Noname manuscript No.
(will be inserted by the editor)

Beyond Good Partition Shapes: An Analysis of Diffusive Graph Partitioning

Henning Meyerhenke · Thomas Sauerwald

the date of receipt and acceptance should be inserted later

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 a practical successful graph partitioner (Meyerhenke et al., J. Parallel and Distrib. Computing, 69(9):750–761, 2009).

We begin by studying the disturbed diffusion scheme FOS/C, which computes the similarity measure used in Bubble-FOS/C and is therefore the most crucial component. By relating FOS/C to random walks, we obtain precise characterizations of the behavior of FOS/C on tori and hypercubes. Besides leading to new knowledge on FOS/C (and therefore also on Bubble-FOS/C), these characterizations have been recently used for the analysis of load balancing algorithms (Berenbrink et al., SODA'11).

We then regard Bubble-FOS/C, which has been shown in previous experiments to produce solutions with good partition shapes and other

Parts of this paper have been published in a preliminary form in the proceedings of the 17th and 21st International Symposium on Algorithms and Computation (ISAAC 2006 and ISAAC 2010), [35,32].
This work was partially supported by German Research Foundation (DFG) Research Training Group GK-693 of the Paderborn Institute for Scientific Computation and by DFG Priority Programme 1307 Algorithm Engineering. H. Meyerhenke was also partially supported by the CASS-MT Center led by Pacific Northwest National Laboratory and NSF Grant CNS-0708307. Parts of this work were performed while the authors were affiliated with the Department of Computer Science, University of Paderborn, Germany, and while H. M. was affiliated with Georgia Institute of Technology, Atlanta, Georgia, USA.

H. Meyerhenke
Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany
Tel.: +49-721-608-41876. E-mail: meyerhenke@kit.edu

T. Sauerwald
Max-Planck Institute for Computer Science, Department 1: Algorithms & Complexity, Saarbrücken, Germany
Tel.: +49 681 9325 833, Fax: +49 681 9325 199. E-mail: sauerwal@mpi-inf.mpg.de

favorable properties. In this paper we prove that it computes a relaxed solution to an edge cut minimizing binary quadratic program (BQP). This result provides the first substantial theoretical insight why Bubble-FOS/C yields good experimental results in terms of graph partitioning metrics. Moreover, we show that in bisections computed by Bubble-FOS/C, at least one of the two parts is connected. Using the aforementioned relation between FOS/C and random walks, we prove that in vertex-transitive graphs both parts must be connected components.

Keywords Diffusive graph partitioning · relaxed cut optimization · disturbed diffusion · random walks.

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 vertices in different subsets. Among many others, the applications of this $\mathcal{NP}$-hard problem include load balancing in numerical simulations [42] and image segmentation [43].

Despite recent approximation algorithms, simpler heuristics are preferred in practice, many of which can be found in surveys on graph partitioning [42] and graph clustering [41]. Spectral algorithms have been widely used [22]; they are global optimizers based on graph eigenvectors. For computational efficiency reasons they have been mostly superseded by local improvement algorithms. Integrated into a multilevel framework, local optimizers such as Kernighan-Lin (KL) [26] can be found in several popular partitioning libraries [8,23]. Unfortunately, theoretical quality guarantees are not known for KL. Another class of improvement strategies comprises diffusion-based methods [33,37]. While they are slower than KL, diffusive methods often yield a better quality, also when repartitioning dynamic graphs [33,34].

1.1 Motivation

The hybrid algorithm DibaP is a multilevel combination of the diffusive algorithms Bubble-FOS/C [34] and TruncCons. Particularly on graphs arising

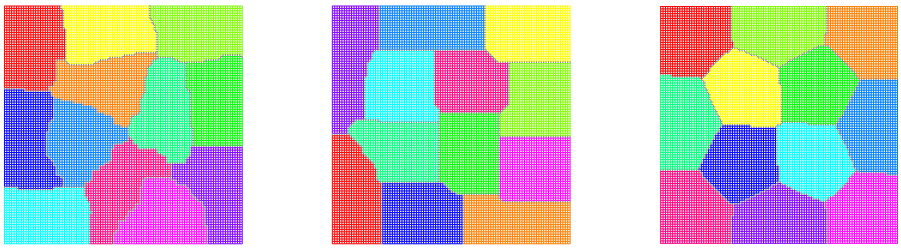

Fig. 1 Dividing a grid into 12 partitions using the software libraries KMeTiS, Jostle, and the shape-optimizing algorithm Bubble-FOS/C (from left to right).

in numerical simulations, DibaP is very successful [33]. For example, it has computed for six of the eight largest graphs of a popular benchmark set [49] a significant number (more than 80 out of 144 when DibaP was developed in 2007, currently this number is 15) 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 [28], 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, see Figure 1 as one example. This comes along with a small number of boundary vertices (i. e., vertices with at least one neighbor in a different part) and a small edge cut [34]. However, apart from intuition (see Section 2.3), there has been no convincing theoretical explanation why Bubble-FOS/C and ultimately DibaP produce such good partitions.

1.2 Contribution

We begin our study with FOS/C, which is used within Bubble-FOS/C to determine how "similar" two vertices or vertex subsets are. First we present in Section 3.1 results that relate FOS/C to minimal flows and show in Section 3.2 that FOS/C represents indeed a *similarity measure* which satisfies two natural properties. Section 3.3 contains our results that relate FOS/C to random walks. Using these results, we prove for tori and hypercubes that the similarity measure computed by FOS/C satisfies a *monotonicity property* similar to the graph distance (Section 3.4). In Section 3.5, we finally outline how Berenbrink et al. [6] have used the results from Section 3.4 for the analysis of load balancing algorithms.

We then continue to answer several open questions regarding diffusion-based partitioning with Bubble-FOS/C. In Section 4.1 we prove our main result 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 vertices 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.

The achievements in Section 4.2 concern the *connectedness* of the parts in a bipartition ($k = 2$) computed by Bubble-FOS/C. We prove a result known for spectral partitioning [16], 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 and hypercube) we use the random walk measure *hitting times* and conditional expectations to show that *both* parts computed by `AssignPartition`, the main partitioning algorithm of Bubble-FOS/C, are always connected.

1.3 Notation

We consider 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* $\omega : E \to \mathbb{R}_{>0}$. Some results apply to unweighted graphs $G = (V, E)$ only, which will be clear from the context. 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 $[\mathbf{M}]_{u,v}$ or $\mathbf{M}_{u,v}$. 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 $[w_i]_v$. It will be clear from the context if the short-hand notation w_v refers to the v-th entry of vector w or the v-th vector in a series. The symmetric positive semidefinite *Laplace matrix* matrix $\mathbf{L}$ of G [7, 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.

Definition 1 Let $\mathbf{A}$ be the vertex-edge incidence matrix of dimension $n \times m$ for the graph $G = (V, E, \omega)$, and let $\mathbf{F}$ be an $m \times m$ diagonal matrix with $[\mathbf{F}]_{e,e} = \sqrt{\omega_e}$. Each column of $\mathbf{A}$ corresponds to an edge, each row to a node. Note that each column has exactly two nonzero entries ($+1$ and -1). In the column of edge $e = \{u, v\}$, the nonzero entries appear in the rows corresponding to the incident nodes u and v.

Observe that the weighted Laplace matrix of G can be written as $\mathbf{L} = \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T$, where $\tilde{\mathbf{A}} = \mathbf{AF}$. In case of undirected graphs, the signs of the nonzero entries of $\tilde{\mathbf{A}}$ define an implicit (and arbitrary) direction of the edges.

2 Diffusive Partitioning with Bubble-FOS/C

Diffusive processes can be used to model a large variety of important transport phenomena arising in such diverse areas as heat flow, particle motion, and the spread of diseases. In computer science one has studied diffusion in graphs as one of the major tools for balancing the load in parallel computations [50]. In such a discrete setting, diffusion is a local iterative process which exchanges splittable load entities between neighboring vertices. In connected non-bipartite graphs, the first order diffusion scheme FOS defined below converges to the balanced state, where all vertices have the same load amount [10].

Definition 2 (cf. [10]) Given a graph $G = (V, E, \omega)$, an initial load vector $w^{(0)}$ and a suitable diffusion parameter α, the first order diffusion scheme FOS performs the following operations in each iteration $t \geq 1$:

$$x^{(t-1)}_{e=(u,v)} = \alpha\omega_e(w^{(t-1)}_u - w^{(t-1)}_v), \qquad w^{(t)}_u = w^{(t-1)}_u - \sum_{e=(u,v)} x^{(t-1)}_e \,.$$

Note that edges are viewed as directed in the definition of x. This is necessary to give the load exchange a unique direction, but it is no limitation. The edge orientation in x should follow the artificial orientation in $\tilde{\mathbf{A}}$ (otherwise the sign of the load exchange needs to be changed).

The load vector $w^{(t)}$ stores the load on each vertex in iteration t, whereas the value $x^{(t)}_e$ denotes the load exchange or *flow* computed by FOS on edge e in iteration t. The *FOS migrating flow* f^* is the sum of all load exchanges via the edges of G during the FOS iteration: $f^* := \sum_{t=0}^{\infty} x^{(t)}$. Diffusion requires only local load exchanges between neighboring vertices, which is important in case vertices represent processors in a distributed system. Equally important, the migrating flow computed by diffusion is $\|\cdot\|_2$-minimal.

Lemma 1 *(cf. [13]) A function $f : E \to \mathbb{R}$ is called* balancing *for $w^{(0)}$ if and only if $\tilde{\mathbf{A}}f = w^{(0)} - \overline{w}$, where $\overline{w}$ is the balanced load vector $(1/n \cdot \sum_{v\in V}[w^{(0)}]_v) \cdot \mathbf{1}$. The solution of the ℓ_2-minimization problem*

$$\textit{minimize } \|\tilde{f}\|_2 \textit{ over all } \tilde{f} \textit{ with } \tilde{\mathbf{A}}\tilde{f} = d$$

is given by $\tilde{f}^* = \tilde{\mathbf{A}}^T z$, where $\mathbf{L}z = d$ with $d, z \in \mathbb{R}^n$, provided that $d \perp \mathbf{1}$. Using this minimization problem, it can be shown that the FOS migrating flow is the unique $\|\cdot\|_2$-minimal balancing flow $\tilde{f}^*$.

It is easy to check that $\tilde{f}^*_{e=(u,v)} = \sqrt{\omega_e}(z_u - z_v)$. Since z can be interpreted as load vector, we may regard the flow over edge $e = (u, v)$ in the steady state as (scaled) load difference between its incident vertices. The FOS load update can be written in matrix-vector notation as $w^{(t)} = \mathbf{M}w^{(t-1)}$, where

$\mathbf{M} = \mathbf{I} - \alpha\mathbf{L}$ is the *diffusion matrix* of G with

$$[\mathbf{M}]_{i,j} = \begin{cases} \alpha\omega(\{i,j\}) & \text{if } \{i,j\} \in E \\ 1 - \sum_{k \neq i} [\mathbf{M}]_{i,k} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathbf{M}$ is stochastic, it can also be interpreted as the transition matrix of a *lazy random walk* [29]. A random walk on a graph $G = (V, E, \omega)$ is a discrete time stochastic process defined as follows: Starting on an initial vertex, a random walk performs the following in each iteration. First, it chooses one of the neighbors of the current vertex v randomly (where the probabilities are proportional to the edge weights). Then, it proceeds to the neighbor just chosen to start the next iteration. In some models, also referred to as lazy random walks, there is a positive probability for staying on the current vertex (this corresponds to diffusion matrices whose diagonal entries are non-zero).

Intuitively, a random walk is likely to stay a long time in a dense graph region which has only a few outgoing edges. There exist many graph clustering/partitioning techniques exploiting this notion (see [41]).

For graph partitioning, diffusive algorithms and similarity measures are used to compute well-shaped partitions [34,37]. These works exploit that diffusive processes send load faster into densely connected graph regions. As we will see in more detail later on, such a preference to dense regions corresponds to the intuition that random walks are likely to be trapped for a long time in dense graph regions.

2.1 The Disturbed Diffusive Similarity Measure FOS/C

The diffusive partitioning algorithm Bubble-FOS/C, which is our main subject of investigation, is composed of the Bubble framework (described later) and the disturbed diffusive algorithm FOS/C [34]. FOS/C acts as a diffusive similarity measure within Bubble. We call a diffusion scheme *disturbed* if its steady state does not result in the balanced load distribution, where each vertex has the same amount of load.

Since the first order diffusion scheme FOS converges towards exactly this balanced distribution, its steady state does not convey information about the graph structure. FOS/C (first order scheme with constant drain) therefore introduces a drain-based disturbance into FOS. Within each time step, every vertex loses a load amount of δ, and the total drain is shared evenly among a specified set of source vertices S (cf. Figure 2(a)).

We will see that, with this disturbance, FOS/C reaches an unbalanced steady state, in which the load vector w represents similarities of vertices

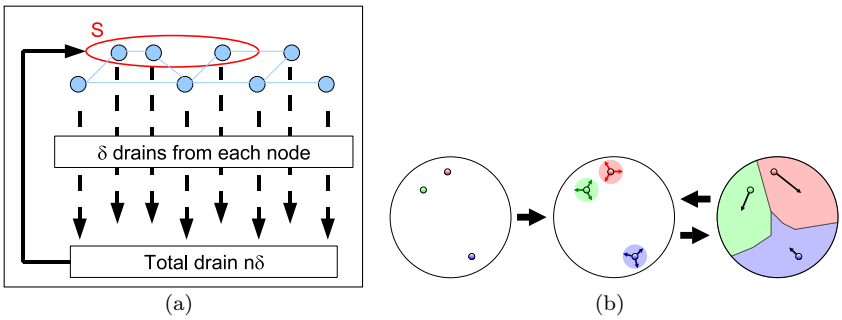

Fig. 2 (a) Sketch of the drain concept of FOS/C in an unweighted graph. (b) Sketch of the main Bubble framework operations: Determine initial centers for each part (left), assign each vertex to the part of the nearest center (middle), and compute new centers (right).

reflecting whether vertices are *well-connected*, i. e., connected by many paths of short length.

Definition 3 (FOS/C) [34] Given a connected and undirected graph $G = (V, E, \omega)$ free of self-loops, a set of source vertices $\emptyset \neq S \subset V$, 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 edge-weighted FOS/C diffusion scheme performs the following operations in each iteration $t > 0$:

$$x_{e=(u,v)}^{(t-1)} = \alpha\omega_e(w_u^{(t-1)} - w_v^{(t-1)})\,, \tag{1}$$

$$w_u^{(t)} = w_u^{(t-1)} + d_u - \sum_{e=(u,v)} x_e^{(t-1)}\,. \tag{2}$$

The FOS/C iteration can be written in matrix-vector notation as $w^{(t)} = \mathbf{M}w^{(t-1)} + d$.

Lemma 2 *[34] For any $d \perp \mathbf{1} = (1, \ldots, 1)^T$ (i. e., $\langle d, (1, \ldots, 1)^T \rangle = 0$), the FOS/C iteration reaches a steady state $w^{(\infty)}$, which can be computed by solving the linear system $\mathbf{L}w^{(\infty)} = \frac{d}{\alpha}$.*

Remark 1 Regarding the steady state of FOS/C, we use $w^{(\infty)}$ if we refer to the actual FOS/C iteration as in Eq. (2). More often we will use the simpler notation $w = \alpha w^{(\infty)}$ as in $\mathbf{L}w = d$, emphasizing the typical solution process with linear solvers.

Remark 2 The solutions of linear systems of the form $\mathbf{L}w = d$ are not unique, because $\mathbf{L}(w + \gamma\mathbf{1}) = \mathbf{L}w + \gamma\mathbf{L}\mathbf{1} = \mathbf{L}w = d$ for any $\gamma \in \mathbb{R}$. If only the relative values in an FOS/C vector are of concern (e. g., if FOS/C vectors are

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

compared to each other), the solution w can be made unique by requiring a normalization, e. g., $\sum_{v \in V} [w]_v = n$.

Definition 4 If $|S| = 1$ ($|S| > 1$), we call the steady state of an FOS/C iteration or its corresponding linear system a *single-source (multiple-source) FOS/C procedure.*

If the source vertex is not clear from the context, we use $[w^{(t)}]_v^s$ or $[w]_v^s$ to denote the load on vertex v in time step t or in the steady state, respectively, of a single-source FOS/C procedure with vertex s as source.

We will show in Section 3 that the load values computed by FOS/C are rows of a similarity matrix. Hence, these values can be used to assess the similarity of graph vertices. Its tight connection to random walks makes FOS/C suitable for graph partitioning. In the next section we discuss how FOS/C is used as an important component in the graph partitioning process.

2.2 The Partitioning Algorithm Bubble-FOS/C

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 [28] 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 sketched in Figure 2(b) and outlined in Figure 3, where $\Pi = \{\pi_1, \ldots, \pi_k\}$ denotes the set of parts, $\Pi(v)$ the part of vertex v, and $Z = \{z_1, \ldots, z_k\}$ the set of the corresponding center vertices. First, the algorithm determines pairwise disjoint initial centers (line 1), which can be done arbitrarily. 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 (by solving the equivalent linear systems for efficiency, see Definition 4) per operation, single-source ones ($S_p = \{z_p\}$) for `AssignPartition` and multiple-source ($S_p = \pi_p$) ones for `ComputeCenters`. Note that, due to the disturbed diffusive approach, the new center vertices are usually no centers in a geometric or ordinary graph-distance sense. The loop is iterated until convergence is reached (convergence is guaranteed [33]) or, if time is important, a constant number of times. To ensure balanced parts, `AssignPartition` is followed by an operation called ScaleBalance (line 10), cf. Section 4.1.1 and [34].

```
Algorithm Bubble-FOS/C(G, k) → Π
01  Z = InitialCenters(G, k)   /* Arbitrary disjoint centers */
02  for t = 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 vertex 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  Π ← ScaleBalance(G, Π)
11  return Π
```

Fig. 3 Sketch of the main Bubble-FOS/C algorithm.

Within the partitioner DibaP one uses Bubble-FOS/C to compute solutions for smaller representations of the input graph with only a few thousand vertices and edges. This computation is reasonably fast and gives initial solutions that are often 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 Meyerhenke et al. [33] 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 approximately one million vertices and edges, DibaP requires less than a minute on standard single- or dual-core processors). One motivation for this work was that there had been no theoretical evidence beyond intuition so far why the important initial solutions produced by Bubble-FOS/C within DibaP are of high quality.

2.3 Related Work on Similarity Measures and Partitioning Techniques

One can view FOS/C as a means to determine the similarity of each vertex to the source set, e. g., the different center vertices, within Bubble-FOS/C. Random-walk based distance or similarity measures often reflect how well-connected two vertices are (cf. [40] and [14, p. 99f.]). This means that they are able to identify dense regions of the graph, where vertices are connected to each other by many paths of short length. This idea is used in several works for distance measures based on random walks and diffusion. They were mostly developed for clustering of point sets and graphs [36, 40, 45, 48, 51], image segmentation [30], and dimensionality reduction [9]. Their approaches rely on

expensive matrix operations, among others the computation of matrix powers [45,48], eigenvectors of a kernel matrix [9,30,36], or the pseudoinverse of the graph's Laplacian [40,51]. This mostly aims at providing a meaningful distance between every pair of vertices.

A complete matrix with a distance or similarity value for each pair of vertices is not necessary for Lloyd's algorithm, because distance or similarity computations are relative to the current centers. The sparse linear systems $\mathbf{L}w = d$ in Bubble-FOS/C can be solved by suitable solvers such as algebraic multigrid methods [47] in linear time, when implemented with care. Note that only a constant number of Bubble-FOS/C iterations are sufficient in practice. Thus, this approach is faster (unless distances between every pair of vertices are necessary in a different setting) than the related methods, which all require $\Omega(n^2)$ operations in the general case.

Meila and Shi [30] connect random walks to spectral partitioning. Spectral methods such as [43] solve relaxations of integer programs (IPs) that minimize the edge cut or the related ratio cut. They build on Fiedler's seminal work on spectral partitioning [16] 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. [52]. Their work is built upon by Dhillon et al. [11] to derive and analyze an iterative heuristic called kernel k-means. This algorithm optimizes the same objectives locally as spectral methods, but requires no eigenvector computations.

Using the concept of local partitioning, Andersen et al. [2] develop an approximation algorithm that outputs asymptotically balanced partitions. Their mechanism is based on the computation of approximate page rank vectors (APR's), which are based on random walks with teleportation (random surfer model). The mechanism starts from a specified vertex, and its random walk rationale is similar to the diffusion process FOS/C. Andersen and Peres [3] devise a faster algorithm similar in spirit. Their approach uses certain Markov chains called evolving set processes to find local sets with small conductance. The probabilistic algorithm obtains an asymptotic complexity of $(m + n\phi^{-1/2}) \cdot \mathcal{O}(\text{polylog}(n))$ and an approximation guarantee of $\mathcal{O}(\phi^{1/2} \log^{1/2} n)$, where ϕ is a target conductance, the ratio of edge cut and volume. Since both approaches do not guarantee exact balancing, their algorithmic results are more valuable for graph clustering than partitioning.

3 Analyzing the Vertex Similarity Measure FOS/C

In this section we analyze the disturbed diffusive mechanism FOS/C in more detail. After some fundamental results concerning flow properties used later on, we show that the load values computed by FOS/C are rows of a similarity

matrix. Hence, these values can be used to assess the similarity of graph vertices. Moreover, we explore FOS/C's connection to random walks. After that, we explore the monotonicity of FOS/C, an important property for a similarity measure, on torus and distance-transitive graphs. Finally, we briefly sketch how these results have been applied to the analysis of discrete load balancing algorithms.

3.1 Flow-related Results

In this section, we present three preliminary observations that relate the steady state of FOS/C to a corresponding flow problem.

Remark 3 [34] Due to Lemma 1 the load differences $\tilde{f} = \tilde{\mathbf{A}}^T w^{(\infty)}$ in the FOS/C steady state equal the $\|\cdot\|_2$-minimal flow $\tilde{f}^*$ that balances the load vector d/α, sending from the vertices in S (sources) the respective load amount δ to every vertex (sinks) in the graph: $\tilde{\mathbf{A}}\tilde{f} = \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T w^{(\infty)} = \mathbf{L}w^{(\infty)} = d/\alpha$.

Using this connection between the flow $\tilde{f}$ and FOS/C, we can easily obtain the following two results.

Proposition 1 *Consider an FOS/C procedure with source set S on the graph $G = (V, E, \omega)$. In the FOS/C steady state, described by the load vector w in $\mathbf{L}w = d$, 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 $w_{v_i} < w_{v_{i+1}}, 0 \leq i < l$.*

Proof The steady state of FOS/C is equivalent to a flow problem where all vertices $v \in V \backslash S$ receive a load amount of δ (cf. Remark 3). In particular, the flow along each edge $e = (u, v) \in E$ is given by the scaled load difference in the steady state, i.e., $\sqrt{\omega_e}(w_u - w_v)$. Hence for every vertex $v \in V \setminus S$, there must be at least one neighbor $u = u(v)$ so that $w_u > w_v$. Since G is finite, the path must eventually lead to a vertex $u \in S$. □

Note that Proposition 1 was derived in similar form before by Grady [20, p. 54, Lemma 1], but only for nonnegative vectors d and with a different proof approach.

Lemma 3 *Consider the load vector w in the steady state of FOS/C and the corresponding flow problem described in Remark 3. Then, each vertex v with maximum load value in w belongs to the set of source vertices S.*

Proof Let v be one of the vertices for which w_v is maximal. Then for every neighbor u of v, $w_v \geq w_u$. Hence, in the corresponding flow problem, the vertex v sends some load amount to all of its neighbors. In particular, the vertex v cannot receive a load amount of δ, and thus, $v \in S$ (cf. Remark 3). □

3.2 FOS/C Load Values as Similarity Measure

To determine the similarity (or distance) of graph vertices to one another, a formal notion of similarity is given.

Definition 5 (cf. [24, p. 440]) Let V be a finite set of vertices. We call a function $\mathcal{S} : V \times V \to \mathbb{R}$ a *similarity measure* for V if

- $\mathcal{S}(u, v) = \mathcal{S}(v, u)$ for all $u, v \in V$ and
- $\mathcal{S}(u, u) \geq \mathcal{S}(u, v)$ for all $u, v \in V$.

The symmetric matrix $\mathbf{S} = (s_{u,v} = \mathcal{S}(u, v))$ is called *similarity matrix.*

Note that the function values of similarity measures are sometimes also required to lie in the interval $[0, 1]$, which is not fulfilled by FOS/C, but could be ensured by suitable scaling.

Proposition 2 *For any edge-weighted graph, the matrix* $(\mathbf{W})_{u,v} = [w]_v^u$ *(i. e., the matrix whose* u*th row is given by the load vector of the single-source FOS/C procedure with source* u*) is a similarity matrix according to Definition 5.*

Proof Since $[w]_u^v = [w]_v^u$ [33], the symmetry property is fulfilled. Also, due to Lemma 3, the similarity of a source vertex to itself is always larger than the similarity to other vertices. □

The FOS/C similarity matrix can actually be written as $\mathbf{W} = n\delta\mathbf{L}^\dagger$ [31, Corollary 3.20], a scalar multiple of the (Moore-Penrose) pseudoinverse [19] of the graph's Laplace matrix $\mathbf{L}$.

Of course, the properties stated in Definition 5 are only the minimum requirements for a function to be called a similarity measure. In order to yield high-quality vertex groupings, a similarity measure used for graph partitioning needs additional, more specific properties.

3.3 Relating FOS/C to Random Walks

It is well-known that ordinary, i. e., undisturbed, diffusion and random walks are closely related, see Lovász's survey on random walks [29]. In particular, the doubly-stochastic diffusion matrix $\mathbf{M}$ can be considered as the transition matrix of a random walk on $V(G)$. Using the random walk notion, $[\mathbf{M}]_{u,v}$ denotes the probability for a random walk located in vertex u to move to vertex v in the next time step.

In order to examine the relationship between *disturbed* diffusion and random walks, we show that the most important part of an FOS/C steady state

is the sum of random walk transition probabilities. These probabilities are determined by the diffusion matrix $\mathbf{M}$, and the random walks have an increasing number of steps. We expand the original definition of FOS/C and obtain

$$\begin{aligned} w^{(t+1)} &= \mathbf{M}w^{(t)} + d \\ &= \mathbf{M}^2 w^{(t-1)} + \mathbf{M}d + d \\ &\vdots \\ &= \mathbf{M}^{t+1} w^{(0)} + (\mathbf{I} + \mathbf{M}^1 + \ldots + \mathbf{M}^t)d. \end{aligned}$$

Using this expansion, we can show that the load differences in the steady state of FOS/C can be expressed as scaled differences of the random walk measure *hitting times*.

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

The next result relates the steady state of FOS/C to hitting times of random walks.

Theorem 1 *Consider a single-source FOS/C procedure with $S = \{s\}$. In the FOS/C steady state, described by the load vector w in $\mathbf{L}w = d$, it holds for two vertices $u, v \in V$ not necessarily distinct from a source $s \in V$:*

$$\begin{aligned} w_u^{(\infty)} - w_v^{(\infty)} &= \frac{1}{\alpha}(w_u - w_v) = \lim_{t\to\infty} n\delta \left(\sum_{i=0}^{t} \mathbf{M}_{u,s}^i - \sum_{i=0}^{t} \mathbf{M}_{v,s}^i \right) \\ &= \delta(H[v, s] - H[u, s]). \end{aligned}$$

Proof We assume without loss of generality that the vertices are ordered in such a way that the source vertex is the first one. Then some rearranging of the FOS/C iteration scheme yields

$$\begin{aligned} w_u^{(t+1)} &= [\mathbf{M}^{t+1} w^{(0)}]_u + [(\mathbf{I} + \mathbf{M}^1 + \ldots + \mathbf{M}^t) \cdot (\delta(n-1), -\delta, \ldots, -\delta)^T]_u \\ &= [\mathbf{M}^{t+1} w^{(0)}]_u + \sum\nolimits_{i=0}^{t} (\delta(n-1)) \mathbf{M}_{u,s}^i \\ &\quad + \sum\nolimits_{i=0}^{t} \sum\nolimits_{v\in V, v\neq s} (-\delta) \mathbf{M}_{u,v}^i \\ &= [\mathbf{M}^{t+1} w^{(0)}]_u + n\delta \sum\nolimits_{i=0}^{t} \mathbf{M}_{u,s}^i - (t+1)\delta. \end{aligned}$$

As $\mathbf{M}^{t+1}w^{(0)}$ converges towards the balanced load distribution [10], we have to consider $\lim_{t\to\infty}\sum_{i=0}^{t}(\mathbf{M}^i_{u,s}-\mathbf{M}^i_{v,s})$ only. By a result of Kemeny and Snell [25, p. 79], $H[u,s]=(-\sum_{i=1}^{\infty}(\mathbf{M}^i_{u,s}-1/n)+\mathbf{Z}_{s,s})\cdot n$, where $\mathbf{Z}$ is the so-called fundamental matrix. Subtracting and dividing by n yields the desired result. □

The FOS/C expansion in the proof of Theorem 1 also shows that for the interpretation of the load distribution in the steady state, only the part $\lim_{t\to\infty} n\delta\sum_{i=0}^{t}[\mathbf{M}^i]_{u,s}$ is of particular interest. The expression $[\mathbf{M}^i]_{u,s}$ denotes the probability of a random walk described by $\mathbf{M}$ to start in s and be located on u after i steps. In its spectral decomposition [46, Ch. 24], this matrix entry can be written as $[\mathbf{M}^i]_{u,s}=\sum_{j=1}^{n}\mu_j^i[z_j]_u[z_j]_s$, where z_j denotes the j-th eigenvector and μ_j the j-th eigenvalue of $\mathbf{M}$. The largest absolute eigenvalue of $\mathbf{M}$ is $\mu_1=1$ [10], it corresponds to the eigenvector $z_1=(1,\ldots,1)^T$ (or any scalar multiple of this vector). Since μ_1 is simple for connected non-bipartite graphs [10], $|\mu_i|<1$ for all $i>1$. Hence, the μ_i^t with $2\le i\le n$ converge to 0 for $t\to\infty$, and the limit of the spectral decomposition is

$$\lim_{t\to\infty}\sum_{j=1}^{n}\mu_j^t[z_j]_u[z_j]_s=[z_1]_u[z_1]_s\,.$$

Thus, all entries of $\mathbf{M}^t$ converge towards $[z_1]_u[z_1]_s$. As z_1 is the balanced distribution with all entries equal, the summands with large i in $\sum_{i=0}^{t-1}[\mathbf{M}^i]_{u,s}$ are of low importance. These values are very similar for large i, regardless of the choice of s and u. In contrast to this, the summands for small values of i reveal by the random walk interpretation if a transition between two vertices is likely or not.

For a similar analogy, let $\mathbf{A}$ and $\mathbf{M}$ denote the adjacency and diffusion matrix of a graph G, respectively. Both matrices are structurally similar. Their nonzero pattern differs only at the diagonal. It is well-known that $[\mathbf{A}^t]_{i,j}$ denotes the number of paths of length exactly t between i and j in G. In a similar way $[\mathbf{M}^t]_{i,j}$ is counting paths of length t as well, with the difference of scaling by α and having a loop probability at the diagonal.

One might wonder why FOS/C needs to be iterated for an infinite number of steps if only the first few iterates contribute significantly to the result. The reason is that by taking the results of all random walks with lengths $0,\ldots,\infty$ into account, FOS/C can be used for general graphs without determining a *specific* suitable walk length. Hence, FOS/C is a robust mechanism for identifying if two vertices u and s are densely connected to each other. This notion of connectedness can be extended to graph regions as well by using a larger source set S.

3.4 FOS/C Monotonicity Results

Next we analyze a single-source FOS/C procedure on torus graphs and hypercubes in the steady state, namely, its edge flow and the corresponding load distribution. For simplicity we restrict the results of this section to unweighted graphs. We start by defining some graph-theoretic notions.

Definition 7 cf. [7, p. 115ff.] Given a graph $G = (V, E)$, a permutation π of V is said to be an *automorphism* if $\{u, v\} \in E \Leftrightarrow \{\pi(u), \pi(v)\} \in E, \forall u, v \in V$. Moreover, G is *vertex-transitive* if for any two distinct vertices of V there is an automorphism mapping one to the other. G is *distance-transitive* if, for all vertices $u, v, x, y \in V$ such that $\text{dist}(u, v) = \text{dist}(x, y)$, there exists an automorphism ϕ for which $\phi(u) = x$ and $\phi(v) = y$.

We continue by giving the formal definition of a k-dimensional torus, which includes the hypercube as a special case.

Definition 8 The k-dimensional torus $T[d_1, \ldots, d_k] = (V, E)$ is defined as:

$$
\begin{aligned}
V &= \{(u_1, \ldots, u_k) \,|\, 0 \leq u_\nu \leq d_\nu - 1 \text{ for } 1 \leq \nu \leq k\} \text{ and} \\
E &= \{\{(u_1, \ldots, u_k), (v_1, ..., v_k)\} \,|\, \exists\, 1 \leq \mu \leq k \\
&\qquad \text{with } v_\mu = (u_\mu + 1) \bmod d_\mu \text{ and } u_\nu = v_\nu \text{ for } \nu \neq \mu\}.
\end{aligned}
$$

The k-dimensional hypercube $Q(k)$ is the torus graph $T[d_1, \ldots, d_k]$ with $d_i = 2$ for each $1 \leq i \leq k$.

Torus graphs with small k (in particular $k \in \{1, 2, 3\}$), are very important in theory [27] and practice [44], e. g., because they have bounded degree, are regular and vertex-transitive, and correspond to the structure of numerical simulation problems that decompose their domain by structured grids with cyclic boundary conditions.

The hypercube network is a very important network for parallel computing [27]. Additionally, it has the nice property of being distance-transitive (cf. [7]).

Now we exploit the simple structure and symmetries of the torus to show monotonicity with respect to the FOS/C steady state load distribution. Since we are only interested in the steady state, we will set $\alpha = (\deg(G) + 1)^{-1}$, so that all entries of the diffusion matrix $\mathbf{M}$ are either 0 or α. This is a usual choice for transition matrices in random walk theory.

Consider an arbitrary k-dimensional torus $T[d_1, \ldots, d_k]$. Recall that each vertex u is uniquely represented as a k-dimensional vector $u = (u_1, \ldots, u_k), \forall i \in 1, \ldots, k : 0 \leq u_i < d_i$. Since any torus is vertex-transitive, we assume without loss of generality that the source vertex is the zero-vector. Denote by $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ the unit-vector containing with exactly

one coordinate being equal to 1, namely in the i-th component. Note that each edge traversal corresponds to the addition (or subtraction) of some $\mathbf{e}_i$, where we always assume that the i-th component is meant to be modulo d_i. It is also easy to see that the distance between two vertices (vectors) is given by $\mathrm{dist}(u,v) = \sum_{i=1}^{k} \min\{|u_i - v_i|, d_i - |u_i - v_i|\}$.

Let u, v, s be three vertices with $\{u,v\} \in E(G)$ such that $\mathrm{dist}(u,s) < \mathrm{dist}(v,s)$, i.e., there exists a shortest path from s to v via u. Assume without loss of generality that u and v are adjacent along the j-th dimension: $v = u + \mathbf{e}_j$. It follows from the formula for $\mathrm{dist}(u,s)$ (and $\mathrm{dist}(v,s)$) that there is a shortest path from $s + \mathbf{e}_i$ (or $s - \mathbf{e}_i$, respectively) to v via u for all $i \neq j$.

Observation 2 *Let $T[d_1, \ldots, d_k] = (V, E)$ be a torus graph. Then, for all $i \in \{1, \ldots, k\}$, the following two functions are automorphisms:*

$$\psi_i(u_1, \ldots, u_k) = (u_1, \ldots, u_{i-1}, u_i + 1, u_{i+1}, \ldots, u_k),$$
$$\varphi_i(u_1, \ldots, u_k) = (u_1, \ldots, u_{i-1}, d_i - u_i, u_{i+1}, \ldots, u_k).$$

Intuitively, ψ_i represents a translation into direction i, while φ_i can be thought of as a reflection along the i-th dimension.

We also observe that for all $\varphi \in \mathrm{Aut}(G)$ and all time steps t we have $\mathbf{M}^t_{u,v} = \mathbf{M}^t_{\varphi(u),\varphi(v)}$ [1, p. 151]. Using this and the automorphisms of Observation 2, we now prove the following monotonicity theorem, which may be of independent interest for random walks in general.

Theorem 3 *Let $T[d_1, \ldots, d_k] = (V, E)$ be a torus graph with $d_i \geq 2$ for all $1 \leq i \leq k$. Consider three vertices u, v, s which satisfy $\{u,v\} \in E$, $\mathrm{dist}(u,s) \leq \mathrm{dist}(v,s)$ and choose $\alpha = (\deg(G) + 1)^{-1}$. Then,*

$$\forall t \in \mathbb{N}_0 : \mathbf{M}^t_{u,s} \geq \mathbf{M}^t_{v,s}.$$

Proof By symmetry of the torus graph, we may assume that $s = (0, \ldots, 0)$. We will prove the statement by induction for all vertices u, v, s on the number of time steps t. Obviously, the claim is true for $t = 0$. Assuming that the induction hypothesis is true for $t-1$, we will prove that it also holds for t. Note that since the torus is regular, the choice of $\alpha = (\deg(G) + 1)^{-1}$ ensures that all entries of $\mathbf{M}$ are either zero or α. Hence,

$$\mathbf{M}^t_{u,s} = \alpha \left(\mathbf{M}^{t-1}_{u,s} + \sum_{i \in \{1,\ldots,k\}} \mathbf{M}^{t-1}_{u,s+\mathbf{e}_i} + \sum_{i \in \{1,\ldots,k\}} \mathbf{M}^{t-1}_{u,s-\mathbf{e}_i} \cdot \mathbf{1}_{d_i > 2} \right), \quad (3)$$

$$\mathbf{M}^t_{v,s} = \alpha \left(\mathbf{M}^{t-1}_{v,s} + \sum_{i \in \{1,\ldots,k\}} \mathbf{M}^{t-1}_{v,s+\mathbf{e}_i} + \sum_{i \in \{1,\ldots,k\}} \mathbf{M}^{t-1}_{v,s-\mathbf{e}_i} \cdot \mathbf{1}_{d_i > 2} \right). \quad (4)$$

Our strategy is now to find for any summand in $\mathbf{M}^t_{v,s}$ a proper summand in $\mathbf{M}^t_{u,s}$ which is not smaller by using the induction hypothesis for $t-1$. Of course, if this is done bijectively, we have shown that $\mathbf{M}^t_{u,s} \geq \mathbf{M}^t_{v,s}$. Let $j \in \{1,\dots,k\}$ be the integer such that $v = u + \mathbf{e}_j$ (the case $v = u - \mathbf{e}_j$ is proven in the same way). To proceed, we divide this proof into two cases.

1. Case $u_j = 0$: By Observation 2 we have

$$\mathbf{M}^{t-1}_{u,s} = \mathbf{M}^{t-1}_{\psi_j(u),\psi_j(s)} = \mathbf{M}^{t-1}_{v,s+\mathbf{e}_j}.$$
$$\mathbf{M}^{t-1}_{u,s+\mathbf{e}_j} = \mathbf{M}^{t-1}_{\varphi_j(u),\varphi_j(s+\mathbf{e}_j)} = \mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} = \mathbf{M}^{t-1}_{\psi_j(u),\psi_j(s-\mathbf{e}_j)} = \mathbf{M}^{t-1}_{v,s}.$$

   Note that if $d_j = 2$, then we don't have to relate $\mathbf{M}_{u,s-\mathbf{e}_j}$ and $\mathbf{M}_{v,s-\mathbf{e}_j}$, since $s - \mathbf{e}_j = s + \mathbf{e}_j$. In case of $d_j \neq 2$, we have additionally to show that $\mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j}$. To do so, we distinguish between the following subcases.

   (a) The first case is $d_j = 3$:
   $\mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} \overset{\text{(s. above)}}{=} \mathbf{M}^{t-1}_{u,s+\mathbf{e}_j} \overset{\psi_j}{=} \mathbf{M}^{t-1}_{u+\mathbf{e}_j,s+2\mathbf{e}_j} \overset{d_j=3}{=} \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j}$.

   (b) $d_j \geq 4$: Then, $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s) + 1$, implying $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s) + 1 \geq \mathrm{dist}(u,s) + 1 \geq \mathrm{dist}(u, s-\mathbf{e}_j)$. Applying the induction hypothesis gives $\mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j}$.

   Note that for all $i \in \{1,\dots,k\}, i \neq j$, there exists a shortest path from v to $s \pm \mathbf{e}_i$ via u, so that we can again conclude inductively that $\mathbf{M}^{t-1}_{u,s\pm\mathbf{e}_i} \geq \mathbf{M}^{t-1}_{v,s\pm\mathbf{e}_i}$. With Equation (4) the claim $\mathbf{M}^t_{u,s} \geq \mathbf{M}^t_{v,s}$ follows.
2. Case $u_j \neq 0$, which implies $u \neq s$ so that all vertices u, v, s are different. We first note that $d_j = 2$ is not possible. For if it was, then $d_j = 2$ together with $u_j \neq 0$ implies $v = s$, a contradiction. Hence, we assume that $d_j \geq 3$ in the following.

   (a) d_j is even. We distinguish between the following cases.
      i. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s)+1$ implies $\mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j}$, $\mathbf{M}^{t-1}_{u,s} \geq \mathbf{M}^{t-1}_{v,s}$, and $\mathbf{M}^{t-1}_{u,s+\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s+\mathbf{e}_j}$.
      ii. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s) - 1$, implying $v_j = \frac{d_j}{2}$ and $u_j = \frac{d_j}{2} - 1$. Using the induction hypothesis and Observation 2, we have

$$\mathbf{M}^{t-1}_{u,s+\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s+\mathbf{e}_j} \overset{\varphi_j}{=} \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j},$$
$$\mathbf{M}^{t-1}_{u,s} \overset{\psi_j}{=} \mathbf{M}^{t-1}_{v,s+\mathbf{e}_j}, \ \mathbf{M}^{t-1}_{v,s} \overset{\psi_j^{-1}}{=} \mathbf{M}^{t-1}_{u,s-\mathbf{e}_j}.$$

      iii. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s)$ is not possible, as d_j is even.

   The same argument as in Case (1) finishes the case where d_j is even.

   (b) d_j is odd. Again we distinguish between the following cases.
      i. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s) + 1$ is exactly done as in 2(a)i.

ii. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s) - 1$. Hence $u_j = \frac{d_j-1}{2}$ and $v_j = \frac{d_j+1}{2}$. Applying the automorphisms yields

$$\mathbf{M}^{t-1}_{s-\mathbf{e}_j,v} \overset{\phi_j}{=} \mathbf{M}^{t-1}_{s+\mathbf{e}_j,u}, \quad \mathbf{M}^{t-1}_{s+\mathbf{e}_j,v} \overset{\phi_j}{=} \mathbf{M}^{t-1}_{s-\mathbf{e}_j,u}, \text{ and } \mathbf{M}^{t-1}_{s,u} \overset{\phi_j}{=} \mathbf{M}^{t-1}_{s,v}.$$

iii. $\mathrm{dist}(v, s-\mathbf{e}_j) = \mathrm{dist}(v,s)$. Hence, $u_j = \frac{d_j-3}{2}$ and $v_j = \frac{d_j-1}{2}$. Applying induction and the automorphisms yields

$$\mathbf{M}^{t-1}_{v,s-\mathbf{e}_j} \overset{\varphi_j}{=} \mathbf{M}^{t-1}_{v+\mathbf{e}_j,s} \overset{\phi_j}{=} \mathbf{M}^{t-1}_{v,s} \leq \mathbf{M}^{t-1}_{u,s} \leq \mathbf{M}^{t-1}_{u,s+\mathbf{e}_j},$$

so that the three inequalities

$$\mathbf{M}^{t-1}_{u,s+\mathbf{e}_j} \geq \mathbf{M}^{t-1}_{v,s-\mathbf{e}_j}, \quad \mathbf{M}^{t-1}_{v,s+\mathbf{e}_j} \overset{\psi_j^{-1}}{=} \mathbf{M}^{t-1}_{u,s}, \text{ and } \mathbf{M}^{t-1}_{u,s-\mathbf{e}_j} \overset{\psi_j}{=} \mathbf{M}^{t-1}_{v,s}$$

finish this case.

As before, the same argument as in Case (1) finishes the case where d_j is even.

□

We observe that if all $d_i = 2$, then $T[d_1, \ldots, d_k]$ is just the k-dimensional hypercube. Hence Theorem 3 generalizes a monotonicity result by Diaconis et al. [12, Lemma 2] for the hypercube. We also mention the general result $\mathbf{M}^{2t}_{u,u} \geq \mathbf{M}^{2t}_{u,v}$ for random walks without loops on vertex-transitive graphs given by Alon and Spencer [1, p. 150], which is generalized by our last theorem on torus graphs.

This concludes the part on (general) torus graphs. In the remainder we consider the hypercube as the most important representative of distance-transitive graphs (see Definition 7 for the definition of distance-transitive graphs). Before we analyze FOS/C, we study the level-structure of distance-transitive graphs.

Definition 9 Given a graph $G = (V, E)$, let $N_i(u) := \{v \in V \mid \mathrm{dist}(u,v) = i\}$ denote the i-neighborhood of $u \in V$. We say that G has a level structure with respect to a vertex $s \in V$ if V can be partitioned into levels $\{s\} = L_0, L_1, \ldots, L_\Lambda$ such that for all $0 \leq i \leq \Lambda$:

$$\forall u, v \in L_i \; \forall j \in \{0, \ldots, \Lambda\} : |N_1(u) \cap L_j| = |N_1(v) \cap L_j| \\ \text{and } L_0 \dot\cup \ldots \dot\cup L_\Lambda = V\,.$$

The next lemma demonstrates that every distance-transitive graph has a level structure.

Lemma 4 *([7, p. 155f.],[18, p. 67]) If $G = (V, E)$ is distance-transitive, then $N_i(s)$ forms the i-th level $L_i(s)$ of a level structure in G with respect to an arbitrary vertex $s \in V$.*

As an example, the k-dimensional hypercube $Q(k)$ has $\Lambda = k+1$ such levels.

Proposition 3 *Consider the steady state of a single-source FOS/C procedure with $S = \{s\}$ on a distance-transitive graph. Then for every time step $t \geq 0$, it holds for all vertices u, v with the same graph distance to s that $w_u^{(t)} = w_v^{(t)}$.*

Proof Let Λ be the number of levels with respect to s. We prove by induction on t that for every step t there is a vector $\tilde{w}^{(t)} \in \mathbb{R}_+^{\Lambda}$, so that for every vertex v which is in a level $L_i = L_i(s) = N_i(s)$, $w_v^{(t)} = \tilde{w}_i^{(t)}$.

For the base case, $t = 0$, the claim is trivially fulfilled. Due to the level structure of G, we can write the FOS/C iteration formula for any vertex v in level $L_i, 1 \leq i \leq \Lambda$ as follows:

$$\begin{aligned} & w_v^{(t+1)} \\ &= w_v^{(t)} + d_v - \alpha \sum_{u \in L_{i-1} \wedge \{u,v\} \in E} \left(w_v^{(t)} - w_u^{(t)} \right) - \alpha \sum_{u \in L_{i+1} \wedge \{u,v\} \in E} \left(w_v^{(t)} - w_u^{(t)} \right) \\ &= \tilde{w}_i^{(t)} + d_v - \alpha \sum_{u \in L_{i-1} \wedge \{u,v\} \in E} \left(\tilde{w}_i^{(t)} - \tilde{w}_{i-1}^{(t)} \right) - \alpha \sum_{u \in L_{i+1} \wedge \{u,v\} \in E} \left(\tilde{w}_i^{(t)} - \tilde{w}_{i+1}^{(t)} \right). \end{aligned}$$

Let $E_{i,i+1} := \{\{u,v\} \in E : u \in L_i, v \in L_{i+1}\}$. By Lemma 4 it holds for every $v \in L_i$ that $|\{u \in L_{i-1} \colon \{u,v\} \in E\}| = |E_{i,i+1}|/|L_i|$, and $|\{u \in L_{i+1} \colon \{u,v\} \in E\}| = |E_{i,i+1}|/|L_i|$. Hence,

$$\begin{aligned} w_v^{(t+1)} = \tilde{w}_i^{(t)} + d_v - \alpha \cdot \frac{|E_{i-1,i}|}{|L_i|} \cdot \left(\tilde{w}_i^{(t)} - \tilde{w}_{i-1}^{(t)} \right) \\ - \alpha \cdot \frac{|E_{i,i+1}|}{|L_i|} \cdot \left(\tilde{w}_i^{(t)} - \tilde{w}_{i+1}^{(t)} \right) =: \tilde{w}_i^{(t+1)}. \end{aligned}$$

Therefore, the induction step also holds for step $t+1$ using the above definition for the vector $\tilde{w}^{(t+1)}$. □

We know by Proposition 1 that for each vertex $v \in V \backslash \{s\}$ of an arbitrary graph there exists a path from v to s such that by traversing it, the load amount increases. Now we can show that for distance-transitive graphs this property holds on *every shortest* path.

Theorem 4 *Consider the steady state of a single-source FOS/C procedure with $S = \{s\}$ on a distance-transitive graph G. Then for all $u, v \in V$ with $\operatorname{dist}(u,s) < \operatorname{dist}(v,s)$ it holds that $[w]_u^s > [w]_v^s$.*

Proof Recall the equivalence of the FOS/C steady state to the $\|\cdot\|_2$-minimal flow problem of Remark 3. When load is sent from vertex s to a vertex $v \in L_i$, this load has to pass all levels $L_{i'}$ with $i' < i$, $0 < i \leq \Lambda$. This means there is at least one vertex $v' \in L_{i-1}$ with a neighbor $\tilde{v} \in L_i \cap N(v')$ and $f^*_{(v',\tilde{v})} > 0$.

By definition of f^*, this implies $f^*_{(v',\tilde{v})} = [w]^s_{v'} - [w]^s_{\tilde{v}} > 0$. Since for each vertex $\hat{v} \in L_{i-1}$, $[w]^s_{\hat{v}}$ is the same (Proposition 3), all vertices of level $i-1$ have a higher load than vertices in level i. □

We now derive an explicit formula for the FOS/C flow in the steady state, assuming that the graph G is distance-transitive.

Theorem 5 *Consider the steady state of a single-source FOS/C procedure with $S = \{s\}$ on a distance-transitive graph. Let $e = \{u, v\} \in E$ be an arbitrary edge with $u \in L_i = L_i(s)$ and $v \in L_{i+1} = L_{i+1}(s)$ $(0 < i < \Lambda)$. Moreover, let $E_{i,i+1}(s) := \{\{u, v\} \in E : u \in L_i, v \in L_{i+1}\}$ denote the set of edges running between levels i and $i+1$, $0 \leq i < \Lambda$. Then, the FOS/C flow in the steady state f^*_e on an edge $e = \{u, v\} \in E_{i,i+1}(s)$ (viewed from u to v) is given by*

$$w_u - w_v = f^*_e = \frac{\delta}{|E_{i,i+1}(s)|} \cdot \sum_{j=i+1}^{\Lambda} |L_j| \,.$$

Proof The amount of load which reaches $u \in L_i$ in the steady state needs to pass all levels $L_{i'}$ with $i' < i$ (cf. Observation 3). As all vertices in levels larger than i need to receive their load amount δ,

$$\sum_{e \in E_{i,i+1}(s)} f^*_e = \delta \sum_{j=i+1}^{\Lambda} |L_j|.$$

Moreover, vertices of the same level have the same load (Proposition 3), and thus for every edge $e \in E_{i,i+1}(s)$, f^*_e is the same and the statement follows by dividing by $|E_{i,i+1}(s)|$. □

Each vertex of the k-dimensional hypercube $Q(k)$ corresponds to a bit-string of length k. Since $Q(k)$ is k-regular, vertex- and edge-transitive [7], we may assume without loss of generality that $s = 0^k$. Due to its known structure, the FOS/C flow in the steady state on the hypercube can be stated more explicitly.

Corollary 1 *Consider the steady state of a single-source FOS/C procedure with $S = \{s\}$ on the k-dimensional hypercube $Q(k) = (V, E)$. Then the FOS/C flow in the steady state f^*_e on an edge $e = \{u, v\} \in E$ (u in level i, v in level $i+1$, $0 \leq i < \Lambda$) is $w_u - w_v = f^*_e = \frac{\delta}{\binom{k}{i}(k-i)} \cdot \sum_{j=i+1}^{k} \binom{k}{j}$, where the flow on e is viewed from u to v.*

Proof Since one chooses i out of k bits to be set to 1 to reach a level-i vertex, level i of $Q(k)$ contains $\binom{k}{i}$ vertices. Consequently, $|E_{i,i+1}(s)| = \binom{k}{i}(k-i)$, as each vertex in level i has $k-i$ neighbors in level $i+1$. □

3.5 Application to Discrete Load Balancing Algorithms

The result of the hypercube can be used to compute the *local divergence* [38, Definition 2], which is defined by

$$\Psi(G) := \max_{s \in V} \sum_{t=0}^{\infty} \sum_{\{u,v\} \in E} \left| \mathbf{M}^t_{u,s} - \mathbf{M}^t_{v,s} \right|.$$

As mentioned by Rabani et al. [38], this natural quantity "appears (...) to be of independent interest" for "the study of the transient behavior of random walks on infinite graphs". A more concrete application of $\Psi(G)$ is to relate it to the performance of a natural discrete load balancing algorithms on finite graphs [38]. To this end, the authors bound $\Psi(G)$ in terms of the second largest eigenvalue and derive an asymptotically tight bound on $\Psi(G)$ for torus graphs. Based on our Theorem 3 and Corollary 1, Berenbrink et al. [6, Theorem 4.4, page 439] were able to compute the exact value of $\Psi(G)$ for the hypercube (we refer to [6] for more details), which improves over the previously best upper bound [38] by a factor of $\Theta(\log n)$.

4 Analyzing the Partitioning Algorithm Bubble-FOS/C

Now that we have gained a deeper understanding of FOS/C, we analyze its combination with the Bubble framework. After all, we use the Bubble-FOS/C algorithm to solve our actual graph partitioning problem heuristically.

It has been shown before [33, Thm. 10] that the iterative optimization performed by 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 with v's most similar center vertex 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, as pointed out before, 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 in Section 4.1, 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 [34], 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 vertices and the shape of the parts. Concerning the edge cut, the improvement over

the other libraries was not as clear, probably because KMETIS and JOSTLE focus primarily on the edge cut.

Two results on the connectedness of partitions conclude Section 4.2. First we use our flow-based results on FOS/C to prove that in a bipartitioning one part always has to be connected. If the graph class is restricted to vertex-transitive graphs, our second result shows that both parts are connected. The proof relies on the relation between FOS/C and hitting times of random walks.

4.1 Edge Cut Minimization

Our plan is to express edge 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 $\mathbf{X} \in \{0,1\}^{n \times k}$ be the matrix whose p-th column is $x_{(p)}$. Moreover, let $y_{(p,p')} := x_{(p)} - x_{(p')}$ and $\mathbf{Y}$ the matrix whose columns are the vectors $y_{(p,p')}$, $1 \leq p < p' \leq k$. Note that we assume throughout this section that k divides n.

It is well-known [16] that $x^T \mathbf{L} x = \sum_{\{u,v\} \in E} \omega(\{u,v\})([x]_u - [x]_v)^2$ for any $x \in \mathbb{R}^n$. Hence, finding a balanced partition with minimum edge cut can be written as:

$$\begin{aligned} \min_{\mathbf{X} \in \{0,1\}^{n \times k}} \quad & \textstyle\sum_{1 \leq p \leq k} x_{(p)}^T \mathbf{L} x_{(p)} \\ \text{subject to} \quad & \|x_{(p)}\|_1 = \tfrac{n}{k} \ \forall 1 \leq p \leq k \quad \text{(balanced parts)} \\ & \textstyle\sum_{1 \leq p \leq k} [x_{(p)}]_v = 1 \ \forall v \in V \ \text{(exactly one part per vertex)}. \end{aligned} \tag{5}$$

4.1.1 `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 vertices, $n/k \in \mathbb{N}$. To find a good solution, BUBBLE-FOS/C alternates the operations `AssignPartition` and `ComputeCenters`. Eventually, it finds a local optimum of the function F described above [33]. 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 for system p that changes according to the set of source vertices, and w_p is the resulting load vector.

To ensure balanced parts, `AssignPartition` is followed by an operation called `ScaleBalance` [34]. `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 4 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, mathematically 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. For technical reasons we also assume $0 < \beta_p \neq \beta_{p'} < 1$ for all $1 \leq p \neq p' \leq k$. For the BQP formulation these assumptions are feasible, which will become clear in the remainder of this section. It is essential that the drain vectors are adapted accordingly.

Definition 10 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

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

The following remark about the combination or fusion of drain vectors will be helpful in our upcoming analysis.

Remark 5 If $k = 2$, instead of solving $\mathbf{L}w_1 = d_1^{(A)}$ 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 5 and 6 that this fusion technique can be extended in a straightforward manner to $k > 2$ parts.

Using the adapted drain vectors, we can now formulate a BQP that describes the minimum cut problem in terms of Bubble-FOS/C notation.

Lemma 5 *Let $G = (V, E, \omega)$ be a graph with $\frac{n}{k} \in \mathbb{N}$, 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 $1 \leq p \neq q \leq k$), and the FOS/C drain vectors $d_p^{(A)}$.*

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:

$$\min_{\mathbf{X}\in\{0,1\}^{n\times k}} \sum_{1\le p\le k} x_{(p)}^T \mathbf{L} x_{(p)} \tag{6}$$

$$\text{subject to } y_{(p,p')}^T d_{(p,p')}^{(A)} = n\delta(\beta_p + \beta_{p'}) \ \forall (p,p') \tag{7}$$

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

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. Note that, to simplify the calculations, we will choose $\delta = 1$ without loss of generality.

Case 1 ($z_p \in \pi_p \ \forall\ 1 \le p \le k$): 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 $\pi_{p'}$ ($1 \le p \ne p' \le k$):

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

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 $|\pi_p| = |\pi_{p'}|$, completing this case.

Otherwise there exist two indices $p \ne p'$ with $z_p \in \pi_{p'} \ne \pi_p$. 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 $[y_{(p,p')}]_{z_p}$ by -1 and $[y_{(p,p')}]_{z_{p'}}$ by 1 as well as some rearranging yields

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

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 $0 < \beta_p, \beta_{p'} < 1$:

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

Case 3 ($z_p \in \pi_p$ and $z_{p'} \in \pi_p$): Substituting both $[y_{(p,p')}]_{z_p}$ and $[y_{(p,p')}]_{z_{p'}}$ by 1 yields

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

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 if $-(n-2)(\beta_p - \beta_{p'}) \neq -2((n-1)\beta_{p'} + \beta_p)$, which can be transformed to

$$(n-4)\beta_p \neq (3n-4)\beta_{p'}.$$

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 fulfilled if $(n-2)(\beta_p - \beta_{p'}) = -2((n-1)\beta_{p'} + \beta_p)$, i. e., if $n\beta_p = -n\beta_{p'}$. Due to $0 < \beta_p < \beta_{p'}$, equality 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 $y_{(p,p')}^T d_{(p,p')}^{(A)}$ evaluate to simple expressions:

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

Fulfilling the constraint would mean $\delta((\beta_p - \beta_{p'})(|\pi_{p'}| - |\pi_p|) - \beta_p n) = n(\beta_p + \beta_{p'})$, i. e., $\beta_p(2n + |\pi_{p'}| - |\pi_p|) = \beta_{p'}(|\pi_{p'}| - n - |\pi_p|)$. Since β_p and $\beta_{p'}$ are both positive, the left side of the equation is positive, while the right side is non-positive. Hence, the constraint cannot be fulfilled.

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. □

Corollary 2 *Let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be a balanced partition with minimum cut. If the set of center vertices $Z = \{z_1, \ldots, z_k\}$ is chosen in Lemma 5 such that $z_p \in \pi_p$ $(1 \leq p \leq k)$, then the BQP (6), (7) computes Π or another balanced partition with minimum cut.*

Since the minimum bisection problem with equally sized parts without specifying centers is $\mathcal{NP}$-hard [17], so is the optimization problem (6), (7). (Testing all $\mathcal{O}(n^2)$ many center combinations suffices to derive a simple reduction for $k = 2$.) The same hardness applies to balanced partitioning into a general number of parts. Andreev and Räcke have shown that for non-constant k the problem cannot be solved in polynomial time with finite approximation factor unless $\mathcal{P} = \mathcal{NP}$ [4, p. 932]. If the balance constraint is relaxed and partitions are allowed to be $(1 + \varepsilon)$ times larger than a balanced one, a polylogarithmic approximation exists [4]. For the case $k = 2$, polynomial time approximation algorithms are known even for the balanced case, the currently best approximation factor is $O(\log n)$ due to Räcke [39].

Focussing on balanced solutions, we continue our analysis by applying a relaxation technique to our problem formulation now. While a finite approximation guarantee is out of reach, relaxation can provide further insights into the optimization process of Bubble-FOS/C. 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 5 (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 (relaxed) optimization problem

$$\min_{\mathbf{Y} \in \mathbb{R}^{n \times \binom{k}{2}}} \sum_{1 \leq p < p' \leq k} y_{(p,p')}^T \mathbf{L} y_{(p,p')} \text{ with constraints as in (7).} \tag{8}$$

Lemma 6 *The global minimum $\overline{\mathbf{Y}}$ of Problem (8) can be computed by solving and evaluating k linear equations of the form $\mathbf{L}z_p = -\frac{1}{2}d_p^{(A)}$ $(1 \leq p \leq k)$, where*

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

Proof Recall that the combination of `AssignPartition` and `ScaleBalance` solves k linear systems of the form $\mathbf{L}x_{(p)} = d_p^{(A)}$ and assigns each vertex to the part with the highest load for that vertex. It is essential to observe that

this is equivalent to solving $\binom{k}{2}$ linear systems of the form $\mathbf{L}y_{(p,p')} = d^{(A)}_{(p,p')}$ and deciding a partial order with respect to to the higher load for each vertex 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 vertex v, the argmax 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 vertex.)

Regarding Eq. (8), using standard multidimensional calculus, one can easily see that the function $f(\mathbf{Y}) := \sum_{1 \le p < p' \le k} y^T_{(p,p')}\mathbf{L}y_{(p,p')}$ is differentiable over $\mathbb{R}^{n \times \binom{k}{2}}$, because it is a sum of differentiable functions. Furthermore, each constraint function $h(y_{(p,p')}) := y^T_{(p,p')}d^{(A)}_{(p,p')} - n\delta(\beta_p + \beta_{p'})$ is continuously differentiable over $\mathbb{R}^n$. Hence, we can use a Karush-Kuhn-Tucker argument (see [5, Ch. 4]) and let $\overline{\mathbf{Y}} = (\overline{y}_{(1,2)}, \overline{y}_{(1,3)}, \ldots, \overline{y}_{(k-1,k)})$ be a feasible solution. For $\overline{\mathbf{Y}}$ 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

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

$$\text{yielding } \; 2\mathbf{L} \sum_{1 \le p < p' \le k} \overline{y}_{(p,p')} = - \sum_{1 \le p < p' \le k} \Lambda_{(p,p')} d^{(A)}_{(p,p')}\,.$$

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

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

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

Hence, a suitable Λ exists. Following Bazaraa et al. [5, 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 exploiting that $\mathbf{L}$ is positive semidefinite ($x^T\mathbf{L}x \ge 0\,\forall x$)), so that $\overline{\mathbf{Y}}$ is a global optimum of Equation (8). 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 $\overline{y}_{(p,p')} = \Lambda_{(p,p')} z_{(p,p')} = \frac{n\delta(\beta_p + \beta_{p'})}{z^T_{(p,p')} \cdot d^{(A)}_{(p,p')}} \cdot z_{(p,p')}$. □

Let us make clear now why the assumption of already known β_p is feasible. First, recall from Remark 4 that the result of the linear systems is the same, regardless when the β_p are introduced into the equations. Lemma 5 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 4.1.1 with the following insight.

Theorem 6 *Let $k \geq 2$. Given a graph $G = (V, E, \omega)$ with n vertices ($n/k \in \mathbb{N}$) and a set Z with one center vertex for each of the k parts.*

The two consecutive operations `AssignPartition` and `ScaleBalance` with suitable β_p ($1 \leq p \leq k$) together compute the global minimum of the Optimization Problem (8), where (8) is a relaxed version of the edge cut minimizing BQP (6), (7).

The solution of the (unrelaxed) BQP (6), (7) is a partition with minimum edge cut if $Z = \{z_{1,\ldots,}z_k\}$ is given such that $z_p \in \pi_p$ with $\Pi = \{\pi_1, \ldots, \pi_p\}$ being a partition with minimum edge cut.

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, $1 \leq p \leq k$. Performing `ScaleBalance` results in the load vector $\beta_p w_p$. With the Remarks 4 and 5 and the proof of Lemma 6, 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 consequence of the results above, for suitable β_p these load vectors $w_{(p,p')}$ correspond to a relaxed optimal solution of the BQP (6), (7).

According to Lemma 5 and Corollary 2, a solution to BQP (6), (7) has minimum edge cut given an optimal placement of the center vertices. □

4.1.2 `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 4.1). 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 4.1.1, 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 any more, 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 6 To establish a BQP for `ComputeCenters` given the input partition Π, we simply replace all occurrences of $d^{(A)}$ by $d^{(C)}$ in Equation (7), eliminate the unnecessary β_p, and use the indicator vectors $x_{(p)}$ here:

$$x_{(p)}^T d_p^{(C)} = \delta(n - |\pi_p|) \;\; \forall 1 \leq p \leq k\,. \tag{9}$$

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 7 *The constraints in Equation (9) ensure that the input partition Π remains unchanged. In particular, the centers $Z = \{z_1, \ldots, z_k\}$ computed by the BQP (6), (9) fulfill $z_p \in \pi_p$ for all $p \in \{1, \ldots, k\}$.*

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, $x_{(p)}^T d_p^{(C)}$ evaluates to

$$x_{(p)}^T d_p^{(C)} = \sum_{j \in \pi_p} [d_p^{(C)}]_j = |\pi_p|(\delta(\frac{n}{|\pi_p|} - 1)) = \delta(n - |\pi_p|)\,,$$

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

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

To fulfill the constraint, we must obtain $s_{p\to p} \cdot \frac{n}{|\pi_p|} - s_{p\to p} - s_{p'\to p} = n - |\pi_p|$. However, regardless of the actual size of $s_{p'\to p}$, $s_{p\to p} \cdot \frac{n}{|\pi_p|} - s_{p\to p} - s_{p'\to p} \leq$

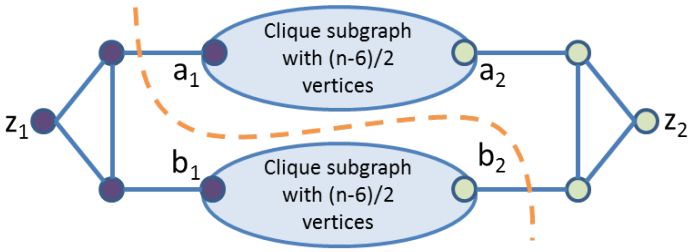

Fig. 4 Example where the optimal cut (shown as dashed orange line) size is $\mathcal{O}(1)$, but Bubble-FOS/C produces a cut of size at least $\Omega(n)$.

$s_{p\to p}(\frac{n}{|\pi_p|}-1) = \frac{s_{p\to p}(n-|\pi_p|)}{|\pi_p|}$. If $s_{p\to p} < |\pi_p|$, i. e., some vertices would leave the current input part, then $\frac{s_{p\to p}(n-|\pi_p|)}{|\pi_p|} < n-|\pi_p|$, so that the constraint cannot be fulfilled. For the remaining part let $s_{p\to p} = |\pi_p|$. If $s_{p'\to p} > 0$, then similarly $s_{p\to p}\cdot\frac{n}{|\pi_p|} - s_{p\to p} - s_{p'\to p} < s_{p\to p}(\frac{n}{|\pi_p|}-1) = \frac{s_{p\to p}(n-|\pi_p|)}{|\pi_p|} = n-|\pi_p|$. Hence, the constraints can only be fulfilled if the input partition Π remains unchanged. □

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 (9). Again, we relax the binary condition on $x_{(p)}$, i. e., let $x_{(p)} \in \mathbb{R}^n$. Then, the following result is due to the fact that $d^{(C)}$ is constant for all vertices of the same part and that $[x_{(p)}]_{z_p} = \operatorname{argmax}_{1\le v\le n}[x_{(p)}]_v$.

Corollary 3 *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 $[x_{(p)}]_{z_p} d^{(C)}_{z_p}$ contributes the highest value of all vertices in π_p to $x^T_{(p)} \cdot d^{(C)}_p$, $1 \le p \le k$.*

4.1.3 Discussion

Our results above provide some explanation for the good solution quality of Bubble-FOS/C by the optimization of the relaxed partitioning problem. However, due to the hardness of the problem, the question arises how expressive the relaxation-based analysis is and if more rigorous results in terms of approximation guarantee are possible. As mentioned before, for the case $k = 2$ polynomial time algorithms with polylogarithmic approximation guarantee are known even for the balanced case. This might give hope to find similar guarantees for Bubble-FOS/C and $k = 2$ as well. Yet, such a result cannot hold in the general case. Assume we are given a graph G with four clique subgraphs, two of which have size 3, the other two have size $(n-6)/2$. The subgraphs

are connected to each other as shown in Figure 4. While the optimal balanced cut (dashed orange line) includes only two edges, Bubble-FOS/C cuts $\Omega(n)$ edges, as explained in more detail in Observation 7.

Observation 7 *If $k = 2$ and z_1 and z_2 are the center vertices in Figure 4, then the solution computed by* `AssignPartition` *is such that $a_1, b_1 \in \pi_1$ and $a_2, b_2 \in \pi_2$. All other vertices in the large cliques have a fused load value of 0 and can be put in either part. Moreover,* `ScaleBalance` *is not successful in this scenario because different β_p lead to an unbalanced partition. Hence, a balanced cut produced by* Bubble-FOS/C *lies within the large cliques, and $\Omega(n)$ edges are cut.*

Recent results show that approximating balanced k-partitioning remains hard even for restricted graph classes such as trees [15].

4.2 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. As Fiedler's classical result [16] about spectral bipartitioning (but by using a proof approach similar to Grady's [20, p. 54f.]), we state that at least one part in a partition $\{\pi_1, \pi_2\}$ computed by Bubble-FOS/C is connected. For the proof we use Proposition 1 and construct load-increasing paths from each vertex v of one part to v's center. Note that all FOS/C vectors used in this section refer to the steady state.

Theorem 8 *If the graph $G = (V, E, \omega)$ is connected, at least one of $k = 2$ parts computed by* Bubble-FOS/C *on G is connected.*

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 $d^{(A)} := d_1^{(A)} - d_2^{(A)}$ are negative except for the entry corresponding to z_1. This means according to Remark 3 that all non-center vertices act as load-consuming sinks. Hence, using the arguments of Proposition 1 and Lemma 3, there must be a path $P = (v = v_1, v_2, \ldots, v_l = z_1)$ from every vertex $v \in \pi_1$ to z_1 on which the

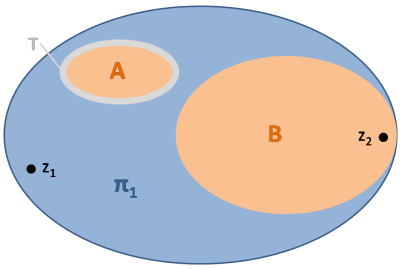

Fig. 5 Sketch of the situation assumed in Theorem 10.

load increases, i. e., $[w]^S_{v_i} < [w]^S_{v_{i+1}}$ 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. Since we included `ScaleBalance` by considering the scale values β, the statement holds for BUBBLE-FOS/C as well. □

Now we tighten the result for all connected *vertex-transitive* graphs (for a definition of vertex-transitivity see Section 3.4), where both parts are shown to be connected. Two well-known vertex-transitive classes are torus graphs and hypercubes (cf. Definition 8), which are important network topologies.

We continue with a simple, yet important observation following from a result by Alon and Spencer [1, p. 151].

Observation 9 *For all unweighted vertex-transitive graphs $G = (V, E)$ and all $u, v \in V$ it holds that $[w]^u_u = [w]^v_v$.*

Theorem 10 *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 $\pi_1 = \{u \in V \mid [w]^{z_1}_u \geq [w]^{z_2}_u\}$ and $\pi_2 = \{u \in V \mid [w]^{z_1}_u < [w]^{z_2}_u\}$. Then, π_1 and π_2 are connected components in G.*

Proof Recall from Definition 6 that the random walk measure *hitting time* $H[u, v]$ between vertices u and v is the expected time step ≥ 0 in which a random walk starting in u visits v for the first time. By using Theorem 1 we know that $\frac{1}{\alpha}([w]^v_u - [w]^v_v) = \delta(H[v, v] - H[u, v])$. Moreover, it holds for any two vertices $u, v \in V(G)$ of a vertex-transitive graph G that $H[u, v] = H[v, u]$ [29, Corollary 2.6]. Assume now for the sake of contradiction that π_2 is not connected. In this case there exists a vertex-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 vertices in π_2. Assume without loss of generality that $z_2 \in B$, as shown in Figure 5. Then for each vertex $a \in A$ we obtain $[w]^{z_2}_a > [w]^{z_1}_a$ i. e., $[w]^{z_2}_a - [w]^{z_2}_{z_2} > [w]^{z_1}_a - [w]^{z_1}_{z_1}$, which can be transformed to $H[z_2, z_2] - H[a, z_2] > H[z_1, z_1] - H[a, z_1]$ and finally $H[a, z_1] > H[a, z_2]$.

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 vertex visited in time step t by a random walk, and let $\mathcal{F}_u(x)$ 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 $\tau_a(T) := \min_{t \in \mathbb{N}_0}\{X^{(t)} \in T \mid X^{(0)} = a\}$ and let $\tau_{a,T}(z_1) := \min_{t \in \mathbb{N}_0}\{X^{(t)} = z_1 \mid X^{(0)} = a\} - \tau_a(T)$. By using conditional expectations ($\mathbb{E}[Y] = \sum_z \mathbf{Pr}[Z = z] \cdot \mathbb{E}[Y \mid Z = z]$) [21], we obtain $H[a, z_1] = \mathbb{E}[\tau_a(z_1)] = \mathbb{E}[\tau_a(T) + \tau_{a,T}(z_1)] = \sum_{x \in T} \mathbf{Pr}[\mathcal{F}_a(x)] \cdot \big(\mathbb{E}[\tau_a(T) + \tau_{a,T}(z_1) \mid \mathcal{F}_a(x)]\big)$, which is transformed by using the linearity of conditional expectations into

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

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

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

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 proof that π_1 is always connected is done in the same way, only 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. After analyzing the disturbed diffusive similarity measure computed by FOS/C in some detail, we have provided theoretical evidence for the aforementioned 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 shown two results on the connectedness of parts, 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 partitioning 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.

Acknowledgments. The authors thank Christoph Buchheim, Burkhard Monien, Peter Sanders, and Christian Schulz for helpful discussions.

References

1. N. Alon and J. H. Spencer. *The Probabilistic Method*. J. Wiley & Sons, 2nd edition, 2000.
2. R. Andersen, F. R. K. Chung, and K. J. Lang. Local graph partitioning using pagerank vectors. In *Proceedings of the 47th Annual Symposium on Foundaions of Computer Science (FOCS'06)*, pages 475–486, 2006.
3. R. Andersen and Y. Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09)*, pages 235–244, New York, NY, USA, 2009. ACM.
4. K. Andreev and H. Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.
5. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming. Theory and Algorithms*. John Wiley, second edition, 1993.
6. P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrich, and T. Sauerwald. Randomized diffusion for indivisible loads. In *Proceedings of the 22nd Annual Symposium on Discrete Algorithms (SODA'11)*, pages 429–439, 2011.
7. N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 1993.
8. C. Chevalier and F. Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(6-8):318–331, 2008.
9. R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. Parts I and II. *Proceedings of the National Academy of Sciences*, 102(21):7426–7437, 2005.
10. G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Parallel and Distributed Computing*, 7:279–301, 1989.
11. I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.
12. P. Diaconis, R. L. Graham, and J. A. Morrison. Asymptotic analysis of a random walk on a hypercube with many dimensions. *Random Structures and Algorithms*, 1(1):51–72, 1990.

13. R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(7):789–812, 1999.
14. P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Math. Assoc. of America, 1984.
15. A. E. Feldmann and L. Foschini. Balanced partitions of trees and applications. In *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, pages 100–111, 2012.
16. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.
17. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
18. C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer-Verlag, April 2001.
19. G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 3rd edition, 1996.
20. L. Grady. *Space-Variant Computer Vision: A Graph-Theoretic Approach*. PhD thesis, Boston University, Boston, MA, 2004.
21. G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, 3rd edition, 2001.
22. 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.
23. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.
24. H. Kaufmann and H. Pape. Clusteranalyse. In L. Fahrmeir, A. Hamerle, and G. Tutz, editors, *Multivariate statistische Verfahren*. Walter de Gryter & Co., 2nd edition, 1996.
25. J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.
26. B. W. Kernighan and S. Lin. An efficient heuristic for partitioning graphs. *Bell Systems Technical Journal*, 49:291–308, 1970.
27. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1992.
28. S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.
29. L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdös is Eighty*, 2:1–46, 1993.
30. M. Meila and J. Shi. A random walks view of spectral segmentation. In *8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.
31. H. Meyerhenke. *Disturbed Diffusive Processes for Solving Partitioning Problems on Graphs*. PhD thesis, Universität Paderborn, 2008.
32. H. Meyerhenke. Beyond good shapes: Diffusion-based graph partitioning is relaxed cut optimization. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC'10), Part II*, volume 6507 of *LNCS*, pages 387–398. Springer-Verlag, 2010.
33. 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.
34. H. Meyerhenke, B. Monien, and S. Schamberger. Graph partitioning and disturbed diffusion. *Parallel Computing*, 35(10–11):544–569, 2009.
35. H. Meyerhenke and T. Sauerwald. Analyzing disturbed diffusion on networks. In *Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC'06)*, pages 429–438. Springer-Verlag, 2006.
36. B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Proceedings of Advances in Neural Information Processing Systems 18 (NIPS'05)*, 2005.

37. F. Pellegrini. A parallelisable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries. In *Proceedings of the 13th International Euro-Par Conference (EURO-PAR'07)*, volume 4641 of *Lecture Notes in Computer Science*, pages 195–204. Springer-Verlag, 2007.
38. Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load balancing schemes. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 694–705, 1998.
39. H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264, 2008.
40. M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationship to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning (ECML'04)*, pages 371–383, 2004.
41. S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.
42. 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.
43. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
44. The BlueGene/L Team. An overview of the BlueGene/L supercomputer. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 1–22. ACM, 2002.
45. N. Tishby and N. Slonim. Data clustering by markovian relaxation and the information bottleneck method. In *Proceedings of Advances in Neural Information Processing Systems 13 (NIPS)*, pages 640–646, 2000.
46. L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, June 1997.
47. U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2000.
48. S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.
49. C. Walshaw. The graph partitioning archive. `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`, 2010. Last access: 22 Dec 2010.
50. C. Xu and F. C. M. Lau. *Load Balancing in Parallel Computers*. Kluwer, 1997.
51. L. Yen, D. Vanvyve, F. Wouters, F. Fouss, M. Verleysen, and M. Saerens. Clustering using a random-walk based distance measure. In *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN'05)*, pages 317–324, 2005.
52. 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)*, pages 1057–1064. MIT Press, 2001.