Beyond Good Shapes: Diffusion-based Graph Partitioning is Relaxed Cut Optimization*

Henning Meyerhenke

University of Paderborn, Department of Computer Science
Fuerstenallee 11, D-33102 Paderborn, Germany
E-mail: henningm@upb.de

Abstract. In this paper we study the prevalent problem of graph partitioning by analyzing the diffusion-based partitioning heuristic BUBBLE-FOS/C, a key component of the practically successful partitioner DIBAP [14]. Our analysis reveals that BUBBLE-FOS/C, which yields well-shaped partitions in experiments, computes a relaxed solution to an edge cut minimizing binary quadratic program (BQP). It therefore provides the first substantial theoretical insights (beyond intuition) why BUBBLE-FOS/C (and therefore indirectly DIBAP) yields good experimental results. Moreover, we show that in bisections computed by BUBBLE-FOS/C, at least one of the two parts is connected. Using arguments based on random walk techniques, we prove that in vertex-transitive graphs actually both parts must be connected components each. All these results may help to eventually bridge the gap between practical and theoretical graph partitioning.

Keywords: Diffusive graph partitioning, relaxed cut optimization, disturbed diffusion.

1 Introduction

Partitioning the vertices of a graph such that certain optimization criteria are met, occurs in many applications in computer science, engineering, and related fields. The most common formulation of the graph partitioning problem for an undirected (possibly edge-weighted) graph $G = (V, E)$ (or $G = (V, E, \omega)$) asks for a division Π of V into k pairwise disjoint subsets (*parts*) $\{\pi_1, \ldots, \pi_k\}$ of size at most $\lceil |V|/k \rceil$ each, such that the *edge cut* is minimized. The edge cut is defined as the total number (or total weight) of edges having their incident nodes in different subsets. Among many others, the applications of this $\mathcal{NP}$-hard problem include load balancing in numerical simulations [19] and image segmentation [6,20].

Despite recent approximation algorithms, simpler heuristics are preferred in practice, many of which can be found in the surveys [19] (graph partitioning) and [18] (graph clustering). Spectral algorithms have been widely used [8]; they are global optimizers based on graph eigenvectors. For computational efficiency or quality reasons, they have been mostly superseded by local improvement algorithms. Integrated into a multilevel framework, local optimizers such as Kernighan-Lin (KL) [10] can be

* Partially supported by German Research Foundation (DFG) Priority Programme 1307 *Algorithm Engineering*.

found in several popular partitioning libraries [4,9]. Unfortunately, theoretical quality guarantees are not known for KL. Another class of improvement strategies comprises diffusion-based methods [14,17]. While they are slower than KL, diffusive methods often yield a better quality, also when repartitioning dynamic graphs [14,15].

Motivation. The hybrid algorithm DIBAP is a multilevel combination of the diffusive algorithms BUBBLE-FOS/C [15] and TRUNCCONS. Particularly on graphs arising in numerical simulations, DIBAP is very successful [14]. For example, it has computed for six of the eight largest graphs of a popular benchmark set [21] a large number (more than 80 out of 144 when DIBAP was published) of their best known partitions with respect to the edge cut. The algorithm BUBBLE-FOS/C, which is related to Lloyd's k-means method [11], is an integral part of DIBAP responsible for good solutions on smaller representations of the input graphs. In experiments with graphs from numerical simulations, BUBBLE-FOS/C computes partitions with well-shaped parts. This comes along with a small number of boundary nodes (i. e., nodes with at least one neighbor in a different part) and a small edge cut [15]. However, apart from intuition (see Section 2), there has been no satisfactory theoretical explanation why BUBBLE-FOS/C and ultimately DIBAP produce such good partitions.

Contribution. With this work we answer several open questions regarding diffusion-based partitioning with BUBBLE-FOS/C. In Section 3 we prove a major insight about the optimization criterion of BUBBLE-FOS/C. The heuristic computes a k-way ($k \geq 2$) balanced partition that is the relaxed solution of a binary quadratic program (BQP) for finding the partition with *minimum edge cut*. As a byproduct, computing new center nodes for each part is related to a very similar BQP by the contribution to the constraints. Note that, while the insight about relaxed cut optimization alone may not be sufficient to guarantee a heuristic's practical success, we pursue here the other way around and analyze a heuristic known to produce good partitions. These results may pave the way for finding bounds on BUBBLE-FOS/C's quality for certain graph classes.

The achievements in Section 4 concern the *connectedness* of the parts in a bipartition ($k = 2$) computed by BUBBLE-FOS/C. We prove a result known for spectral partitioning [5], which is new for BUBBLE-FOS/C: In any undirected connected graph G, at least one of the two parts is connected. For vertex-transitive graphs (such as torus or hypercube), we use the random walk measure *hitting times* and conditional expectations to show that *both* parts are always connected.

2 Notation and Related Work

2.1 Notation

We consider in this paper undirected edge-weighted graphs $G = (V, E, \omega)$, which are triples with the set of n *vertices* (or *nodes*) V, a set of m *edges* $E \subseteq V \times V$, and an *edge weight function* $\omega : E \to \mathbb{R}_{>0}$. We also assume that the graphs are finite, connected, and simple, i. e., they do not contain self-loops (u,u) or multiple edges with the same endpoints. Connectedness can be enforced by focusing on the connected components.

Matrices $\mathbf{M}$ are written in bold font, a matrix entry at position (u,v) as $[\mathbf{M}]_{u,v}$. We use column vectors; the v-th entry of a vector w is denoted by $[w]_v$. In case we refer to the

v-th entry of the i-th vector, we write $[w_i]_v$. The symmetric positive semidefinite *Laplace matrix* matrix $\mathbf{L}$ of G [3, p. 27ff.] has the entries $[\mathbf{L}]_{u,v} = -\omega(\{u,v\})$ for $\{u,v\} \in E$, $[\mathbf{L}]_{u,u} = \deg(u)$ (with $\deg(u) = -\sum_{v \neq u}[\mathbf{L}]_{u,v}$), and $[\mathbf{L}]_{u,v} = 0$ otherwise.

2.2 Diffusion-based and Related Partitioning Techniques

Intuitively, a random walk [12] is likely to stay a long time in a dense graph region before leaving it via one of the few outgoing edges. There exist many graph clustering/partitioning techniques exploiting this notion (see [18]). Many diffusive processes are described by stochastic matrices and are therefore related to random walks [12]. Diffusion models many important transport phenomena such as heat flow; another application is localized load balancing in parallel computations. In such a discrete setting, diffusion is a local iterative process which exchanges splittable load entities between neighboring vertices, usually until all vertices have the same amount of load. For graph partitioning, diffusive algorithms and similarity measures are used to compute well-shaped partitions [15,17]. These works exploit the fact that diffusive processes send load faster into densely connected graph regions, which corresponds to the intuition that random walks stay longer in dense graph regions.

Meila and Shi [13] connect random walks to spectral partitioning. Spectral methods such as [20] solve relaxations of IPs that minimize the edge cut or the related ratio cut. They build on Fiedler's seminal work on spectral partitioning [5] and use eigenvectors of Laplace or adjacency matrices for partitioning. A spectral relaxation to the geometric k-means clustering problem is given by Zha et al. [22].

The diffusive partitioning algorithm BUBBLE-FOS/C, which we consider in this paper, is composed of the BUBBLE framework (described below) and the similarity measure FOS/C [15]. FOS/C (first order scheme with constant drain) introduces a drain-based disturbance into the first order diffusion scheme. With the disturbance, FOS/C reaches a steady state whose load vector w represents similarities of nodes. These similarities reflect whether nodes are connected by many paths of short length.

Definition 1. *(FOS/C) [15] Given a connected and undirected graph $G = (V,E,\omega)$ free of self-loops, a set of source nodes $\emptyset \neq S \subset V$, initial load vector $w^{(0)}$, and constants $0 < \alpha \leq (\deg(G)+1)^{-1}$ and $\delta > 0$.[1] Let the drain vector d (which is responsible for the disturbance) be defined as $[d]_v = (\delta n/|S|) - \delta$ if $v \in S$ and $[d]_v = -\delta$ otherwise. Then, the FOS/C iteration in time step $t \geq 1$ is defined as $w^{(t)} = \mathbf{M}w^{(t-1)} + d$, where $\mathbf{M} = \mathbf{I} - \alpha\mathbf{L}$ is the doubly-stochastic diffusion matrix and $\mathbf{L}$ the Laplace matrix of G.*

Lemma 1. *[15] For any $d \perp (1,\ldots,1)^T$ (i. e., $\langle d,(1,\ldots,1)^T\rangle = 0$), the FOS/C iteration reaches a steady state, which can be computed by solving the linear system $\mathbf{L}w = d$.*

Definition 2. *If $|S| = 1$ ($|S| > 1$), we call the FOS/C iteration to the steady state or its corresponding linear system a* single-source (multiple-source) FOS/C procedure. *Also, let $[w^{(t)}]_v^u$ ($[w]_v^u$) denote the load on node v in time step t (in the steady state) of a single-source FOS/C procedure with node u as source.*

[1] Here, the maximum degree of G is defined as $\deg(G) := \max_{u \in V} \deg(u)$.

Remark 1. [16] $[w]_v^u = \lim_{t\to\infty}([\mathbf{M}^t w^{(0)}]_v^u + n\delta(\sum_{l=0}^{t-1}[\mathbf{M}^l]_{v,u}) - t\delta)$, where $[\mathbf{M}^l]_{v,u}$ is the probability of a random walk (defined by the stochastic diffusion matrix $\mathbf{M}$) starting at v to be on u after l steps. $[\mathbf{M}^t w^{(0)}]_v^u$ converges to the balanced load distribution. Thus, the important part of an FOS/C load in the steady state is $n\delta(\sum_{l=0}^{t-1}[\mathbf{M}^l]_{v,u})$ – the sum of transition probabilities of random walks with increasing lengths. Observe that load vectors w can be made comparable by normalizing them such that $\sum_{v\in V}[w]_v = n$.

Algorithm Bubble-FOS/C$(G, k) \to \Pi$

```
01  Z = InitialCenters(G, k)  /* Arbitrary disjoint centers */
02  for τ = 1,2,... until convergence
      /* AssignPartition: */
03    parallel for each part π_p
04      Init d_p (S_p = {z_p}), solve and normalize L w_p = d_p
05    parallel for each node v ∈ V
06      Π(v) = argmax_{1≤p≤k} [w_p]_v
      /* ComputeCenters: */
07    parallel for each part π_p
08      Initialize d_p (S_p = π_p) and solve L w_p = d_p
09      z_p = argmax_{v∈π_p} [w_p]_v
10  return Π
```

Fig. 1. Sketch of the main BUBBLE-FOS/C algorithm.

The generic algorithmic framework behind the partitioning algorithm BUBBLE-FOS/C is the so-called BUBBLE framework. BUBBLE is related to Lloyd's k-means clustering algorithm [11] and transfers Lloyd's idea to graphs. The framework's first step chooses one initial representative (*center*) for each of the k parts. All remaining vertices are assigned to the closest (with respect to some measure) center vertex. After that, each part computes its new center for the next iteration. Then, the two latter operations are repeated alternately. BUBBLE-FOS/C is outlined in Figure 1, where $\Pi = \{\pi_1, \ldots, \pi_k\}$ denotes the set of parts, $\Pi(v)$ the part of node v, and $Z = \{z_1, \ldots, z_k\}$ the set of the corresponding center nodes. First, the algorithm determines possibly arbitrary, but pairwise disjoint initial centers (line 1). After that, with the new centers, the main loop is executed. It determines in alternating calls a new partition (`AssignPartition`, lines 3-6) and new centers (`ComputeCenters`, lines 7-9). BUBBLE-FOS/C implements these framework operations with k FOS/C procedures (more precisely with equivalent linear systems for efficiency) per major operation, single-source ones ($S_p = \{z_p\}$) for `AssignPartition` and multiple-source ($S_p = \pi_p$) ones for `ComputeCenters`. The loop is iterated until convergence is reached (convergence is guaranteed [14]) or, if time is important, a constant number of times.

Within the partitioner DIBAP one uses BUBBLE-FOS/C to compute solutions for smaller representations of the input graph with only a few thousand nodes and edges. This computation is reasonably fast and gives initial solutions that are usually better suited than KL-based ones. Initial solutions are refined by a faster local diffusion process (which yields initial solutions of lower quality, but refines well) within a multilevel process, see [14] for details. DIBAP is the combination of these two diffusive algorithms and yields very good experimental results in a reasonable amount of running

time (as an example, for $k \leq 16$ and graphs with less than one million nodes and edges, DIBAP requires less than a minute on standard hardware). Except for the connection to random walks and other intuitive explanations mentioned earlier in this section, there has been no theoretical evidence until now why the important initial solutions produced by BUBBLE-FOS/C have a high quality.

3 Optimization Criterion of BUBBLE-FOS/C

It has been shown before [14, Thm. 10], that the iterative optimization performed by the graph partitioning heuristic BUBBLE-FOS/C can be described by a potential function. This function F sums up the diffusion load of each vertex $v \in V$ in a single-source FOS/C procedure in time step τ with v's closest center vertex z_p as source. In fact, the results computed by the operations `AssignPartition` and `ComputeCenters` each maximize F for their fixed input (centers or parts, respectively). Moreover, random walks (and also related diffusion processes) can identify dense vertex subsets because they do not escape these regions easily via one of the few external edges. However, it has been unclear so far how these facts relate to the good experimental results of BUBBLE-FOS/C with respect to metrics more specific to graph partitioning.

With the upcoming analysis of BUBBLE-FOS/C, we show that – under mild conditions – BUBBLE-FOS/C solves a relaxed edge cut minimization problem. This is slightly surprising: In previous experiments with numerical simulation graphs [15], BUBBLE-FOS/C was compared to the popular partitioning libraries KMETIS and JOSTLE. The best improvements by BUBBLE-FOS/C could be seen regarding the number of boundary nodes and the shape of the parts. Yet, concerning the edge cut, the improvement over the other libraries was not as clear, probably because KMETIS and JOSTLE focus chiefly on the edge cut.

3.1 Edge Cut Minimization

Our plan is to express cut minimization by a binary quadratic programming problem (BQP) based on matrices and vectors equivalent or similar to those used in BUBBLE-FOS/C. For this purpose we introduce some notation first. Define a binary indicator vector $x_{(p)} \in \{0,1\}^n$ for part p, $1 \leq p \leq k$, with $[x_{(p)}]_v = 1 \Leftrightarrow v \in \pi_p$. Let $X \in \{0,1\}^{n \times k}$ be the matrix whose p-th column is $x_{(p)}$. Moreover, let $y_{(p,p')} := x_{(p)} - x_{(p')}$ and Y the matrix whose columns are the vectors $y_{(p,p')}$, $1 \leq p < p' \leq k$.

It is well-known [5] that $x^T \mathbf{L} x = \sum_{\{u,v\} \in E} \omega(\{u,v\})([x]_u - [x]_v)^2$. Hence, finding a balanced partition with minimum edge cut can be written as:

$$\begin{aligned} \min_{X \in \{0,1\}^{n \times k}} \quad & \textstyle\sum_{1 \leq p \leq k} x_{(p)}^T \mathbf{L} x_{(p)} && (1) \\ \text{subject to} \quad & \|x_{(p)}\|_1 = \tfrac{n}{k} && \text{(balanced parts)} \\ & \textstyle\sum_{1 \leq p \leq k} [x_{(p)}]_v = 1 \; \forall v \in V && \text{(exactly one part per node).} \end{aligned}$$

3.2 `AssignPartition` Computes Relaxed Minimum Cuts

Assume we use BUBBLE-FOS/C to find a balanced ($|\pi_i| = |\pi_j| \ \forall 1 \leq i, j \leq k$) k-partition with minimum (or in practice at least small) edge cut of an undirected graph $G = (V, E, \omega)$ with n nodes, $n/k \in \mathbb{N}$. To find a good solution, BUBBLE-FOS/C alternates the operations `AssignPartition` and `ComputeCenters`. Eventually, it finds a local optimum of the potential function F described above [14]. In the original formulation of BUBBLE-FOS/C, we solve k linear systems $\mathbf{L}w_p = d_p$, $1 \leq p \leq k$, for each `AssignPartition` and `ComputeCenters` operation, respectively. Recall that d_p is the drain vector that changes according to the set of source nodes and w_p is the resulting load vector.

To ensure balanced parts, `AssignPartition` can be followed by an operation called `ScaleBalance` [15]. `ScaleBalance` searches iteratively for scalars β_p such that the assignment of vertices to parts according to $\operatorname{argmax}_{1 \leq p \leq k}[\beta_p w_p]_v$ (instead of $\operatorname{argmax}_{1 \leq p \leq k}[w_p]_v$) results in balanced parts. A simple iterative search for suitable β_p is not always successful in practice, but in many cases it is.

Remark 2. Let $1 \leq p \leq k$. If the β_p were known beforehand, they could be integrated into the drain vector. The resulting linear systems to solve would be $\mathbf{L}(\beta_p w_p) = (\beta_p d_p)$. Hence, it does not make a difference whether suitable β_p are searched such that $\operatorname{argmax}_{1 \leq p \leq k}[\beta_p w_p]_v$ results in balanced parts or if we solve $\mathbf{L}(\beta_p w_p) = (\beta_p d_p)$ from the very beginning.

That is why we assume the scalars β_p to be known for now with $0 < \beta_p \neq \beta_{p'} < 1$ for all $1 \leq p \neq p' \leq k$. For the BQP formulation this is feasible, as will become clear in the remainder of this section. It is essential that the drain vectors are adapted accordingly:

Definition 3. *The entry of vertex $v \in V$ in the drain vector $d_p^{(A)}$ (A for* assign*) for the FOS/C procedure of part π_p with center z_p in the operation `AssignPartition` with scale value β_p is defined as*

$$[d_p^{(A)}]_v = \delta \cdot \beta_p \cdot \begin{cases} (n-1) & \text{if } v = z_p \\ -1 & \text{otherwise.} \end{cases}$$

Remark 3. If $k = 2$, instead of solving $\mathbf{L}w = d_1^{(A)}$ and $\mathbf{L}w_2 = d_2^{(A)}$, it is sufficient to solve $\mathbf{L}(w_1 - w_2) = d_1^{(A)} - d_2^{(A)}$. Then, to assign vertices to parts, one does not search for argmax (the part with the highest load for the vertex) but makes a sign test. Such a new linear system $\mathbf{L}w_{(p,p')} = d_{(p,p')}^{(A)}$ with $w_{(p,p')} := w_p - w_{p'}$ and $d_{(p,p')}^{(A)} := d_p^{(A)} - d_{p'}^{(A)}$ (if $k = 2$, then $p = 1$ and $p' = 2$) is called *fused (linear) system*. We will see in the proofs of Lemmas 2 and 3 that this fusion technique can be extended easily to $k > 2$ parts.

Lemma 2. *Let $1 \leq p \leq k$. Given a graph $G = (V, E, \omega)$ with n vertices and $\frac{n}{k} \in \mathbb{N}$, its Laplace matrix $\mathbf{L}$, k center vertices $Z = \{z_1, \ldots, z_k\}$, k pairwise different real scalars $0 < \beta_p < 1$ (with $\frac{1}{3} < \frac{\beta_p}{\beta_q} < 3$ for $1 \leq p \neq q \leq k$), the FOS/C drain constant δ, and the corresponding drain vectors $d_p^{(A)}$ (one for each part p).*

The BQP for finding a balanced k-partition $\Pi = \{\pi_1, \ldots, \pi_k\}$ with minimum cut in G under the condition $z_p \in \pi_p$ can be reformulated as:

$$\min_{X \in \{0,1\}^{n \times k}} \sum_{1 \le p \le k} x_{(p)}^T \mathbf{L} x_{(p)} \tag{2}$$

$$\text{subject to } y_{(p,p')}^T d_{(p,p')}^{(A)} = n\delta(\beta_p + \beta_{p'}) \ \forall (p,p') \tag{3}$$

$$\text{with } y_{(p,p')} := x_{(p)} - x_{(p')} \text{ for all } 1 \le p < p' \le k.$$

In the proof (full paper version) it becomes clear that the $\binom{k}{2}$ constraints are chosen such that the center vertices do not change their parts and that the parts have equal size.

Corollary 1. *Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be a balanced partition with minimum cut. If the set of center nodes $Z = \{z_1, \ldots, z_k\}$ is chosen in Lemma 2 such that $z_p \in \pi_p$ ($1 \le p \le k$), then the BQP (2), (3) computes Π or another balanced partition with minimum cut.*

Due to the $\mathcal{NP}$-hardness of BQP optimization, we aim at relaxed solutions. Instead of choosing only 0 or 1 in the indicator vectors, we now allow the entries of the relaxed indicator vectors $x_{(p)}$ to take on arbitrary real values. Moreover, we use $y_{(p,p')} := x_{(p)} - x_{(p')}$ in the objective function to use the fusion technique described in Remark 3 (in the integral problem, the use of $y_{(p,p')}$ instead of $x_{(p)}$ in the objective function would still model the edge cut, as the change is constant). These changes yield the new optimization problem

$$\min_{Y \in \mathbb{R}^{n \times \binom{k}{2}}} \sum_{1 \le p < p' \le k} y_{(p,p')}^T \mathbf{L} y_{(p,p')} \text{ with constraints as in (3).} \tag{4}$$

Lemma 3. *The global minimum $\overline{Y}$ of Problem (4) can be computed by solving and evaluating k linear equations of the form $\mathbf{L} z_p = -\frac{1}{2} d_p^{(A)}$ ($1 \le p \le k$), where*

$$\overline{y}_{(p,p')} = \frac{n\delta(\beta_p + \beta_{p'})}{z_{(p,p')}^T \cdot d_{(p,p')}^{(A)}} \cdot z_{(p,p')} \text{ and } z_{(p,p')} := z_p - z_{p'},\ 1 \le p < p' \le k.$$

Proof. Recall that `AssignPartition` solves k linear systems of the form $\mathbf{L} x_{(p)} = d_p^{(A)}$ and assigns each node to the part with the highest load for that node. It is essential to observe that this is equivalent to solving $\binom{k}{2}$ linear systems of the form $\mathbf{L} y_{(p,p')} = d_{(p,p')}^{(A)}$ and deciding a partial order with respect to the higher load for each node based on its sign in $y_{(p,p')} = x_{(p)} - x_{(p')}$. Note that, before performing scale balancing, all load vectors x are normalized by adding a proper multiple of $\mathbf{1} = (1, \ldots, 1)^T$ such that $\sum_{v \in V} [x]_v = n$. This ensures a common basis for comparison and does not affect the equations, because $\mathbf{L}\mathbf{1} = 0$ and $d_p \cdot \mathbf{1} = 0$. After $\binom{k}{2}$ comparisons for each node v, the "winning" part (i. e., the one with the highest load for v) has been determined. (Of course, for efficiency reasons, one would not perform such a large number of comparisons. Instead one solves k linear systems and makes $k-1$ comparisons per node.)

Regarding Eq. (4), using standard multidimensional calculus, one can easily see that the function $f(Y) := \sum_{1 \le p < p' \le k} y_{(p,p')}^T \mathbf{L} y_{(p,p')}$ is differentiable over $\mathbb{R}^{n \times \binom{k}{2}}$, because it is a sum of differentiable functions. Furthermore, each constraint function

$h(y_{(p,p')}) := y_{(p,p')}^T d_{(p,p')}^{(A)} - n\delta(\beta_p + \beta_{p'})$ is continuously differentiable over $\mathbb{R}^n$. Hence, we can continue by using a Karush-Kuhn-Tucker argument (see [2, Ch. 4]) and let $\overline{Y} = (\overline{y}_{(1,2)}, \overline{y}_{(1,3)}, \ldots, \overline{y}_{(k-1,k)})$ be a feasible solution. For $\overline{Y}$ to be a global minimum, a vector $\Lambda = (\Lambda_{(1,2)}, \Lambda_{(1,3)}, \ldots, \Lambda_{(k-1,k)}) \in \mathbb{R}^{\binom{k}{2}}$ must exist with

$$\nabla f(\overline{Y}) + \textstyle\sum_{1 \le p < p' \le k} \Lambda_{(p,p')} \nabla h(\overline{y}_{(p,p')}) = 0,$$

$$\text{which yields} \qquad 2\mathbf{L} \textstyle\sum_{1 \le p < p' \le k} \overline{y}_{(p,p')} = -\sum_{1 \le p < p' \le k} \Lambda_{(p,p')} d_{(p,p')}^{(A)} .$$

Such a vector Λ indeed exists: We first solve the linear systems $\mathbf{L} z_p = -\frac{1}{2} d_p^{(A)}$ for all $1 \le p \le k$. With $z_{(p,p')} := z_p - z_{p'}$ we have for all $1 \le p < p' \le k$: $\mathbf{L} z_{(p,p')} = -\frac{1}{2} d_{(p,p')}^{(A)}$, so that $\mathbf{L} \sum_{1 \le p < p' \le k} z_{(p,p')} = -\frac{1}{2} \sum_{1 \le p < p' \le k} d_{(p,p')}^{(A)}$. Let $\overline{y}_{(p,p')} := \Lambda_{(p,p')} z_{(p,p')}$, so that we arrive at

$$\mathbf{L} \overline{y}_{(p,p')} = -\frac{1}{2} \Lambda_{(p,p')} d_{(p,p')}^{(A)} \ \forall 1 \le p < p' \le k$$

$$\Rightarrow \mathbf{L} \sum_{1 \le p < p' \le k} \overline{y}_{(p,p')} = -\frac{1}{2} \sum_{1 \le p < p' \le k} \Lambda_{(p,p')} d_{(p,p')}^{(A)} .$$

Hence, a suitable Λ exists. Following [2, Thm. 4.3.8], f and h are convex functions or the sum of convex functions (again, this is easy to check, for f by using that $\mathbf{L}$ is positive semidefinite ($x^T \mathbf{L} x \ge 0 \forall x$)), so that $\overline{Y}$ is a global optimum of Equation (4). Finally, some rearranging suffices to compute each $\overline{y}_{(p,p')}$ and $\Lambda_{(p,p')}$ from $z_{(p,p')}$. □

Let us make clear now why the assumption of already known β_p is feasible. First, recall from Remark 2 that the result of the linear systems is the same regardless when the β_p are introduced into the equations. Lemma 2 tells us that the actual choice of the β_p is hardly relevant for the BQP to work – as long as they are not equal, lie between 0 and 1, and their quotient is neither too small nor too large. Hence, we choose suitable β_p such that the BQP works. In practice, however, we cannot make such a choice for BUBBLE-FOS/C a priori. Yet, given the mild conditions, we can assume that the scalars β_p computed by `ScaleBalance` will fulfill the constraints mentioned above in the vast majority of cases. Therefore, we can conclude this section with the following insight:

Theorem 1. *Let $k \ge 2$. Given a graph $G = (V, E, \omega)$ with n nodes ($n/k \in \mathbb{N}$) and a set Z with one center vertex for each of the k parts. Then, the two consecutive operations `AssignPartition` and `ScaleBalance` with suitable β_p ($1 \le p \le k$) together compute the global minimum of the Optimization Problem (4). Problem (4) is a relaxed version of BQP (2), (3). If $Z = \{z_{1,\ldots,}z_k\}$ is given such that $z_p \in \pi_p$ and $\Pi = \{\pi_1, \ldots, \pi_p\}$ is an (unknown) optimal (with respect to the edge cut) partition, then this BQP computes an optimal partition.*

Proof. We solve for each `AssignPartition` operation the linear systems $\mathbf{L} w_p = d_p$, where each d_p is the original drain vector without integration of β_p, $1 \le p \le k$. Performing `ScaleBalance` results in the load vector $\beta_p w_p$. With the Remarks 2 and 3

and the proof of Lemma 3, it follows that the assignment process can be regarded as making $\binom{k}{2}$ comparisons per vertex, i. e., vertices are assigned according to their sign in the $\binom{k}{2}$ fused load vectors $w_{(p,p')} = \beta_p w_p - \beta_{p'} w_{p'}$. As a direct consequence of the results above, for suitable β_p these load vectors $w_{(p,p')}$ correspond to a relaxed optimal solution of the BQP (2), (3). As shown before, this BQP would find the solution with minimum edge cut given an optimal placement of the center nodes. □

3.3 `ComputeCenters` Maximizes Constraint Contribution

Recall that the iteration of BUBBLE-FOS/C with its alternating calls to `AssignPartition` and `ComputeCenters` maximizes the potential function F (see the beginning of Section 3). Insofar it is interesting to find out if a similar optimization property holds when `ComputeCenters` is described as the relaxation of a cut-minimizing BQP. Note that in the case of `ComputeCenters` we are given a fixed partition and need to return one center vertex for each part.

Compared to our derivation in Section 3.2, the drain vector for part π_p is not $d_p^{(A)}$ any more, but $d_p^{(C)}$ (C for centers). This change reflects that the total drain is not given to one center vertex, but shared among all vertices of the part under consideration. Moreover, the scale values β_p are not needed any more, i. e., they can be set to 1 here. Consequently, $[d_p^{(C)}]_v = \delta\beta_p(n/|\pi_p| - 1)$ if $v \in \pi_p$ and $[d_p^{(C)}]_v = -\beta_p\delta$ if $v \notin \pi_p$.

Remark 4. To establish a BQP for `ComputeCenters` given the input partition Π, we simply replace all occurrences of $d^{(A)}$ by $d^{(C)}$ in Equation (3), eliminate the unnecessary β_p, and use the indicator vectors $x_{(p)}$ here:

$$x_{(p)}^T d_p^{(C)} = \delta(n - |\pi_p|) \;\forall 1 \le p \le k\,. \tag{5}$$

As shown below, the modified constraints ensure that all vertices stay in their part. This is important because the operation `ComputeCenters` is not supposed to change the partition. In particular, the computed centers must come from different parts.

Lemma 4. *The constraints in Equation (5) ensure that the centers $Z = \{z_1, \ldots, z_k\}$ computed by the BQP (2), (5) are in pairwise different parts with respect to Π.*

Immediately the question arises how the computation of centers is supposed to minimize the edge cut. Indeed, the BQP formulation only computes an indicator vector that represents the input partition. Yet, the new centers do have an extremal property, the contribution to Constraint (5). Again, we relax the binary condition on $x_{(p)}$, i. e., let $x_{(p)} \in \mathbb{R}^n$. Since $d^{(C)}$ is constant for all vertices of the same part and $[x_{(p)}]_{z_p} = \operatorname{argmax}_{1 \le v \le n}[x_{(p)}]_v$:

Corollary 2. *Given a partition $\Pi = \{\pi_1, \ldots, \pi_k\}$, let `ComputeCenters` compute the vertices $Z = \{z_1, \ldots, z_k\}$ as new centers. The respective entry $[x_{(p)}]_{z_p} d_{z_p}^{(C)}$ contributes the highest value of all vertices in π_p to $x_{(p)}^T \cdot d_p^{(C)}$, $1 \le p \le k$.*

4 Connectedness Properties of BUBBLE-FOS/C

For some applications that use partitioning as an intermediate step (e. g., tracking particles in parallel), it is advantageous that the parts are connected, i. e., that they have exactly one connected component each. Experiments with graphs from finite element discretizations reveal that the subdomains computed by BUBBLE-FOS/C are (nearly always) connected if the algorithm is allowed to perform sufficiently many iterations. Unfortunately, there has been no theoretical evidence for this observation until now.

In this section we make a step towards gaining more knowledge about the connectedness properties of BUBBLE-FOS/C. Similar to Fiedler's classical result [5] about spectral bipartitioning (but by using a different proof approach), we state that at least one part in a partition $\{\pi_1, \pi_2\}$ computed by BUBBLE-FOS/C is connected.

Theorem 2. *If the graph $G = (V, E, \omega)$ is connected, then at least one of $k = 2$ parts computed by `AssignPartition` on G is connected.*

The proof relies on the fact that the diffusion loads increase monotonically on some path from a vertex to a center. Note that the results of this section as well as some missing auxiliary results are proved in the full version of this paper. Now we tighten the result for all connected *vertex-transitive* graphs (a graph is vertex-transitive if for any pair of distinct vertices there is an automorphism mapping one to the other [3]), where both parts are shown to be connected. Two well-known vertex-transitive classes are torus graphs and hypercubes, which are important network topologies.

Theorem 3. *Let $G = (V, E)$ be a connected vertex-transitive graph. Fix two arbitrary different vertices $z_1, z_2 \in V$. Let the operation `AssignPartition` divide V into the two subdomains $\pi_1 = \{u \in V \mid [w]_u^{z_1} \geq [w]_u^{z_2}\}$ and $\pi_2 = \{u \in V \mid [w]_u^{z_1} < [w]_u^{z_2}\}$. Then, π_1 and π_2 are each connected components in G.*

Proof. The random walk measure *hitting time* $H[u, v]$ between nodes u and v is the expected timestep in which a random walk starting in u visits v for the first time. By using [16, Thm. 1], we know that $\frac{1}{\alpha}([w]_u^v - [w]_v^v) = \delta(H[v, v] - H[u, v])$. First, we show that hitting times are symmetric on vertex-transitive graphs. For this we use that $[w]_u^u = [w]_v^v$, which follows from the fact that $[\mathbf{M}^t]_{v,v} = [\mathbf{M}^t]_{u,u}$ for all $u, v \in V$ and all $t \geq 0$ for an unweighted vertex-transitive graph G [1, p. 151]. Also, the symmetry $[w]_v^u = [w]_u^v$ holds for all $u, v \in V$ [16] and $H[v, v]$ is zero (definition of hitting times). Thus:

$$([w]_v^v = [w]_u^u) \wedge ([w]_v^u = [w]_u^v) \Rightarrow [w]_u^v - [w]_v^v = [w]_v^u - [w]_u^u \Rightarrow H[u, v] = H[v, u].$$

Assume now for the sake of contradiction that π_2 is not connected. In this case there exists a node-separator $T \subseteq \pi_1$ such that there are at least two components $A, B \subseteq \pi_2$ which are not connected by a path via nodes in π_2. Assume w. l. o. g. that $z_2 \in B$. Then for each vertex $a \in A$ we obtain $[w]_a^{z_2} > [w]_a^{z_1} \Leftrightarrow [w]_a^{z_2} - [w]_{z_2}^{z_2} > [w]_a^{z_1} - [w]_{z_1}^{z_1} \Leftrightarrow H[z_2, z_2] - H[a, z_2] > H[z_1, z_1] - H[a, z_1] \Leftrightarrow H[a, z_1] > H[a, z_2]$.

In the same manner we have for each vertex $x \in T$ that $H[x, z_1] \leq H[x, z_2]$. Let $X^{(t)}$ be the random variable representing the node visited in time step t by a random walk, and let $\mathscr{F}_u(x)$ be the event that a fixed vertex x is the first vertex visited in T of a random walk starting from $u \in V$. Furthermore, denote by $\tau_a(T) := \min_{t \in \mathbb{N}}\{X^{(t)} \in$

$T \mid X^{(0)} = a\}$ and let $\tau_{a,T}(z_1) := \min_{t\in\mathbb{N}}\{X^{(t)} = z_1 \mid X^{(0)} = a\} - \tau_a(T)$. By using conditional expectations ($\mathbb{E}[Y] = \sum_z \mathbf{Pr}[Z=z]\cdot\mathbb{E}[Y\,|\,Z=z]$) [7], we obtain $H[a,z_1] = \mathbb{E}[\tau_a(z_1)] = \mathbb{E}[\tau_a(T)+\tau_{a,T}(z_1)] = \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(T)+\tau_{a,T}(z_1)\mid\mathscr{F}_a(x)]\big)$, which is transformed by using the linearity of conditional expectations into

$$\begin{aligned}H[a,z_1] &= \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(T)\mid\mathscr{F}_a(x)]+\mathbb{E}[\tau_{a,T}(z_1)\mid\mathscr{F}_a(x)]\big)\\ &= \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(x)\mid\mathscr{F}_a(x)]+\mathbb{E}[\tau_x(z_1)\mid\mathscr{F}_a(x)]\big)\\ &= \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(x)\mid\mathscr{F}_a(x)]+H[x,z_1]\big)\,.\end{aligned}$$

Exactly the same arguments yield $H[a,z_2] = \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(x)\mid\mathscr{F}_a(x)]+H[x,z_2]\big)$. Due to $H[x,z_1]\le H[x,z_2]$ for each $x\in T$, we finally obtain

$$\begin{aligned}H[a,z_1] &= \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(x)\mid\mathscr{F}_a(x)]+H[x,z_1]\big)\\ &\le \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\big(\mathbb{E}[\tau_a(x)\mid\mathscr{F}_a(x)]+H[x,z_2]\big) = H[a,z_2]\,,\end{aligned}$$

which is a contradiction to our assumption $H[a,z_1] > H[a,z_2]$. Therefore, the subdomain π_2 has to be connected. The remainder of the proof is analogous (switch π_1 and π_2). □

Generalizing this result to other graph classes will probably require new techniques, as the FOS/C load property $[w]_v^v = [w]_u^u$ does not hold any more. Also, our hitting time argument in the proof cannot be generalized to $k > 2$ in a straightforward manner since the vertex separator may contain vertices from more than one part.

5 Conclusions and Future Work

As explained in the introduction, diffusion-based graph partitioning has proved to be very successful in practice. Here we have provided the first substantial theoretical evidence for this success by proving that the assignment of vertices to parts in the partitioning algorithm BUBBLE-FOS/C is relaxed cut optimization. In this sense BUBBLE-FOS/C is similar to spectral partitioning, but does not require the (possibly numerically problematic) computation of eigenvectors. Moreover, we have proved two results on the connectedness of parts, which is a property that is important for some applications.

With these new tools at hand, we would like to consider the iterative nature of BUBBLE-FOS/C and explore the faster partitioning algorithm DIBAP in future work. DIBAP uses BUBBLE-FOS/C as one of two key components. It will be interesting to learn more about the interaction of these components, whose combination is responsible for obtaining high quality at reasonable speed. Eventually, it might also be possible to derive an approximation guarantee on BUBBLE-FOS/C's and DIBAP's quality from our relaxed BQP results, at least for certain graph classes. Since there are no such guarantees known for the popular KL heuristic, such a result would be a major step towards uniting theoretical and practical graph partitioning.

Acknowledgments. The author thanks T. Sauerwald, who contributed to the proof of Thm. 3, and C. Buchheim, R. Feldmann, and B. Monien for helpful discussions.

References

1. N. Alon and J. H. Spencer. *The Probabilistic Method*. J. Wiley & Sons, 2nd edition, 2000.
2. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming. Theory and Algorithms*. John Wiley, second edition, 1993.
3. N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 1993.
4. C. Chevalier and F. Pellegrini. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel Comput.*, 34(6-8):318–331, 2008.
5. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.
6. L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(3):469–475, 2006.
7. G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, 3rd edition, 2001.
8. B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.
9. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.
10. B. W. Kernighan and S. Lin. An efficient heuristic for partitioning graphs. *Bell Systems Technical Journal*, 49:291–308, 1970.
11. S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.
12. L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdös is Eighty*, 2:1–46, 1993.
13. M. Meila and J. Shi. A random walks view of spectral segmentation. In *Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.
14. H. Meyerhenke, B. Monien, and T. Sauerwald. A new diffusion-based multilevel algorithm for computing graph partitions. *Journal of Parallel and Distributed Computing*, 69(9):750–761, 2009. Best Paper Awards and Panel Summary: IPDPS 2008.
15. H. Meyerhenke, B. Monien, and S. Schamberger. Graph partitioning and disturbed diffusion. *Parallel Computing*, 35(10–11):544–569, 2009.
16. H. Meyerhenke and T. Sauerwald. Analyzing disturbed diffusion on networks. In *Proc. 17th Int. Symp. on Algorithms and Computation*, pages 429–438. Springer-Verlag, 2006.
17. F. Pellegrini. A parallelisable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries. In *Proc. 13th International Euro-Par Conference*, volume 4641 of *Lecture Notes in Computer Science*, pages 195–204. Springer-Verlag, 2007.
18. S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.
19. K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high performance scientific simulations. In *The Sourcebook of Parallel Computing*, pages 491–541. Morgan Kaufmann, 2003.
20. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
21. C. Walshaw. The graph partitioning archive. `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`, 2010. Last access: 1 Mar 2010.
22. H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *Proceedings of Advances in Neural Information Processing Systems 14 (NIPS'01)*, pages 1057–1064. MIT Press, 2001.

Appendix

A Optimization Criterion Results (Section 3)

A.1 Proof of Lemma 2

Proof. The constraints ensure that the center vertices do not change their parts in the computed partition and that the new parts have equal size.

Case 1. If all centers are contained within their corresponding parts, then all $\binom{k}{2}$ constraints can only be fulfilled by a balanced partition. To see this, consider two arbitrary parts π_p and $\pi_{p'}$ ($1 \leq p, p' \leq k$):

$$\begin{aligned} y_{(p,p')}^T d_{(p,p')}^{(A)} &= (x_{(p)}^T - x_{(p')}^T)(d_p^{(A)} - d_{p'}^{(A)}) = (\sum_{v \in \pi_p} [d_p^{(A)} - d_{p'}^{(A)}]_v + \sum_{v \in \pi_{p'}} [d_{p'}^{(A)} - d_p^{(A)}]_v) \\ &= \delta(\beta_p(n-1-|\pi_p|+1) + \beta_{p'}(n-1-|\pi_{p'}|+1) + \beta_p|\pi_{p'}| + \beta_{p'}|\pi_p|) \\ &= \delta(n(\beta_p + \beta_{p'}) + (\beta_{p'} - \beta_p)(|\pi_p| - |\pi_{p'}|)). \end{aligned}$$

This equation can only fulfill the constraint if $(\beta_{p'} - \beta_p)(|\pi_p| - |\pi_{p'}|) = 0$. Hence, either $\beta_{p'} = \beta_p$ or $|\pi_p| = |\pi_{p'}|$. Since the former is excluded in the initial choice (all β_i are pairwise different), we have $|\pi_p| = |\pi_{p'}|$. We have chosen $|\pi_p|$ and $|\pi_{p'}|$ arbitrarily, so that this case is completed.

Otherwise there exists a p with $z_p \notin \pi_p$ and a p' with $z_p \in \pi_{p'}$. We need to distinguish a few more cases to show that the constraint corresponding to the pair (p, p') cannot be fulfilled.

Case 2. $z_p \in \pi_{p'}$ and $z_{p'} \in \pi_p$: This case would mean that both centers change their parts. Substituting $[y]_{z_p}$ by -1 and $[y]_{z_{p'}}$ by 1 as well as some rearranging yields

$$y_{(p,p')}^T d_{(p,p')}^{(A)} = \delta \left(-n(\beta_p + \beta_{p'}) - \sum_{j \neq z_p, z_{p'}} (\beta_p - \beta_{p'})[y_{(p,p')}]_j \right).$$

The expression above can only fulfill the constraint if $\sum_{j \neq z_p, z_{p'}} (\beta_p - \beta_{p'})[y_{(p,p')}]_j = -2n(\beta_p + \beta_{p'})$, which is impossible given $\beta_p, \beta_{p'} > 0$:

$$\sum_{j \neq z_p, z_{p'}} (\beta_p - \beta_{p'})[y_{(p,p')}]_j \geq -(n-2)(\beta_p - \beta_{p'}) > -(n-2)(\beta_p + \beta_{p'}) > -2n(\beta_p + \beta_{p'}).$$

Case 3. $z_p \in \pi_p$ and $z_{p'} \in \pi_p$: Substituting both $[y]_{z_p}$ and $[y]_{z_{p'}}$ by 1 yields

$$y_{(p,p')}^T d_{(p,p')}^{(A)} = \delta \left((n-2)(\beta_p - \beta_{p'}) - \sum_{j \neq z_p, z_{p'}} (\beta_p - \beta_{p'})[y_{(p,p')}]_j \right).$$

The value we need to achieve by the sum over all $j \neq z_p, z_{p'}$ is

$$-(n(\beta_p + \beta_{p'}) - (n-2)(\beta_p - \beta_{p'})) = -2((n-1)\beta_{p'} + \beta_p)$$

to meet the constraint. Let us assume for now that $\beta_p > \beta_{p'}$. The largest absolute contribution of the sum is obtained by putting all remaining $n-2$ vertices into $\pi_{p'}$. Then, the sum evaluates to $-(n-2)(\beta_p - \beta_{p'})$. Finally, the constraint is *not* fulfilled whenever

$$\begin{aligned} -(n-2)(\beta_p - \beta_{p'}) &\neq -2((n-1)\beta_{p'} + \beta_p) \\ \Leftrightarrow (n-4)\beta_p &\neq (3n-4)\beta_{p'}. \end{aligned}$$

Thus, a choice of $\frac{\beta_p}{\beta_{p'}} < 3 = \frac{3n-12}{n-4} < \frac{3n-4}{n-4}$ avoids the constraint to be fulfilled with $z_p \in \pi_p, z_{p'} \in \pi_p$ and $\beta_p > \beta_{p'}$. The remaining part with $\beta_{p'} > \beta_p$ is analogous: We need to put all remaining vertices into π_p and the sum evaluates to $(n-2)(\beta_p - \beta_{p'})$. The constraint is not fulfilled whenever $(n-2)(\beta_p - \beta_{p'}) \neq -2((n-1)\beta_{p'} + \beta_p) \Leftrightarrow n\beta_1 \neq -n\beta_2$. Since $0 < \beta_p < \beta_{p'}$, a different sign is not possible.

Case 4. $z_p \in \pi_{p'}$ and $z_{p'} \notin \pi_p \cup \pi_{p'}$: Since both centers are not in their respective part, the respective scalar products $y^T_{(p,p')} d^{(A)}_{(p,p')}$ evaluate to simple expressions:

$$\begin{aligned} y^T_{(p,p')} d^{(A)}_{(p,p')} &= (x^T_{(p)} - x^T_{(p')})(d^{(A)}_p - d^{(A)}_{p'}) = \delta(-\beta_p \cdot |\pi_p| - \beta_{p'} \cdot |\pi_{p'}| - \beta_p \cdot |\pi_{p'}| + \beta_{p'} \cdot |\pi_p|) \\ &= \delta((\beta_p - \beta_{p'})(|\pi_{p'}| - |\pi_p|)) \end{aligned}$$

Fulfilling the constraint would mean $\delta((\beta_p - \beta_{p'})(|\pi_{p'}| - |\pi_p|)) = n(\beta_p + \beta_{p'}) \Leftrightarrow |\pi_{p'}| - |\pi_p| = n \cdot \frac{\beta_p + \beta_{p'}}{\beta_p - \beta_{p'}}$. The choice of β_p and $\beta_{p'}$ results in $\big|\,|\pi_{p'}| - |\pi_p|\,\big| = \left| n \cdot \frac{\beta_p + \beta_{p'}}{\beta_p - \beta_{p'}} \right| > n$, which is not possible because $\big|\,|\pi_{p'}| - |\pi_p|\,\big|$ cannot exceed n.

All other possible cases can be reduced to the ones above. In particular, if more than two centers are in one part, at least one of the respective constraints is violated in a very similar way as shown above. □

A.2 Proof Remainder for Lemma 3

To compute the missing expressions, some rearrangements are necessary: $\overline{y}^T_{(p,p')} d^{(A)}_{(p,p')} = n\delta(\beta_p + \beta_{p'}) \Rightarrow \Lambda_{(p,p')} \frac{\overline{y}_{(p,p')}}{\Lambda_{(p,p')}} d^{(A)}_{(p,p')} = n\delta(\beta_p + \beta_{p'}) \Rightarrow \Lambda_{(p,p')} = \frac{n\delta(\beta_p + \beta_{p'})}{z^T_{(p,p')} \cdot d^{(A)}_{(p,p')}}$ and finally $\overline{y}_{(p,p')} = \Lambda_{(p,p')} z_{(p,p')} = \frac{n\delta(\beta_p + \beta_{p'})}{z^T_{(p,p')} \cdot d^{(A)}_{(p,p')}} \cdot z_{(p,p')}$. □

A.3 Proof of Lemma 4

Proof. Let the part p be chosen arbitrarily with $1 \leq p \leq k$. Recall that scale balancing is not required, so that $\beta_p = \beta_{p'} = 1$ here. If π_p remains unchanged as desired, then $x^T_{(p)} d^{(C)}_p$ evaluates to

$$x^T_{(p)} d^{(C)}_p = \sum_{j \in \pi_p} [d^{(C)}_p]_j = |\pi_p|(\delta(\frac{n}{|\pi_p|} - 1)) = \delta(n - |\pi_p|)\,,$$

which fulfills the constraint. Assume now for the sake of contradiction that π_p does not remain unchanged. We categorize the vertices into the sets π_{pp} and $\pi_{pp'}$ such that

the vertices in the set π_{pp} are in the input part π_p before and after `ComputeCenters`, and such that the vertices in $\pi_{p'p}$ have not been in π_p before `ComputeCenters`, but are so afterwards. We need to show that $\pi_{p'p}$ is in fact empty and that $\pi_{pp} = \pi_p$. With $s_{pp} := |\pi_{pp}|$ and $s_{p'p} := |\pi_{p'p}|$, one can rewrite $x_{(p)}^T d_p^{(C)}$ as

$$\begin{aligned} x_{(p)}^T d_p^{(C)} &= \sum_{v \in \pi_{pp}} [d_p^{(C)}]_v + \sum_{v \in \pi_{p'p}} [d_p^{(C)}]_v \\ &= \delta(s_{pp} \cdot \frac{n}{|\pi_p|} - s_{pp} - s_{p'p}) . \end{aligned}$$

To fulfill the constraint, we must obtain $s_{pp} \cdot \frac{n}{|\pi_p|} - s_{pp} - s_{p'p} = n - |\pi_p|$. However, regardless of the actual size of $s_{p'p}$, $s_{pp} \cdot \frac{n}{|\pi_p|} - s_{pp} - s_{p'p} \leq s_{pp}(\frac{n}{|\pi_p|} - 1) = \frac{s_{pp}(n-|\pi_p|)}{|\pi_p|}$. If $s_{pp} < |\pi_p|$, i. e., some nodes would leave the current input part, then $\frac{s_{pp}(n-|\pi_p|)}{|\pi_p|} < n - |\pi_p|$, so that the constraint cannot be fulfilled. If $s_{pp} = |\pi_p|$ and $s_{p'p} > 0$, then similarly $s_{pp} \cdot \frac{n}{|\pi_p|} - s_{pp} - s_{p'p} < s_{pp}(\frac{n}{|\pi_p|} - 1) = \frac{s_{pp}(n-|\pi_p|)}{|\pi_p|} = n - |\pi_p|$. Hence, the constraints can only be fulfilled if the input partition Π remains unchanged. □

B Connectedness Results (Section 4)

B.1 Proof of Theorem 2

For the proof we need a few auxiliary results first.

Lemma 5. *(due to Hu and Blake[2] and Diekmann et al.[3]) Let $\tilde{\mathbf{A}} = \mathbf{AF}$ be the (edge-weighted) node-edge incidence matrix of $G = (V, E, \omega)$ and let $\mathbf{F}$ be an $m \times m$ diagonal matrix with $[\mathbf{F}]_{i,i} = \sqrt{\omega_i}$. The solution of the ℓ_2-minimization problem minimize $\|\mathbf{F}^{-1} f\|_2$ over all f with $\tilde{\mathbf{A}} f = d$ is given by $f = \tilde{\mathbf{A}}^T z$, where $\mathbf{L}z = d$ with $d, z \in \mathbb{R}^n$, provided that $d \perp \overline{w}$. Using this minimization problem, it can be shown that the FOS migrating flow f^* is the unique $\|\cdot\|_2$-minimal balancing flow.*

Remark 5. [15] The load differences $f = \mathbf{A}^T w^{(\infty)}$ in the converged state of the disturbed diffusion scheme FOS/C equal the $\|\cdot\|_2$-minimal flow f that balances the load vector d/α, sending from the vertices in S (sources) the respective load amount δ to every vertex in the graph (sinks): $\mathbf{A}f = \mathbf{A}\mathbf{A}^T w^{(\infty)} = \mathbf{L}w^{(\infty)} = d/\alpha$.

Proposition 1. *[16] Let the graph $G = (V, E, \omega)$ and the load vector w of an FOS/C procedure with source set S be given. Then for each vertex $v \in V$ there is a path $(v = v_0, v_1, \ldots, v_l = s)$ with $s \in S$ and $\{v_i, v_{i+1}\} \in E$ such that $w_{v_i} < w_{v_{i+1}}, 0 \leq i < l$.*

[2] Y. F. Hu and R. F. Blake, An Improved Diffusion Algorithm for Dynamic Load Balancing, Parallel Computing (1999), p. 417–444.

[3] R. Diekmann and A. Frommer and B. Monien, Efficient schemes for nearest neighbor load balancing, Parallel Computing (1999), p. 789–812.

Proof. Assume that the claim is false, so that no such monotonously increasing path exists. Moreover, recall that the convergence state of FOS/C is equivalent to a flow problem where all vertices $v \in V \setminus S$ receive a load amount of δ. Now, let j be the smallest index such that the monotonous path from v to $s \in S$ stops in $v_j \notin S$ because $w_{v_j} \geq w_{v'} \forall (v_j, v') \in E$. This means that v_j is a local maximum w. r. t. its load, so that the flow on its incident edges directs from v_j away. Hence, v_j would not receive any load. As all non-source vertices must receive a load amount of δ, our assumption is wrong and the claim true. □

Lemma 6. *Consider the load vector w in the convergence state of FOS/C and the corresponding flow problem described in Remark 5. Then, the node v with maximum load value in w belongs to the set of source nodes S.*

Proof. Assume the opposite, i.e., $v \notin S$. Since v has the highest load, no flow is directed towards v because the flow on an edge is the load difference of its incident nodes. Hence, in the flow problem equivalent to FOS/C v does not receive any load. This is a contradiction to the initial setting because all nodes not in S receive a load amount of δ by definition. □

Using this lemma, we can construct load-increasing paths from each vertex to its center. The nodes on these paths all belong to the same part, which yields the following connectedness result for general graphs.

Proof of Theorem 2

Proof. In a single-source `AssignPartition` operation the source set S of diffusion system p, $1 \leq p \leq 2$, contains only the center vertex z_p. Assume for now that $\beta_1 > \beta_2$. Then all entries of the drain vector $d^{(A)}$ are negative except for the entry corresponding to z_1. This means according to Remark 5 that all non-center vertices act as load-consuming sinks. Hence, using the arguments of Proposition 1 and Lemma 6, there must be a path $P = \langle v = v_1, v_2, \ldots, v_l = z_1 \rangle$ from every vertex $v \in \pi_1$ to z_1 on which the load increases, i. e., $[w]_{v_i} < [w]_{v_{i+1}}$ for $1 \leq i < l$. All vertices on this path have a positive load in the fused load vector and belong to π_1, so that π_1 is connected. If $\beta_1 < \beta_2$, the same argument applies analogously. We only need to change the signs, direction of inequalities, and local maxima become local minima. □

B.2 Proof of Theorem 3

For the proof (remainder), we need to establish a few auxiliary definitions and lemmas used therein.

Definition 4. *[3, p. 115] Given a graph $G = (V,E)$, a permutation π of V is an* automorphism *of G if*

$$\{u,v\} \in E \Leftrightarrow \{\pi(u), \pi(v)\} \in E, \forall u,v \in V.$$

Definition 5. *[3] A graph $G = (V,E)$ is* vertex-transitive *if for any two distinct vertices of V there is an automorphism mapping one to the other.*

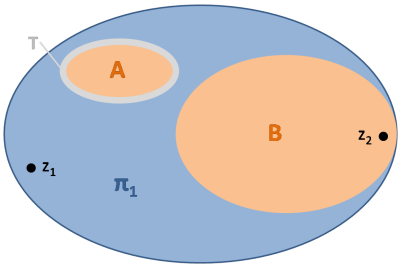

Fig. 2. Sketch of the situation assumed in Theorem 3.

Definition 6. *Let $X_u^{(t)}$ be the random variable representing the node visited in timestep t by a random walk induced by the diffusion matrix $\mathbf{M}$ starting in u in timestep 0 and let τ_u be defined as $\tau_u := \min\{t \geq 0 : X_u^{(t)} = s\}$ for any $u \in V$. Then, the (expected)* hitting time *H is defined as $H[u,s] := \mathbb{E}[\tau_u]$.*

FOS/C loads are related to hitting times by the following lemma:

Lemma 7. *[16] In the FOS/C steady state described by the load vector w in $\mathbf{L}w = d$ it holds for two nodes $u,v \in V$ not necessarily distinct from a source $s \in V$: $[w^{(\infty)}]_u^s - [w^{(\infty)}]_v^s = \frac{1}{\alpha}([w]_u^s - [w]_v^s) = \delta(H[v,s] - H[u,s])$.*

The following lemma is crucial for the proof of the main theorem:

Lemma 8. *For all unweighted vertex-transitive graphs $G = (V,E)$ and all $u,v \in V$ it holds that $[w]_u^u = [w]_v^v$.*

Proof. As noted by Alon and Spencer[4], we have $[\mathbf{M}^t]_{v,v} = [\mathbf{M}^t]_{u,u}$ for all $u,v \in V$ and all $t \geq 0$ of an unweighted vertex-transitive graph G. Since $[w]_v^u = \lim_{t\to\infty}([\mathbf{M}^t w^{(0)}]_v^u + n\delta(\sum_{l=0}^{t-1}[\mathbf{M}^l]_{v,u}) - t\delta)$ [16] and $[\mathbf{M}^t w^{(0)}]_v^u$ converges towards the balanced distribution irrespective of u and v, we get

$$[w]_u^u - [w]_v^v = n\delta\left(\lim_{t\to\infty}\sum_{l=0}^{t}\mathbf{M}_{u,u}^l - \mathbf{M}_{v,v}^l\right) = 0$$

□

Proof Remainder for Theorem 3

Proof. The proof that π_1 is always connected is done in the same way. Assume the converse and let A and B be two disconnected components of π_1 with a node separator $T \subseteq \pi_2$ such that $z_1 \in B$. For a vertex $a \in A$ we have $H[a,z_1] \leq H[a,z_2]$ and for every vertex $x \in T$ it holds that $H[x,z_1] > H[x,z_2]$. Consequently,

$$\begin{aligned} H[a,z_2] &= \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\left(\mathbb{E}[\tau_a(x) \mid \mathscr{F}_a(x)] + H[x,z_2]\right) \\ &< \sum_{x\in T}\mathbf{Pr}[\mathscr{F}_a(x)]\cdot\left(\mathbb{E}[\tau_a(x) \mid \mathscr{F}_a(x)] + H[x,z_1]\right) \\ &= H[a,z_1], \text{ which contradicts our assumption } H[a,z_1] \leq H[a,z_2]. \ \square \end{aligned}$$

[4] N. Alon and J. H. Spencer, The Probabilistic Method, J. Wiley & Sons (2000), p. 151.