A Derivations

Proof. [**Proof-theorem 1**]

Proof of Eq. 4.2: Using the definitions we made in section 3, the original problem $\mathbf{r} = \mathbf{E} \cdot \mathbf{r} + \mathbf{q}$ can be expressed in the form of block matrices as follows:

$$\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_k \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{11} & \mathbf{E}_{12} & \dots & \mathbf{E}_{1k} \\ \mathbf{E}_{21} & \mathbf{E}_{22} & \dots & \mathbf{E}_{2k} \\ \dots & \dots & \ddots & \vdots \\ \mathbf{E}_{k1} & \mathbf{E}_{k2} & \dots & \mathbf{E}_{kk} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_k \end{bmatrix} + \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_k \end{bmatrix} \tag{A.1}$$

From Eq. A.1, the equation of any cluster i can be derived:

$$\mathbf{r}_i = \mathbf{E}_{ii}\mathbf{r}_i + \sum_{j \neq i} \mathbf{E}_{ij}\mathbf{r}_j + \mathbf{q}_i \tag{A.2}$$

Combining with Def. 4, this equals to:

$$\mathbf{r}_i = \mathbf{Z_i}(\sum_{j \neq i} \mathbf{E}_{ij}\mathbf{r}_j + \mathbf{q}_i) \tag{A.3}$$

Combining with Def. 3, we have:

$$\mathbf{E}_{ij}\mathbf{r}_j = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{E}}_{ij} \end{bmatrix} \begin{bmatrix} \mathbf{r}_j^{in} \\ \mathbf{r}_j^{\partial\mathcal{V}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \widetilde{\mathbf{E}}_{ij}\mathbf{r}_j^{\partial\mathcal{V}} \end{bmatrix} \tag{A.4}$$

Combining with Def. 5, Eq. A.3 can be written as:

$$\mathbf{r}_i = \mathbf{Z_i}(\begin{bmatrix} \mathbf{0} \\ \mathbf{f}_i \end{bmatrix} + \mathbf{q}_i) \tag{A.5}$$

Further expanding $\mathbf{Z_i}$ (Def. 4), $\mathbf{f}_i$ (Def. 5), $\mathbf{r}_i$ and $\mathbf{q}_i$ into block matrices and vectors, we get:

$$\begin{bmatrix} \mathbf{r}_i^{in} \\ \mathbf{r}_i^{\partial\mathcal{V}} \end{bmatrix} = \mathbf{Z}_i \begin{bmatrix} \mathbf{q}_i^{in} \\ \mathbf{q}_i^{\partial\mathcal{V}} \end{bmatrix} + \begin{bmatrix} \mathbf{Z}_i^{I} & \mathbf{Z}_i^{\mathrm{II}} \\ \mathbf{Z}_i^{\mathrm{III}} & \mathbf{Z}_i^{\mathrm{IV}} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_i \end{bmatrix} = \mathbf{Z}_i \begin{bmatrix} \mathbf{q}_i^{in} \\ \mathbf{q}_i^{\partial\mathcal{V}} \end{bmatrix} + \begin{bmatrix} \mathbf{Z}_i^{\mathrm{II}} \\ \mathbf{Z}_i^{\mathrm{IV}} \end{bmatrix} \mathbf{f}_i, \tag{A.6}$$

Thus we prove Eq. 4.2.

Proof of Eq. 4.3: Before proving the equation, we first introduce and prove the following lemma:

LEMMA A.1. *The global solution for boundary nodes is:*

$$\mathbf{r}^{\partial\mathcal{V}} = diag(\mathbf{Z}_1^{IV}, \dots, \mathbf{Z}_k^{IV})(\mathbf{f}+\mathbf{q}^{\partial\mathcal{V}}) + diag(\mathbf{Z}_1^{III}, \dots, \mathbf{Z}_k^{III})\mathbf{q}^{in} \tag{A.7}$$

Expanding $\mathbf{Z_i}$ (Def. 4) and combining Eq. A.7, for each cluster i, we get:

$$\mathbf{r}_i^{\partial\mathcal{V}} = \begin{bmatrix} \mathbf{Z}_i^{\mathrm{III}} & \mathbf{Z}_i^{\mathrm{IV}} \end{bmatrix} \begin{bmatrix} \mathbf{q}_i^{in} \\ \mathbf{q}_i^{\partial\mathcal{V}} \end{bmatrix} + \mathbf{Z}_i^{\mathrm{IV}}\mathbf{f}_i = \mathbf{Z}_i^{\mathrm{IV}}(\mathbf{f}_i + \mathbf{q}_i^{\partial\mathcal{V}}) + \mathbf{Z}_i^{\mathrm{III}}\mathbf{q}_i^{in} \tag{A.8}$$

Stacking the vectors of each cluster, Lemma. A.1 can be proved.

Based on the definition of flow (Def. 5), we stack the flow vectors from each cluster and obtain the relation between global solution for boundary nodes and global flows:

$$\mathbf{f} = \widetilde{\mathbf{E}} \cdot \mathbf{r}^{\partial\mathcal{V}}, \tag{A.9}$$

where $\widetilde{\mathbf{E}}$ is composed of rows and columns in $\mathbf{E}$ which correspond to boundary nodes.

Multiplying $\widetilde{\mathbf{E}}$ with two sides in Eq. A.7 and substituted with Eq. A.9, we get:

$$\mathbf{f} = \widetilde{\mathbf{E}} \cdot (diag(\mathbf{Z}_1^{\mathrm{IV}}, \dots, \mathbf{Z}_k^{\mathrm{IV}})(\mathbf{f}+\mathbf{q}^{\partial\mathcal{V}}) + diag(\mathbf{Z}_1^{\mathrm{III}}, \dots, \mathbf{Z}_k^{\mathrm{III}})\mathbf{q}^{in}), \tag{A.10}$$

which is equal to Eq. 4.3.

Proof. [**Proof-theorem 2**]

Proof of Eq. 5.4: Using $\mathbf{r}_i^{in} = [\mathbf{r}_i^{pin}; \mathbf{r}_i^{ov}]$, $\mathbf{q}_i^{in} = [\mathbf{q}_i^{pin}; \mathbf{q}_i^{ov}]$, $\mathbf{Z}_i^{\mathrm{II}} = [\mathbf{Z}_i^{\mathrm{II,pin}}; \mathbf{Z}_i^{\mathrm{II,ov}}]$ to substitute $\mathbf{r}_i^{in}$, $\mathbf{q}_i^{in}$ and $\mathbf{Z}_i^{\mathrm{II}}$ in Eq. A.6, we can get Eq. 5.4

Proof of Eq. 5.5: In the overlapping setting, boundary nodes receive messages not only from boundary nodes but also overhead nodes in other clusters. In that case, Eq. A.9 does not hold. But using the new definition of flow, we can get a similar equation:

$$\mathbf{f} = \hat{\mathbf{E}} \cdot \mathbf{r}^{\partial\mathcal{V}+ov}, \tag{A.11}$$

where $\hat{\mathbf{E}}$ is still a block diagonal matrix stacked by $\widetilde{\mathbf{E}}_{ij}$, but here $\widetilde{\mathbf{E}}_{i,j}$ stores cross-edges between boundary nodes in cluster i and boundary plus overhead nodes in cluster j, which has size $|\partial V_i| \times (|\partial V_j| + |V_j^{ov}|)$.

Analogous to Lemma A.1, we can get global solution for both boundary nodes and overhead nodes:

$$\begin{aligned} \mathbf{r}^{\partial\mathcal{V}+ov} = diag(&\begin{bmatrix} \mathbf{Z}_1^{\mathrm{II,ov}} \\ \mathbf{Z}_1^{\mathrm{IV}} \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{Z}_k^{\mathrm{II,ov}} \\ \mathbf{Z}_k^{\mathrm{IV}} \end{bmatrix}) (\mathbf{f} + \mathbf{q}^{\partial\mathcal{V}}) \\ &+ diag(\begin{bmatrix} \mathbf{Z}_1^{\mathrm{I,ov}} \\ \mathbf{Z}_1^{\mathrm{III}} \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{Z}_k^{\mathrm{I,ov}} \\ \mathbf{Z}_k^{\mathrm{III}} \end{bmatrix}) \mathbf{q}^{in} \end{aligned} \tag{A.12}$$

Multiplying $\hat{\mathbf{E}}$ with two sides in Eq. A.12 and substituted with Eq. A.11, we prove Eq. 5.5.

B Additional Related Work

Our proposed method requires a set of clusters as an input. Although proposing a new clustering method is out of scope for this paper, we review some traditional graph clustering methods below.

Clustering Methods. Graph clustering and partitioning is a very active research area. Clustering methods such as modularity-based optimization methods [28, 3], information-theoretic co-clustering [25], cut-based and spectral partitioning [10, 24, 21] find hard assignments of nodes to clusters. Most of these works try to minimize some version of the edge cuts (normalized/unnormalized). Others partition the graph with

heuristic approaches to the NP-hard problem of finding minimal vertex separators in a graph [26], or modified versions of PageRank for local partitioning [23]. As we will see, FlowR benefits from a small number of boundary nodes (which may be related to edge cuts, but there is no 1-1 mapping), as that controls the size of the systems, and we propose a strategy to convert a partition to overlapping with exactly that goal.

C Hardness of Proposed Problem

As discussed in Section 5.3, we want to partition the graph into clusters that are not individually too large such that the number of boundary nodes is minimized. We observe, however, that finding the optimal partition is related to (**but not the same as**) the following NP-hard problem: *Given a graph $G = (\mathcal{V}, \mathcal{E})$, an upper bound T on the total number of vertices in all clusters, and an upper bound $M < |\mathcal{V}|$ on the maximum number of vertices in any single cluster, find a cover of G containing k possibly overlapping clusters such that the number of boundary nodes is minimized.* Setting $k = 2$, $T = |\mathcal{V}|$, and $M < \lceil \frac{|\mathcal{V}|}{2} \rceil$ makes this problem an instance of the vertex bisection problem, which looks for a partition of the graph into two disjoint clusters of approximately equal size such that the number of vertices with cross-cluster edges–which are boundary nodes by definition–is minimized. This problem is NP-hard [5]. The exact problem we face is slightly modified: the quantity we would like to minimize in our methods **counts replicated boundary nodes multiple times**, and also the number of clusters need not be fixed. We conjecture that the problem remains hard, but a precise analysis remains an interesting future direction.

D Complexity

In this section we analyze the asymptotic complexity of FlowR with and without the use of overlapping clusters. A full analysis is deferred to a longer version of this paper.

FlowR: Distinct Clusters. During preprocessing, we compute for each cluster the inverse cluster matrices of Def. 4, which takes time $O(\sum_{i=1}^{k} |C_i|^3)$. In lines 8-9, we compute the component of the global flow that does not depend on the query vector $\mathbf{q}$. The bottleneck is the inversion of a $|\partial\mathcal{V}| \times |\partial\mathcal{V}|$ matrix, which takes time $O(|\partial\mathcal{V}|^3)$. In a distributed environment with sufficient resources, the k per-cluster processes can be computed in parallel, making preprocessing effectively run in time $O(\max_i |C_i|^3 + |\partial\mathcal{V}|^3)$.

In the query stage, we first multiply the matrix computed at the end of preprocessing by the entries of the query vector corresponding to boundary nodes, in $O(|\partial\mathcal{V}|^2)$ time. Then, we compute one matrix-vector product to get a $|\partial\mathcal{V}|$-dimensional vector in $O(|\partial\mathcal{V}||\mathcal{V}^{in}|)$ time before computing a second matrix-vector product with this vector in $O(|\partial\mathcal{V}|^2)$ time. Lastly, we compute the final result iteratively with Eq. (4.2), using $O(\max_i(|C_i||\partial\mathcal{V}_i|))$ time per iteration in a large distributed environment. As this analysis shows, the key way to speed up both stages is to reduce the number of boundary nodes $|\partial\mathcal{V}|$. This motivates our usage of overlapping clusters.

FlowR-OV: Overlapping Clusters. When we use overlapping instead of distinct clusters, we replicate some boundary nodes across clusters. Replication inherently incurs some additional costs, but crucially reduces the total number of boundary nodes (even counting replicated ones) to $|\partial\mathcal{V}|$; as discussed in our method, our replication policies are such that $|\partial\mathcal{V}| < |\partial\mathcal{V}^{old}|$, the number of boundary nodes in the nonoverlapping case. This leads to computational savings that more than compensate for the replication costs.

Now the matrix $\mathbf{H_4}$, formerly a block diagonal matrix, has size $(|\partial\mathcal{V}| + |\mathcal{V}^{ov}|) \times |\partial\mathcal{V}|$. This may increase the matrix multiplication time to compute $\mathbf{T_1}$ during preprocessing if $|\mathcal{V}^{ov}|$ is greater than $|\partial\mathcal{V}^{old}| - |\partial\mathcal{V}|$, the decrease in the number of boundary nodes. The end result, however, is that $\mathbf{T_1}$ is a $|\partial\mathcal{V}| \times |\partial\mathcal{V}|$ matrix. This will speed up the inversion at the end of preprocessing compared to inverting the larger $|\partial\mathcal{V}^{old}| \times |\partial\mathcal{V}^{old}|$ matrix in the non-overlapping case. Since $|\partial\mathcal{V}| < |\partial\mathcal{V}^{old}|$, this means that the query stage, which uses the inverted matrix, will also be faster.

E Further Analysis for FlowR-Hyb

As we mentioned in Sec. 6.2, our hybrid method FlowR-Hyb is effective in enhancing both preprocessing and query performance of FlowR-OV, and combines the strengths of both FlowR and iterative methods. Here, we investigate more how our method compares to the latter. Specifically, we compare FlowR variants and POWER in Table 6, since POWER is the best iterative method with respect to both convergence and speed. Columns 2-3 show the number of queries needed so that our methods' amortized cost per query (incl. preprocessing) is smaller than that of POWER. Columns 4-5 give the speedup in query time only (excl. preprocessing). FlowR-Hyb needs significantly fewer queries than FlowR-OV. As Tab. 6 suggests, even if preprocessing time is taken into account, our methods outperform POWER for only a few thousand queries.

References

Table 6: FlowR vs. POWER: Our methods have up to 32× faster query time. Even with the preprocessing time included, our methods outperform POWER for a few thousand queries (col. 2-3).

	# Queries req. to outperform		Query speedup	
Datasets	**FlowR-OV**	**FlowR-Hyb**	**FlowR-OV**	**FlowR-Hyb**
Amazon	1,436,775	–	1.02	0.55
RoadPA	–	4,793	0.76	1.93
RoadTX	–	7,398	0.78	1.87
Web-Stan	1,090	1,039	32.13	23.72
Google	2,018	673	6.77	10.59
Web-Berk	17,212	12,115	7.22	6.90

[21] Charles J Alpert, Andrew B Kahng, and So-Zen Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90(1):3–26, 1999.

[22] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast Unfolding of Communities in Large Networks. *JSTAT*, 2008(10):P10008, 2008.

[23] Fan Chung. A local graph partitioning algorithm using heat kernel pagerank. *Internet Mathematics*, 6(3):315–330, 2009.

[24] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. A Fast Kernel-based Multilevel Algorithm for Graph Clustering. In *KDD*, pages 629–634. ACM, 2005.

[25] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-Theoretic Coclustering. In *KDD*, pages 89–98, 2003.

[26] Cem Evrendilek. Vertex separators for partitioning a graph. *Sensors*, 8(2):635–657, 2008.

[27] George Karypis and Vipin Kumar. Multilevel k-way Hypergraph Partitioning. pages 343–348, 1999.

[28] Mark E. J. Newman and Michelle Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69(2):026113+, February 2004.