemes are indistinguishable; we choose the second scheme for our remaining experiments since it directly derives from the vector partitioning instance. The fourth scheme performed surprisingly poorly; however, we observed that although H fluctuated, it remained considerably higher than $\lambda_2 + \lambda_d$, i.e., the eigenvectors were not scaled by much. We hypothesize that this scheme may be more successful for larger d, From these results, it appears (at least for the MELO ordering construction) that the best weighting strategy is to make H small in order to maximize scaling.

[3] Our initial experiments did not include the balu, struct, and biomed test cases; hence, we only used the second and third schemes for these cases.

- **Scheme #3:** Instead of using MELO in the context of Equation (6), we wanted to preserve the original projection weights from Equation (2), i.e., $f(P^2) = \sum_{j=1}^{n} \alpha_{j1}^2 \lambda_j$. Recall that $\sum_{j=1}^{n} \alpha_{j1}^2 = |C_h|$ is a constant, hence the coefficients for the smaller λ_j terms should be as large as possible. Instead of minimizing $\sum_{j=1}^{n} \alpha_{j1}^2 \lambda_j$, we considered maximizing $\sum_{j=1}^{d} \frac{\alpha_{j1}^2}{\lambda_j}$, which is roughly equivalent. With this objective each pair of α_{j1}^2 terms remain in the same proportion as in Equation (2).

- **Scheme #4:** We are trying to maximize the sum in Equation (6), since this summation is proportional to the min-cut objective. In an attempt to minimize the error of this approximation, we require that the sum of the "contributions" of the unused $n-d$ eigenvectors is zero, i.e., $\sum_{j=d+1}^{n} \alpha_{j1}^2 (H - \lambda_j) = 0$. This occurs when[2]

$$H = \frac{E_1 - \sum_{j=1}^{d} \alpha_{j1}^2 \lambda_j}{|C_h| - \sum_{j=1}^{d} \alpha_{j1}^2}.$$

  Thus, after Step 6 in Figure 4, H is recomputed using C_1 as the set of vertices corresponding to vectors in S, and the coordinates of the vectors in Y and S are then readjusted. This scheme does not increase MELO's time complexity; however, it may make possible speedups infeasible.

5 Experimental Results

In this section, we present four sets of experiments that we performed with MELO:

- Comparisons of the weighting schemes proposed in the previous section,
- Comparisons with different values of d,
- Multi-way partitioning comparisons with EIG1, KP, and SFC algorithms,
- Balanced 2-way partitioning comparisons with EIG1 and Paraboli.

For these experiments, we use the set of ACM/SIGDA benchmarks listed in Table 1, the first nine of which were acquired from the authors of [2] and the others from the authors [15]. The eigenvectors computations were performed using LASO2 code, provided by the authors of [7]. Before the eigenvectors were computed, each netlist was transformed into a graph using the *partitioning-specific* net model introduced in [1]. This model constructs a clique for each p-pin net and assigns weight $\frac{4}{p(p-1)} \cdot \frac{2^p - 2}{2^p}$ to each edge in the clique. We also discuss use of the *standard* and *Frankle* [8] net models which respectively assign weights $\frac{1}{p-1}$ and $(\frac{2}{p})^{\frac{3}{2}}$ to edges in the clique.

To generate multi-way partitionings from MELO orderings, we apply the "DP-RP" algorithm of [2]. DP-RP accepts a vertex ordering and returns a *restricted partitioning*, i.e., a k-way partitioning such that each cluster is a contiguous subset of the ordering. DP-RP uses dynamic programming to find the optimal set of $k-1$ splitting points in the ordering; it applies to all of the partitioning objectives that we discuss. We choose to minimize the *Scaled Cost* objective because (i) we want to avoid artificial size constraints, (ii)

[2] This choice for H was originally suggested by [8] for 1-dimensional placement.

The MELO Algorithm	
Input:	Edge weighted graph $G(V, E)$ Number of eigenvectors to compute d
Output:	A linear ordering of V
1. Compute the d smallest eigenvectors of the Laplacian of G and construct the scaled eigenvector matrix V_d.	
2. Set $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, where $\vec{y_i^d}$ is the i^{th} row of V_d. Set $S = \emptyset$.	
3. **for** $j = 1$ **to** n **do**	
4. Find $\vec{y_i^d} \in Y$ such that the magnitude of the vector $\sum_{\vec{y} \in S \cup \vec{y_i^d}} \vec{y}$ is maximum.	
5. Add $\vec{y_i^d}$ to S and remove $\vec{y_i^d}$ from Y.	
6. Label v_i as the j^{th} vertex in the ordering.	

Figure 3: The MELO (Multiple Eigenvector Linear Orderings) Algorithm

[3]), e.g., choose some vertex to be in its own cluster and iteratively add vertices to the cluster to minimize cut. However, with this approach, each vector contains global partitioning information while the set of edges connected to any vertex is strictly local information. With graph traversals, there often are many equivalent choices of vertices, while with MELO ties are extremely rare.

We can also motivate MELO's greedy approach in the context of our maximum graph partitioning objective ($nH - f(P^k)$) for $k = 2$. In this case, the degree of the first cluster is the same as the degree of the second cluster (i.e., $E_1 = E_2$) and we can equivalently write Equation (4) as $H - \frac{f(P^2)}{2} = \sum_{j=1}^{n} \alpha_{j1}^2 (H - \lambda_j)$. Since in practice, we have only d of the eigenvectors, we cannot evaluate all the terms in this summation; however, the first d terms should provide a reasonable approximation, i.e.,

$$H - \frac{f(P^k)}{2} \quad \propto \quad \sum_{j=1}^{d} \alpha_{j1}^2 (H - \lambda_j). \tag{6}$$

If we can find a cluster C_1 that maximizes this summation, then our partitioning objective $H - \frac{f(P^k)}{2}$ should be close to optimal. Thus, a reasonable algorithm for constructing C_1 might be to start with $C_1 = \emptyset$, and then iteratively add the vertex to C_1 that optimizes the summation in Equation (6). Continuing until every vertex is a member of C_1 induces a linear ordering of vertices, and this is exactly the MELO construction.

We have not yet discussed how to choose H for this algorithm. When $d = n$, the choice of H is inconsequential; however, in practice, the choice of H can significantly affect the linear ordering. We tried the following different schemes for scaling the eigenvectors – experimental data for each scheme appears in the next section.

- **Scheme #1:** $H = \infty$. This is equivalent to no scaling, i.e., the eigenvector projections are equally weighted.
- **Scheme #2:** $H = \lambda_d + \lambda_2$. Here H is chosen to be relatively close to the largest used eigenvalue, ensuring that the $\sqrt{H - \lambda_j}$ scaling factors will vary considerably with j.

vector partitioning objective, instead of using Equation (3), we maximize

$$g(S^k) = \sum_{h=1}^{k} \frac{||\vec{Y_h}||^2}{|S_h|}.$$

Notice that this objective captures an "average vector" formulation: we wish to maximize the sum of the squared magnitudes of each subset vector divided by the number of vectors in the subset.

Minimum Largest Cut: For FPGA partitioning, it might be desirable to minimize the maximum degree of any cluster rather than the sum of the degrees as in Equation (1). This objective would the be to minimize $f(P^k) = \max_{1 \leq h \leq k} E_h = \max_{1 \leq h \leq k} \alpha_{jh}^2 \lambda_j$, or equivalently, to maximize $f'(P^k) = \min_{1 \leq h \leq k} \alpha_{jh}^2 (H - \lambda_j)$. Here, expressing f in terms f' is more nebulous than for the other objectives (subtracting f from nH or kH will not work); however, for this maximization objective, we can define the vector partitioning objective as

$$g(S^k) = \min_{1 \leq h \leq k} ||Y_h^n||^2 = \min_{1 \leq h \leq k} \alpha_{jh}^2 (H - \lambda_j) = f'(P^k).$$

Thus, vector partitioning is equivalent to a wide range of graph partitioning objectives. We believe that this observation opens the door for a new class of graph partitioning heuristics that try to solve the associated vector partitioning formulations.

4 Linear Ordering Algorithm

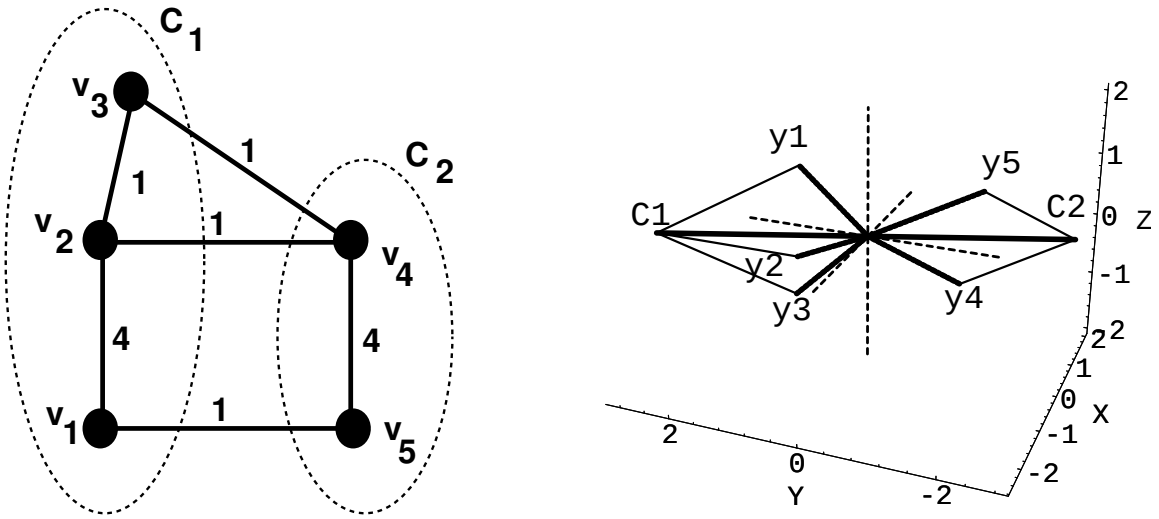
We now propose a simple, greedy graph partitioning heuristic that utilizes the corresponding vector partitioning instance. Instead of explicitly solving the vector partitioning problem, we construct a *linear ordering* of the vectors, which induces to a linear ordering of the graph vertices. Previous work [2] [11] [15] has shown that this approach can derive high-quality partitionings. Our linear ordering construction seeks to combine d distinct eigenvector orderings into a single ordering that utilizes all the eigenvector information.

Recall that in a good vector partitioning solution S^k, any given subset of vectors S_h will generally consist of vectors that sum to a vector of large magnitude. Thus, in Figure 4, we present the MELO (Multiple Eigenvector Linear Orderings) algorithm which greedily seeks to construct such a subset. MELO begins by constructing the d-dimensional vector partitioning instance, and initializes the current S to empty (Steps 1-2). MELO then adds the vector with largest magnitude to S, removes this vector from Y, and labels the vector's corresponding vertex first in the ordering. In each subsequent iteration, we choose the "best vector" from Y, meaning that when this vector is added to S the magnitude of the sum of the vectors in S is as large as possible, i.e., this vector is the best candidate for building our current subset. Thus, at any time during the algorithm S should consist of a reasonably good subset for vector partitioning. MELO's complexity is no worse that $O(dn^2)$, though speedups are possible; we discuss this in detail in the last section.

This idea of iteratively constructing a cluster (i.e., subset of vectors) seems to be the most direct method for construct a linear ordering. At first, MELO may not seem much different from a graph traversal (c.f.

Other Graph Partitioning Formulations

Finally, we close this section by noting how other graph partitioning objectives can induce corresponding vector partitioning objectives.



(a) (b)

Figure 2: (a) A 2-way partitioning solution for a five node graph and, (b) the construction of the subset vectors $\vec{Y_1}$ and $\vec{Y_2}$ (shown as C1 and C2) for the corresponding vector partitioning instance.

Min-cut with Cluster Bounds: Instead of requiring that $|C_h| = m_h$ for a prescribed m_h, we may allow each $|C_h|$ to have range, i.e., for each $1 \leq h \leq k$, there are user-specified bounds $L_h \leq |C_h| \leq W_h$. This constraint does not affect the min-cut objective, hence we still use the max-sum objective for vector partitioning but require that $L_h \leq |S_h| \leq W_h$, for $1 \leq h \leq k$.

Minimum Scaled Cost: Chan et al. [7] proposed to minimize $f(P^k) = \sum_{h=1}^{k} \frac{E_h}{|C_h|}$, i.e., a "generalized ratio-cut". The only cluster size bounds are that each cluster is nonempty. For the Scaled Cost objective, instead of using Equation (4) as our maximization objective, we use

$$kH - f(P^k) = \sum_{h=1}^{k}(H - \frac{E_h}{|C_h|}) = \sum_{h=1}^{k}(H - \frac{1}{|C_h|}\sum_{j=1}^{n}\alpha_{jh}^2\lambda_j) =$$

$$\sum_{h=1}^{k}(\frac{H}{|C_h|}\sum_{j=1}^{n}\alpha_{jh}^2 - \frac{1}{|C_h|}\sum_{j=1}^{n}\alpha_{jh}^2\lambda_j) = \sum_{h=1}^{k}\sum_{j=1}^{n}\alpha_{jh}^2\frac{H-\lambda_j}{|C_h|}.$$

Recall from the proof Theorem 3 that $||\vec{Y_h^n}||^2 = H|C_h| - E_h$, implying $\frac{||\vec{Y_h^n}||^2}{|C_h|} = H - \frac{E_h}{|C_h|}$. This is exactly the contribution of cluster C_h to our new maximization objective for graph partitioning; hence, for the

Theorem 3 still holds. Due to the orthogonality of the eigenvectors, $\vec{\mu_j}^T \vec{\mu_1} = 0$ for each $2 \leq j \leq n$ implying that for each such $\vec{\mu_j}$, $\sum_{i=1}^{n} \mu_{ij} = 0$ (i.e., the components of $\vec{\mu_j}$ sum to 0). Thus, by disregarding $\vec{\mu_1}$, the following desirable result is obtained.

Theorem 4: For any vector partitioning solution S^k for $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, when the first component of each $\vec{y_i^d}$ is disregarded, $\sum_{h=1}^{k} \sum_{\vec{y} \in S_h} \vec{y} = \vec{0}$.

Proof:

$$\sum_{h=1}^{k} \sum_{\vec{y} \in S_h} \vec{y} = \sum_{\vec{y} \in Y} \vec{y} = \sum_{i=1}^{n} \vec{y_i^d} = \sum_{i=1}^{n} [\nu_{i2}, \nu_{i3}, \ldots, \nu_{id}]^T.$$

The $(j-1)^{st}$ component of this vector is $\sum_{i=1}^{n} \nu_{ij}$. Since $\sum_{i=1}^{n} \mu_{ij} = 0$, we have

$$\sum_{i=1}^{n} \nu_{ij} = \sum_{i=1}^{n} \sqrt{H - \lambda_j}\, \mu_{ij} = 0.$$

Thus, every component of this vector is 0, proving the theorem. □

Theorem 4 states that the sum of all the subset vectors is the zero vector. Hence, when $k = 2$, any vector partitioning solution will yield two subset vectors $\vec{Y_1}$ and $\vec{Y_2}$ that are of the same magnitude and point in opposite directions. This zero-sum property may be useful in designing vector partitioning heuristics.

An Example

Figure 2(a) gives a possible 2-way partitioning for a five node graph. The eigenvalues for the Laplacian of this graph are

$$\lambda_1 = 0.000, \quad \lambda_2 = 2.298, \quad \lambda_3 = 2.469, \quad \lambda_4 = 8.702, \quad \lambda_5 = 10.531.$$

We choose $H = \lambda_5$, so that the last component of each vector in our vector partitioning instance is zero. We have

$$U_5 = \begin{bmatrix} 0.447 & 0.309 & 0.530 & 0.452 & 0.467 \\ 0.447 & 0.131 & 0.468 & -0.532 & -0.530 \\ 0.447 & -0.880 & 0.000 & 0.159 & 0.000 \\ 0.447 & 0.131 & -0.468 & -0.532 & 0.530 \\ 0.447 & 0.309 & -0.530 & 0.452 & -0.467 \end{bmatrix} \quad V_5 = \begin{bmatrix} 1.451 & 0.886 & 1.505 & 0.612 & 0.000 \\ 1.451 & 0.377 & 1.329 & -0.719 & 0.000 \\ 1.451 & -2.526 & 0.000 & 0.215 & 0.000 \\ 1.451 & 0.377 & -1.329 & -0.719 & 0.000 \\ 1.451 & 0.886 & -1.505 & 0.612 & 0.000 \end{bmatrix}$$

The first five rows of V_4 yield our vector partitioning instance, $Y = \{\vec{y_1}, \ldots, \vec{y_5}\}$, but Equation (5) permits us to disregard the first column, and our choice of H permits us to disregard the last column. Thus, the subset vectors corresponding to the 2-way partitioning solution are

$$\vec{Y_1} = \vec{y_1} + \vec{y_2} + \vec{y_3} = [-1.263, 2.834, 0.108]^T, \quad \vec{Y_2} = \vec{y_4} + \vec{y_5} = [1.263, -2.834, -0.108]^T.$$

The construction of these vectors is illustrated in Figure 2(b).

We can now verify Theorem 3 for this example. We have $||\vec{Y_1}||^2 = ||\vec{Y_2}||^2 = 9.637$. The constant from Equation (5) is $\sum_{h=1}^{2} \frac{10.531}{5} |C_h|^2 = 27.378$, hence $g(S^2) = 9.637 + 9.637 + 27.378 = 46.655$. The maximization objective $nH - f(P^k)$ for graph partitioning evaluates to $5 \cdot 10.531 - (3 + 3) = 46.652$, and we see that the objectives (except for rounding errors) are identical.

and the theorem will follow. For every i, the j^{th} component of $\vec{y_i^d}$ is ν_{ij}, hence the j^{th} component of $\vec{Y_h^d}$ is $\sum_{\vec{y_i^d} \in S_h} \nu_{ij}$. Therefore,

$$||\vec{Y_h^d}||^2 = || \sum_{y_i^d \in S_h} \vec{y_i^d}||^2 = \sum_{j=1}^{d} (\sum_{\vec{y_i^d} \in S_h} \nu_{ij})^2 = \sum_{j=1}^{d} (\sqrt{H - \lambda_j} \sum_{\vec{y_i^d} \in S_h} \mu_{ij})^2$$

and since membership of $\vec{y_i^d} \in S_h$ corresponds to membership of $v_i \in C_h$,

$$= \sum_{j=1}^{d} (\sqrt{H - \lambda_j} \sum_{v_i \in C_h} \mu_{ij})^2 = \sum_{j=1}^{d} (\sqrt{H - \lambda_j} (\vec{\mu_j}^T \vec{X_h}))^2 = \sum_{j=1}^{d} (\sqrt{H - \lambda_j} (\alpha_{jh}))^2 = \sum_{j=1}^{d} \alpha_{jh}^2 (H - \lambda_j)$$

Recalling that $\sum_{j=1}^{n} \alpha_{jh}^2 = |C_h|$, we now find that

$$||\vec{Y_h^n}||^2 = \sum_{j=1}^{n} \alpha_{jh}^2 (H - \lambda_j) = H \sum_{j=1}^{n} \alpha_{jh}^2 - \sum_{j=1}^{n} \alpha_{jh}^2 \lambda_j = H|C_h| - E_h .$$

It follows that

$$g(S^k) = \sum_{h=1}^{k} ||\vec{Y_h^n}||^2 = \sum_{h=1}^{k} (H|C_h| - E_h) = nH - f(P^k).$$

□

Corollary 2: Min-cut graph partitioning reduces to max-sum vector partitioning, hence max-sum vector partitioning is NP-Hard.

It is not hard to show that max-sum vector partitioning is actually NP-Complete, though optimum solutions may exist for certain fixed low values of d.

Corollary 3: $||\vec{y_i^n}||^2 = H - deg(i)$.

This corollary s just a special case of $||\vec{Y_h^n}||^2 = H|C_h| - E_h$ for $C_h = \{v_i\}$; however, it gives understanding as to why the graph partitioning and vector partitioning formulations are equivalent. We see that the degree of each graph vertex can be deduced by finding the magnitude of its corresponding vector, and further, when these vectors are added together, the magnitude of the resulting subset vector reveals the cost of the edges cut by the cluster.

A final theoretical result shows that ignoring the eigenvector $[\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]^T$ leads to the same formulation but in a geometrically more intuitive form. Recall that $\lambda_1 = 0$ and that the multiplicity of this eigenvalue equals the number of connected components of G. Assume that the 0 eigenvalues are ordered such that $\vec{\mu_1} = [\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]^T$. In this case, for any C_h with cluster vector $\vec{X_h}$, the magnitude of the projection of C_h onto $\vec{\mu_1}$ is $\alpha_{1h} = \vec{\mu_1}^T \vec{X_h} = \frac{|C_h|}{\sqrt{n}}$. Thus, for any P^k,

$$\sum_{h=1}^{k} \alpha_{1h}^2 (H - \lambda_1) = \sum_{h=1}^{k} (H - 0)(\frac{|C_h|}{\sqrt{n}})^2 = \sum_{h=1}^{k} \frac{H|C_h|^2}{n}, \tag{5}$$

which is a constant. Thus, when comparing two different partitioning solutions, the $\alpha_{1h}^2 (H - \lambda_1)$ terms can be disregarded. Similarly, for vector partitioning we can ignore the first component of each $\vec{y_i^n}$, and

Max-Sum Vector Partitioning: Given a set Y of n vectors, and subset sizes $m_1 \geq m_2 \geq \ldots \geq m_k > 0$ with $\sum_{h=1}^{k} m_h = n$, find S^k that satisfies $|S_h| = m_h$, for each $1 \leq h \leq k$, and maximizes

$$g(S^k) = \sum_{h=1}^{k} ||\vec{Y_h}||^2 \quad where \quad \vec{Y_h} = \sum_{\vec{y} \in S_h} \vec{y}. \tag{3}$$

We call $\vec{Y_h}$ the *subset vector* for S_h. Intuitively, the goal of max-sum vector partitioning is to find subsets of vectors that point in the same general direction, for these will form "natural" subsets. Notice that max-sum vector partitioning has a distinct advantage over min-cut partitioning, namely, an obvious pairwise similarity comparison. If $\vec{y_i}$ and $\vec{y_j}$ sum to a vector of large magnitude (relative to the sizes of $\vec{y_i}$ and $\vec{y_j}$), then these vectors likely belong in the same subset. Graph partitioning has no natural analogous similarity measure, as witnessed by the many ad hoc means in the literature (e.g., all-pairs shortest paths, $k-l$ connectivity [9], degree/separation [10], etc.).

Because the min-cut partitioning objective of Equation (1) requires us to *minimize* f and the max-sum objective of Equation (3) attempts to *maximize* g, the objectives at first seem incompatible. However, following the work of [8], we may transform f into a maximization objective. Let H be some constant greater than or equal to λ_n (for now, the choice of H is inconsequential). Using Equation (2), we formulate a new maximization objective for graph partitioning as

$$nH - f(P^k) = \sum_{h=1}^{k}(H|C_h| - \sum_{j=1}^{n} \alpha_{jh}^2 \lambda_j) = \sum_{h=1}^{k}(H \sum_{j=1}^{n} \alpha_{jh}^2 - \sum_{j=1}^{n} \alpha_{jh}^2 \lambda_j) = \sum_{h=1}^{k}\sum_{j=1}^{n} \alpha_{jh}^2 (H - \lambda_j). \tag{4}$$

The choice of $H \geq \lambda_n$ ensures that $nH - f(P^k) \geq 0$.

Let $U_d = (\mu_{ij})$ be the $n \times d$ matrix with column vectors $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_d}$, i.e., U_d consists of the first d columns of U (and $U_n = U$). We now construct a vector partitioning instance using a special scaling technique.

Definition: The $n \times d$ *scaled eigenvector matrix* $V_d = (\nu_{ij})$ is given by $\nu_{ij} = \mu_{ij}\sqrt{H - \lambda_j}$, i.e., by the eigenvector matrix U_d with each eigenvector column $\vec{\mu_j}$ scaled by the factor $\sqrt{H - \lambda_j}$.

Let $\vec{y_i^d}$ denote the i^{th} row of V_d. Then, consider the d-dimensional vector partitioning instance with the vector set $Y = \{\vec{y_1^d}, \vec{y_2^d}, \ldots, \vec{y_n^d}\}$, i.e., each graph vertex v_i corresponds to a vector $\vec{y_i^d}$. More specifically, $\vec{y_i^n}$ is the cluster vector for $\{v_i\}$ in the scaled eigenspace. For this choice of Y, we say that a graph partitioning solution $P^k = \{C_1, C_2, \ldots, C_k\}$ *corresponds to* a vector partitioning solution $S^k = \{S_1, S_2, \ldots, S_k\}$ if and only if $v_i \in C_h$ whenever $\vec{y_i^d} \in S_h$. Our main result shows that when $d = n$, the min-cut graph partitioning and max-sum vector partitioning objectives are identical, implying that optimal max-sum vector partitioning solutions corresponds to optimal min-cut graph partitioning solutions.

Theorem 3: Assume that P^k corresponds to S^k, then for $d = n$,

$$nH - f(P^k) = g(S^k).$$

Proof: For a given $S_h \in S^k$, we define $\vec{Y_h^d} = \sum_{\vec{y_i^d} \in S_h} \vec{y_i^d}$. First we establish that $||\vec{Y_h^d}||^2 = \sum_{j=1}^{d} \alpha_{jh}^2 (H - \lambda_j)$

$$f(P^2) = trace\left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} 8 & -6 & -2 \\ -6 & 7 & -1 \\ -2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\right) =$$

$$trace\left(\begin{bmatrix} 1.155 & 0.577 \\ -0.813 & 0.813 \\ -0.078 & 0.078 \end{bmatrix}^T \begin{bmatrix} 0.000 & 0.000 & 0.000 \\ 0.000 & 4.417 & -0.000 \\ 0.000 & 0.000 & 13.58 \end{bmatrix} \begin{bmatrix} 1.155 & 0.577 \\ -0.813 & 0.813 \\ -0.078 & 0.078 \end{bmatrix}\right) = trace\left(\begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix}\right) = 6.$$

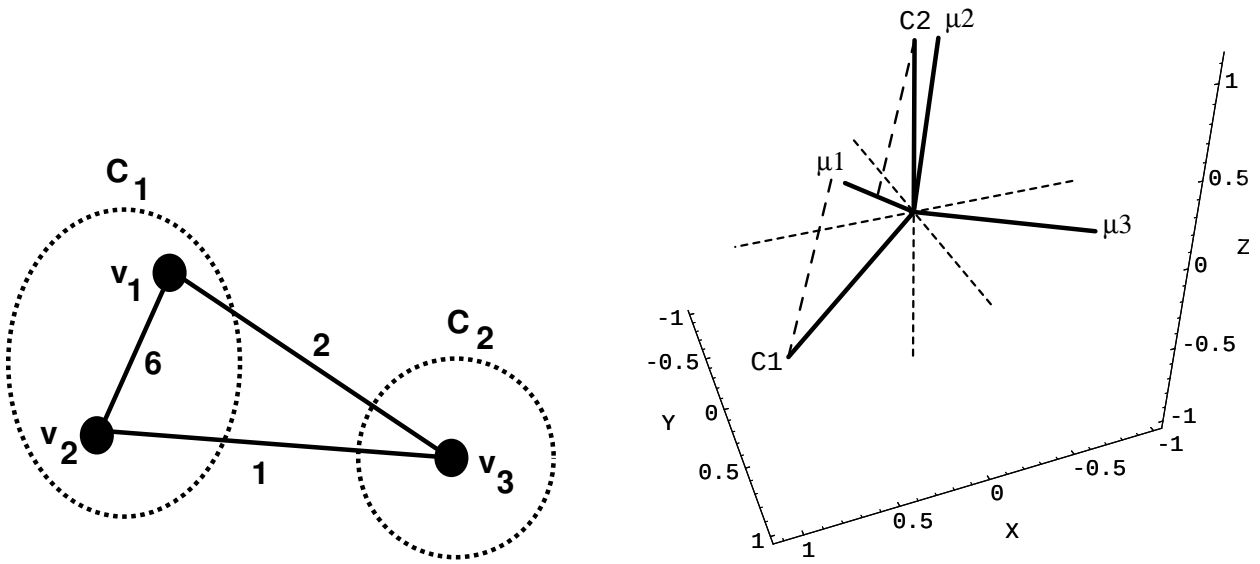


Figure 1: (a) A 2-way partitioning solution for a three node graph and, (b) the cluster vectors for C_1 and C_2 shown in the original and new eigenvector basis. The projections from C_1 and C_2 onto $\vec{\mu_1}$ are given by the dotted lines; the magnitudes of these projections are $\alpha_{11} = 1.155$ and $\alpha_{12} = 0.577$.

In summary, we have seen that a cluster C_h can be represented by an n-dimensional vector $\vec{X_h}$, and we have shown how to express $\vec{X_h}$ (and therefore a partitioning solution X) in terms of projections onto eigenvectors. We will now define the max-sum vector partitioning problem, show how to create an instance of this problem, and prove that an optimal max-sum vector partitioning solution implies an optimal min-cut graph partitioning solution.

3 The Vector Partitioning Problem

As one might imagine, vector partitioning is much like graph partitioning:

Definition: A k-way *vector partitioning* of a set of vectors $Y = \{\vec{y_1}, \vec{y_2}, \ldots, \vec{y_n}\}$ is a set of k subsets $S^k = \{S_1, S_2, \ldots, S_k\}$ such that each $\vec{y} \in Y$ is a member of exactly one S_h, $1 \leq h \leq k$.

Because the eigenvectors form a basis, we can write any n-dimensional vector $\vec{x}$ in terms of this new basis. Since the eigenvectors are orthonormal, $U^T U = I$, and also $UU^T = I$ since $U^T = U^{-1}$. Hence, we may write $\vec{x} = UU^T\vec{x} = U(U^T\vec{x})$, or equivalently $\vec{x} = \sum_{j=1}^{n}(\vec{\mu_j}^T\vec{x})\vec{\mu_j}$.

Definition: The $n \times k$ *projection matrix* $\Gamma = (\alpha_{jh})$ is given by $\alpha_{jh} = \vec{\mu_j}^T\vec{X_h}$, and thus $\Gamma = U^T X$. The h^{th} column of Γ is given by $\vec{\Gamma_h} = U^T\vec{X_h}$. We say that α_{jh} is the *magnitude* of the projection of $\vec{X_h}$ onto $\vec{\mu_j}$.

Thus, $\vec{\Gamma_h}$ is the representation of cluster C_h using the eigenvectors as a basis. We can now write each cluster vector $\vec{X_h}$ as $U\vec{\Gamma_h}$, or equivalently, $\vec{X_h} = \sum_{j=1}^{n}\alpha_{jh}\vec{\mu_j}$. Notice that $|C_h| = ||\vec{X_h}||^2 = \sum_{j=1}^{n}\alpha_{jh}^2 = ||\vec{\Gamma_h}||^2$. As an example, consider the graph in Figure 1(a) and its optimum 2-way partitioning solution, $P^2 = \{C_1, C_2\}$ with $C_1 = \{v_1, v_2\}$ and $C_2 = \{v_3\}$. We have

$$A = \begin{bmatrix} 0 & 6 & 2 \\ 6 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} 8 & -6 & -2 \\ -6 & 7 & -1 \\ -2 & -1 & 3 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 0.577 & -0.339 & -0.743 \\ 0.577 & -0.474 & 0.665 \\ 0.577 & 0.813 & 0.078 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} 0.000 & 0.000 & 0.000 \\ 0.000 & 4.417 & 0.000 \\ 0.000 & 0.000 & 13.58 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1.155 & 0.577 \\ -0.813 & 0.813 \\ -0.078 & 0.078 \end{bmatrix}$$

In Figure 1(b), we see that each $\vec{X_h}$ column gives the cluster vector coordinates for C_h using the original basis of solution vectors, and that each $\vec{\Gamma_h}$ gives the cluster vector coordinates using the new eigenvector basis (or eigenspace). Thus, $\vec{X_h}$ and $\vec{\Gamma_h}$ give two representations of the same cluster vector (observe from the X and Γ matrices that $||\vec{X_1}|| = ||\vec{\Gamma_1}|| = \sqrt{2}$, $||\vec{X_2}|| = ||\vec{\Gamma_2}|| = 1$). Also, notice that the second column of Γ is identical to the third row of U; this occurs because C_2 is a singleton cluster. In general, the i^{th} row of U gives the coordinates of the cluster vector for $\{v_i\}$ in the eigenspace.

Finally, we can express the min-cut objective in terms of eigenvector projections.

Theorem 2: $trace(X^T Q X) = trace(\Gamma^T \Lambda \Gamma)$.

Proof: By definition, for each $\vec{\mu_j}$, $Q\vec{\mu_j} = \lambda_j\vec{\mu_j}$, hence $QU = U\Lambda$ and

$$trace(X^T Q X) = trace(X^T U \Lambda U^T X) = trace((U^T X)^T \Lambda (U^T X)) = trace(\Gamma^T \Lambda \Gamma).$$

□

By combining Theorems 1 and 2 and performing the matrix multiplication, we observe:

Corollary 1:

$$f(P^k) = \sum_{h=1}^{k}\sum_{j=1}^{n}\alpha_{jh}^2\lambda_j\,. \tag{2}$$

This corollary is the k-way extension of Frankle and Karp's result for 2-way partitioning: $E_h = \vec{X_h}^T Q\vec{X_h} = \sum_{j=1}^{n}\alpha_{jh}^2\lambda_j$, for $h = 1$ and $h = 2$. Returning to our Figure 1 example, one can verify that

Definition: For any k-way partitioning P^k, the corresponding $n \times k$ *assignment* matrix $X = (x_{ih})$ has $x_{ih} = 1$ if $v_i \in C_h$ and $x_{ih} = 0$ otherwise. The h^{th} column of X, corresponding to cluster C_h, is a *cluster vector* denoted by $\vec{X_h}$.[1]

In other words, X represents a k-way partitioning solution as a $0-1$ matrix with k columns and exactly n nonzero entries. We can express the min-cut objective f explicitly in terms of the Laplacian and assignment matrices (a similar result was given by [7]).

Theorem 1: $f(P^k) = trace(X^T Q X)$.

Proof: The *trace* of a matrix is defined as the sum of its diagonal elements, e.g., $trace(A) = \sum_{i=1}^{n} a_{ii}$. We first show that $\vec{X_h}^T Q \vec{X_h} = E_h$, and the theorem will follow. Consider the vector $\vec{X_h}^T Q = \vec{X_h}^T (D - A)$. If $v_i \in C_h$, then the i^{th} entry of this vector is $deg(i) - \sum_{v_j \in C_h} a_{ij}$. If $v_i \notin C_h$, the i^{th} entry does not matter since it will be multiplied by 0 when $\vec{X_h}^T Q$ is multiplied by $\vec{X_h}$. Thus,

$$\vec{X_h}^T Q \vec{X_h} = \sum_{v_i \in C_h} [deg(i) - \sum_{v_j \in C_h} a_{ij}] = \sum_{v_i \in C_h} [\sum_{j=1}^{n} a_{ij} - \sum_{v_j \in C_h} a_{ij}] = \sum_{v_i \in C_h} \sum_{v_j \notin C_h} a_{ij} = E_h.$$

Now, the theorem easily follows:

$$trace(X^T Q X) = \sum_{h=1}^{k} \vec{X_h}^T Q \vec{X_h} = \sum_{h=1}^{k} E_h = f(P^k).$$

Observe that the non-diagonal terms in $X^T Q X$ are 0 since any two columns of X are orthogonal. □

Each $\vec{X_h}$ can be viewed as a linear combination of the n basis (solution) vectors corresponding to singleton clusters, e.g., $[11001]^T = [10000]^T + [01000]^T + [00001]^T$. Alternatively, we can express $\vec{X_h}$ in terms of a different set of basis vectors, namely, the eigenvectors of the Laplacian. The benefits of this change of basis will soon become clear.

Definition: An n-dimensional vector $\vec{\mu}$ is an *eigenvector* of Q with *eigenvalue* λ if and only if $Q\vec{\mu} = \lambda\vec{\mu}$. We denote the set of eigenvectors of Q by $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}$ with corresponding eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. The $n \times n$ *eigenvector matrix* $U = (\mu_{ij})$ has columns $\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_n}$ and the $n \times n$ *eigenvalue matrix* $\Lambda = (\lambda_{ii})$ has diagonal entries $\lambda_{ii} = \lambda_i$ and 0 entries elsewhere.

We assume that the eigenvectors are normalized, i.e., for $1 \leq j \leq n$, $\vec{\mu_j}^T \vec{\mu_j} = ||\vec{\mu_j}||^2 = 1$. The eigenvectors of Q have many interesting properties, including the following [14]:

- The eigenvectors are all mutually orthogonal, hence they form a basis in n-dimensional space.
- Each eigenvalue λ_j of Q is real.
- The smallest eigenvalue $\lambda_1 = 0$ and has a corresponding eigenvector $\vec{\mu_1} = [\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}}]^T$.
- If G is connected then $\lambda_2 > 0$. (In general, the multiplicity of 0 as an eigenvalue of Q is equal to the number of connected components of G.)

[1] We will implicitly assume that X and other related matrices represent a specific P^k. Ideally, we could write something like $X(P^k)$, but this would make the notation more cumbersome.

We propose a new geometric representation that utilizes a transformation from graph partitioning into a *vector partitioning* problem. As in the third approach, we construct n vectors using the coordinates of the first d Laplacian eigenvectors, but then we scale each coordinate by a function of the eigenvalue. The resulting vectors comprise a *vector partitioning* instance; the 1-1 correspondence between graph vertices and vectors permits us to develop a correspondence between graph and vector partitioning solutions. We make the following specific contributions:

- We propose a new max-sum vector partitioning problem. Vectors are partitioned into k disjoint subsets and the vectors in each subset are added together to form k *subset vectors*. The objective is to maximize the sum of the squared magnitudes of the subset vectors. Unlike partitioning using directional cosines [7], this objective captures both vector magnitude and direction.

- There are n 2-way partitioning solutions of the form $P^2 = \{\{v\}, \{V-v\}\}$; the n corresponding solution vectors comprise our vector partitioning instance. However, unlike [8] who uses probes to search for good solution vectors, we add these solution vectors together to form subset vectors that correspond to clusters of a multi-way partitioning solution.

- We show that when n eigenvectors are used, the min-cut graph partitioning and max-sum vector partitioning objectives are identical: good vector partitioning solutions correspond to good graph partitioning solutions. When only $d < n$ eigenvectors are used, the vector partitioning instance is an approximation of graph partitioning, but as d gets larger, the instance becomes a progressively better approximation to the real problem, until it is exact for $d = n$. Hence, unlike the works [1] [4] [7] that propose using $d = k$ eigenvectors for k-way partitioning, we believe that d should be as large as possible.

- Given a vector partitioning instance, we propose a greedy linear ordering heuristic called MELO, and demonstrate that MELO yields high-quality 2-way and multi-way partitionings. Our results suggest that more sophisticated vector partitioning heuristics hold much promise for the graph partitioning problem.

The rest of our paper is as follows. Section 2 shows how the min-cut graph partitioning objective can be expressed completely in terms of eigenvector projections. Section 3 presents the vector partitioning problem and proves the equivalence between graph to vector partitioning. We also show how this equivalence can be established for alternate graph partitioning objectives. Section 4 presents the MELO algorithm; Section 5 presents detailed experimental results for multi-way and 2-way partitioning; and Section 6 concludes with promising directions for future research.

2 Eigenvectors and the Min-Cut Objective

We now show how the min-cut objective can be expressed in terms of eigenvector projections.

Almost all of the proposed multi-way partitioning objectives involve some combination of the number of *cut edges* and cluster *size balance*. We assume that cluster sizes are fixed, and that the objective is to minimize the total cost of all cut edges. (Below, we discuss optimizing less constrained partitioning objectives.)

Min-Cut Graph Partitioning: Given the adjacency matrix A corresponding to a graph G, and cluster sizes $m_1 \geq m_2 \geq \ldots \geq m_k > 0$ with $\sum_{h=1}^{k} m_h = n$, find P^k that satisfies $|C_h| = m_h$, for each $1 \leq h \leq k$, and minimizes

$$f(P^k) = \sum_{h=1}^{k} E_h \quad where \quad E_h = \sum_{v_i \in C_h} \sum_{v_j \notin C_h} a_{ij}. \tag{1}$$

Hence, each E_h is the *degree* of cluster C_h. Notice that f counts every cut edge twice; we do this to avoid $\frac{1}{2}$ terms throughout this work.

Min-cut graph partitioning is known to be NP-complete, so heuristic methods must be invoked. Previous approaches have included seeded epitaxial growth, iterative improvement [16], genetic algorithms [6], etc. Spectral methods [1] [2] [4] [7] [8] [11] [13] [15] have been successful in recent years and are of particular interest for our present work. These works share a common trait of using eigenvectors to construct some type of geometric representation of the graph. We note four such representations:

- **Linear ordering or 1-dimensional placement:** Hall [13] showed that the second eigenvector of the Laplacian yields an optimum 1-dimensional placement in terms of squared wirelength. Barnes [4] proposed a method using multiple eigenvectors that reduces to sorting the coordinates of the adjacency matrix's largest eigenvector when $k = 2$. Hagen and Kahng [11] extended Barnes' idea to ratio-cut partitioning by considering all possible splits of the linear ordering induced by the Laplacian's second smallest eigenvector. Finally, Riess et al. [15] used analytic methods to create a 1-dimensional placement, also trying all possible splits to construct a 2-way partitioning.

- **Points in d-dimensional space:** Hall [13] also proposed using the coordinates of the second and third Laplacian eigenvectors for constructing a 2-dimensional placement. Alpert and Kahng [1] [2] extended this idea to higher dimensions, using the best d eigenvectors of the Laplacian to construct d-dimensional embeddings. In [1], geometric partitioning heuristics were applied to the embedded vertices; in [2], a spacefilling curve (SFC) was used to induce a linear ordering of the embedded vertices and dynamic programming was then used to split the linear ordering into a k-way partitioning.

- **d-dimensional vectors:** Chan et al. [7] use the same coordinates as in [1] [2], but view each embedded vertex as a *vector* rather than as a point in space. Their KP algorithm uses the directional cosine between two vectors as a similarity measure between vertices, and finds clusters accordingly.

- **Solution vectors:** Any two-way partitioning solution can be represented as an n-dimensional 0-1 *solution vector*. Frankle and Karp [8] proposed an algorithm that sends multiple *probes* into n-dimensional space, and uses eigenvectors to find the best *solution vector* with similar direction to a given probe.

Spectral Partitioning: The More Eigenvectors, The Better

Charles J. Alpert, So-Zen Yao†

UCLA Computer Science Department, Los Angeles, CA 90024-1596
† Cadence Design Sytems, San Jose, CA 94135

Abstract

A *spectral* partitioning heuristic is one that uses eigenvectors of a graph's adjacency or Laplacian matrix for constructing a solution. Previous heuristics have used eigenvectors to construct useful geometric representations of the graph, e.g., 1-dimensional placements. Our new representation maps each graph vertex to a vector in d-dimensional space, where d is the number of eigenvectors; these vectors constitute an instance of the *vector partitioning* problem. We show that when all the eigenvectors are used, an optimal vector partitioning solution produces an optimal graph partitioning solution: graph partitioning *exactly* reduces to vector partitioning. Motivated by this theoretical result, we use low-dimensional vector partitioning instances to construct linear orderings of the graph vertices. These linear orderings yield high-quality multi-way partitionings that significantly outperform the EIG1 [11], KP [7], and SFC [2] algorithms, and also produce balanced 2-way partitionings. Our experimental results suggest that solving vector partitioning is an effective approach to graph partitioning; we believe that this approach potentially opens the door to a new class of effective heuristics.

1 Introduction

Automatic circuit partitioning has become an essential tool for many important VLSI CAD applications. For example, hardware emulators require a partitioner to hierarchically divide a circuit into subcircuits; the partitioning solution determines the cost (e.g., the number of chips required for emulation) and the speed of the emulation. Partitioning plays a similarly important role in the design of multiple-FPGA systems. In IC design, circuit partitioners have been used in hierarchical placement, so that the quality of the partitioning solution strongly impacts the final die size of the design.

Given a netlist hypergraph, we assume that some transformation (e.g., replacing each hyperedge by a clique of weighted edges) has been used to derive a *graph partitioning* instance. Such an instance consists of a weighted graph $G(V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, and a symmetric $n \times n$ *adjacency matrix* $A = (a_{ij})$, where $a_{ij} > 0$ is the cost of the edge $(v_i, v_j) \in E$, and $a_{ij} = 0$ if no such edge exists. Let $deg(i) = \sum_{j=1}^{n} a_{ij}$ be the *degree* of v_i. The $n \times n$ *degree matrix* D is given by $d_{ii} = deg(i)$ and $d_{ij} = 0$ if $i \neq j$. The $n \times n$ *Laplacian* matrix of G is defined as $Q = D - A$.

Definition: A *k-way partitioning* of G is a set of *clusters* (subsets of V) $P^k = \{C_1, C_2, \ldots, C_k\}$ such that each $v_i \in V$ is a member of exactly one C_h, $1 \leq h \leq k$.