Congestion and Almost Invariant Sets in Dynamical Systems*

Michael Dellnitz[1] and Robert Preis[1,2]

[1] Department of Mathematics and Computer Science, Universität Paderborn, Paderborn, Germany, {dellnitz,robsy}@uni-paderborn.de
[2] Computer Science Research Institute, Sandia National Laboratories**, Albuquerque, United States of America, rpreis@sandia.gov

Abstract. An *almost invariant set* of a dynamical system is a subset of state space where typical trajectories stay for a long period of time before they enter other parts of state space. These sets are an important characteristic for analyzing the macroscopic behavior of a given dynamical system. For instance, recently the identification of almost invariant sets has successfully been used in the context of the approximation of so-called chemical *conformations* for molecules.
In this paper we propose new numerical and algorithmic tools for the identification of the number and the location of almost invariant sets in state space. These techniques are based on the use of *set oriented* numerical methods by which a graph is created which models the underlying dynamical behavior. In a second step graph theoretic methods are utilized in order to both identify the number of almost invariant sets and for an approximation of these sets. These algorithmic methods make use of the notion of *congestion* which is a quantity specifying bottlenecks in the graph. We apply these new techniques to the analysis of the dynamics of the molecules Pentane and Hexane. Our computational results are compared to analytical bounds which again are based on the congestion but also on spectral information on the transition matrix for the underlying graph.

1 Introduction

The dynamical behavior of a real world system can consist of motions on different time scales. For instance, a molecule can stay close to a certain geometric configuration in position space for quite a long time while exhibiting high frequency oscillations around this configuration. In this case the high frequency oscillations are irrelevant and it is desirable to identify the corresponding macroscopic structures for the given molecule. In general – that is, for an arbitrary dynamical

* Research is supported by the Priority Program 1095 of the Deutsche Forschungsgemeinschaft.
** Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

system – such macroscopic structures in state space are called *almost invariant sets*. In this article we propose a new combination of numerical and graph theoretic tools for both the identification of the number and the location of almost invariant sets in state space.

The crucial observation for the developments in this article is the following. If the underlying dynamics can be represented by a graph then the almost invariant sets are subgraphs which show a high coupling inside them and only a low coupling to other parts of the graph. Thus, paths beginning in a subgraph representing an almost invariant set will, with high probability, remain in the set for a long period of time. Having this point of view the identification of almost invariant sets boils down to the identification of corresponding appropriate cuts in the representing graph. Using this idea the topic of this paper is to develop a combination of numerical and graph theoretical methods for the identification of decompositions of phase space into regions which are almost invariant. Moreover it is in general a non-trivial task to identify the number of almost invariant sets and we will also attack this problem in this paper.

As indicated above the main application we have in mind is the identification of chemical *conformations* for molecules, see e.g. [9, 40]. However, in this paper we are not going to address the specific analytical and numerical problems which arise due to the fact that molecular dynamics is modeled via Hamiltonian systems. Rather this present work has to be viewed as a first step towards the efficient combination of graph theory and the numerical analysis for general dynamical systems.

Our topic is closely related to the graph partitioning problem in a natural way. In general this problem consists of the task to partition the vertices of a graph into a prescribed number of parts such that certain conditions on the number of edges between the parts are optimized. An enormous amount of work has been done to analyze the graph partitioning problem, to develop efficient methods for solving the problem and to design efficient code implementations in order to be used in several different applications. One important field of application is the efficient load balancing of graph-structured tasks on parallel computer systems.

It is our goal to use results from graph theory and graph algorithms and apply them in the context of dynamical systems. In this context a crucial question is to choose adequate cost functions which makes it necessary to modify some of the analytical results and to extend several existing algorithms. A first step in this direction has been made in [18] where greedy algorithms have been proposed for the identification of almost invariant sets. In our paper we extend this work significantly, in particular by proposing an algorithm which allows to approximate all the almost invariant sets by using the *congestion* of a graph.

A more detailed outline of the paper is as follows. In Section 2 we set the scene by stating the problem and by introducing the basic definitions. Moreover we give an overview of analytic bounds on the cost functions which are adequate for the underlying problem.

In Section 3 we briefly describe state-of-the-art graph partitioning methods. There are freely available graph partitioning tools such as our own tool PARTY. The problem we address here is different to the standard graph partitioning tools. Thus, we modify and extend some of the algorithms in PARTY. Furthermore, we implement some new algorithms in order to optimize the cost functions defined in Section 2.

In Section 4 we introduce the congestion of a graph. This quantity has been used in graph theory in order to specify bottlenecks within the graph. Since the computation of the congestion can be very costly we propose a couple of alternative approaches for approximating this quantity in a faster way.

Also the congestion of a graph can be used for bounding the magnitude of almost invariance of a subset of state space. In Section 5 we compare the bound derived by the congestion with the bound based on the spectrum of the graph and show that the congestion is highly suitable for identifying the quality of a specific decomposition. Moreover we quantify the increase in the congestion which is going to be observed if two invariant sets are loosely coupled and become almost invariant.

Finally, in Section 6 we use the congestion of a graph for the identification of an appropriate number of almost invariant sets for the given dynamical system. More precisely the congestion of a graph is used in combination with recursive bisection methods in order to decide whether or not a subgraph should be further partitioned. All the algorithms and in particular this partitioning strategy are illustrated by a number of examples.

2 Background

In this section we present the basic definitions and state the problem of finding almost invariant sets. The approach that we review here involves the optimization of certain cost functions and has already been presented in [18]. In addition to this review we state well known results on bounds of these cost functions by using spectral information on the corresponding transition matrix.

2.1 Basic Definitions and Problem Formulations

Dynamical System: Let $T : X \to X$ be a continuous map on a compact manifold $X \subset \mathbb{R}^n$. Then the transformation T defines a discrete time dynamical system of the form

$$x_{k+1} = T(x_k), \quad k = 0, 1, 2, \ldots$$

on X. In typical applications T is an explicit numerical discretization scheme for an ordinary differential equation on a subset of $\mathbb{R}^n$.

Measure: In the following we assume that μ is an **invariant measure** for the transformation T describing the statistics of long term simulations of the dynamical system. That is, the probability measure μ satisfies

$$\mu(A) = \mu(T^{-1}(A)) \quad \text{for all measurable } A \subset X.$$

Moreover we assume that μ is a unique so-called **SRB-measure** in the sense that this is the only invariant measure which is robust under small random perturbations. In other words μ is the only physically relevant invariant measure for the dynamical system T. For a precise definition of SRB-measures see e.g. [7].

For two sets $A_1, A_2 \subset X$ we define the **transition probability** ρ from A_1 to A_2 as

$$\rho(A_1, A_2) := \frac{\mu(A_1 \cap T^{-1}A_2)}{\mu(A_1)}$$

whenever $\mu(A_1) \neq 0$.

Almost Invariance: The transition probability $\rho(A, A)$ from a set $A \subset X$ into itself is called the **invariant ratio** of A. Heuristically speaking a set A is **almost invariant** if $\rho(A, A)$ is close to one, that is, if almost all preimages of the points in A are in A itself. Using this observation we state the optimization problem for the identification of almost invariant sets as follows.

Problem 1 (Almost Invariant). Let $p \in \mathbb{N}$. Find p pairwise disjoint sets $A_1, \ldots, A_p \subset X$ with $\bigcup_{1 \le i \le p} A_i = X$ such that

$$\frac{1}{p} \sum_{j=1}^{p} \rho(A_j, A_j) = \max!$$

Observe that in this optimization problem we have assumed that the number p of almost invariant sets is known. In this article we are going to address the question of how to identify an appropriate p in Section 6.

Box Collection: In general the infinite dimensional optimization problem 1 cannot be solved directly. Therefore we have to discretize the problem. Following [18] we suppose that we have a (fine) box covering of X consisting of d boxes $B_1, \ldots, B_d$ such that

$$X = \bigcup_{i=1}^{d} B_i.$$

In practice this box covering can be created by using a subdivision scheme as described in [6]. The optimization problem 1 is now reduced to all subsets which are finite unions of boxes, that is, on subsets of

$$\mathcal{C}_d = \left\{ A \subset X : A = \bigcup_{i \in I} B_i, I \subset \{1, \ldots, d\} \right\}.$$

For two sets $A_1^d, A_2^d \in \mathcal{C}_d$ the **box transition probability** ρ_d from A_1^d to A_2^d is nothing else but

$$\rho_d(A_1^d, A_2^d) := \rho(A_1^d, A_2^d) = \frac{\mu(A_1^d \cap T^{-1}A_2^d)}{\mu(A_1^d)}$$

assuming that $\mu(A_1^d) \neq 0$.

Transition Matrix / Markov Chain / Undirected Graph: The box collection $\mathcal{C}_d$ can be used to define a weighted transition matrix for our dynamical system. This is easily accomplished by defining the transition matrix $P \in \mathbb{R}^{d\times d}$ with $P_{ij} = \rho_d(B_i, B_j)$, $1 \leq i,j \leq d$. By this procedure we have constructed an approximation of our dynamical system via a finite state Markov chain: the boxes are the vertices and two vertices i and j are connected by an edge if the image of box i under the transformation T has a nontrivial intersection with box j. The transition probability is given by P_{ij}.

By construction the matrix P is stochastic, that is $\sum_{1\leq j\leq n} P_{ij} = 1$, and by our assumption on the SRB-measure μ it is reasonable to assume that P is also irreducible. Denote by $\mu^d \in \mathbb{R}^d$ the corresponding unique **stationary distribution** of P and by M the matrix which has the components of μ^d on its diagonal. Then it appears very unlikely that P is **reversible** in the sense that for all $1 \leq i,j \leq d$ we have

$$\mu_i^d P_{ij} = \mu_j^d P_{ji} \ .$$

However, as shown in [19] the value of the cost functions that we are going to consider remains unchanged if we are going to consider the reversibilization R of P given by the transformation

$$R := \frac{1}{2}(P + M^{-1}P^T M) \ .$$

From now on we denote by P a (not necessarily) reversible stochastic matrix and by R its corresponding reversible matrix.

The transition matrix R of a reversible Markov chain can be viewed as the adjacency matrix A of an undirected, edge-weighted graph $G = (V,E)$ with vertex set $V = \{v_1, \ldots, v_d\}$. Since R is reversible it follows that A is symmetric. The entry $A_{ij} = \mu_i^d R_{ij} = \mu_j^d R_{ji}$ denotes the edge weight between two vertices v_i and v_j. It is $\{v_i, v_j\} \in E$ if and only if $A_{ij} > 0$.

Partition of the Index Set / Graph: Let $I = \{1, \ldots, d\}$ be a set of indices. For two (not necessarily disjoint) subsets $S_1, S_2 \subset I$ let

$$\tilde{E}_{S_1,S_2}(P) = \sum_{i\in S_1, j\in S_2} \mu_i P_{ij}$$

and

$$E_{S_1,S_2}(P) = \frac{\tilde{E}_{S_1,S_2}(P) + \tilde{E}_{S_2,S_1}(P)}{2} \ .$$

Clearly, it is $\tilde{E}_{S_1,S_2}(R) = E_{S_1,S_2}(R)$ for a reversible matrix R. Furthermore, it is quite easy to see that

$$E_{S_1,S_2}(P) = E_{S_1,S_2}(R).$$

In this paper we focus on the two cases when S_1 is equal to S_2 and when S_1 is the complement of S_2. Clearly, for any $S \subset I$ it is $\tilde{E}_{S,S}(P) = E_{S,S}(P)$. For a subset $S \subset I$ let $\bar{S} = I \backslash S$ be the complement of S with respect to I.

We summarize the observations by defining

$$E_{S,S} := \tilde{E}_{S,S}(P) = E_{S,S}(P) = \tilde{E}_{S,S}(R) = E_{S,S}(R)$$
$$E_{S,\bar{S}} := E_{S,\bar{S}}(P) = \tilde{E}_{S,\bar{S}}(R) = E_{S,\bar{S}}(R)\ .$$

As discussed above, the transition matrix can be viewed as an undirected, edge-weighted graph $G = (V, E)$. Thus, a partition of the index set I can be viewed as a partition of $V = \{v_1, \ldots, v_d\}$, i.e. a set $S \subset I$ represents a subset $S_V \subset V$ with $S_V = \{v_i \in V : i \in S\}$. It is an easy task to rewrite the edge sets defined above and the cost functions defined below in graph notation.

We have now turned the optimization problem 1 into a problem of finding a minimal cut in an undirected graph G.

Cost Functions: By a slight abuse of notation we define $\mu(S) = \sum_{i \in S} \mu_i^d$ for a set $S \subset I$. For a set $S \subset I$ with $S \neq \emptyset$ and $S \neq I$ let

$$C_{int}(S) := \frac{E_{S,S}}{\mu(S)} \tag{1}$$

be the **internal cost of** S and let

$$C_{ext}(S) := \frac{E_{S,\bar{S}}}{\mu(S) \cdot \mu(\bar{S})} \tag{2}$$

be the **external cost of** S.

For $p \in \mathbb{N}$ let $\pi : I \to \{1, \ldots, p\}$ be a partition of I into $p \geq 2$ disjoint sets $I = S_1 \uplus \ldots \uplus S_p$ with $S_i \neq \emptyset$ for all $1 \leq i \leq p$. Define

$$C_{int}(\pi) := \frac{1}{p} \sum_{i=1}^{p} C_{int}(S_i) \tag{3}$$

to be the **internal cost of** π and let

$$C_{ext}(\pi) := \frac{\sum_{1 \leq i < j \leq p} E_{S_i,S_j}}{\prod_{i=1}^{p} \mu(S_i)} \tag{4}$$

be the **external cost of** π.

Remark 1. Both the internal and external costs are relevant cost functions. The following discussion shows that the minimization of the external cost is not equivalent to the maximization of the internal cost. In fact, the maximization of the internal cost favors in general parts that are on average very loosely coupled to the rest of the system. However, the size of these parts can in principle become very small. On the other hand the minimization of the external cost favors balanced weighting of the components.

One can easily construct examples for which the decomposition provided by the optimization of the internal cost is very different to the one given by the external cost and vice versa, but these examples have a rather artificial character. We have observed that in our examples, a decomposition with an optimal or almost optimal internal cost will also have an optimal or almost optimal external cost.

We can now state two optimization problems with respect to the internal and external costs. These have to be viewed as discrete versions of the continuous optimization problem 1.

Problem 2 (Internal Cost). Let $p \in \mathbb{N}$ and $I = \{1, \ldots, d\}$ be a set of indices. Find a partition $\pi : I \to \{1, \ldots, p\}$ of I into p disjoint sets $I = S_1 \uplus \ldots \uplus S_p$ such that

$$C_{int}(\pi) = \max!$$

Problem 3 (External Cost). Let $p \in \mathbb{N}$ and $I = \{1, \ldots, d\}$ be a set of indices. Find a partition $\pi : I \to \{1, \ldots, p\}$ of I into p disjoint sets $I = S_1 \uplus \ldots \uplus S_p$ such that

$$C_{ext}(\pi) = \min!$$

2.2 Spectral Bounds on the Cost Functions

In this section we state some bounds on the internal and external costs of a partition. These bounds are obtained via part of the spectrum of the underlying reversible Markov chain R.

Sinclair [43] introduces the **conductance** of an ergodic, reversible Markov chain. The bounds on the conductance can be transformed to bounds of the internal cost of a set. For a set $S \subset I$ let

$$\Phi(S) := \frac{E_{S,\bar{S}}}{\mu(S)} .$$

With $\mu(S) = E_{S,S} + E_{S,\bar{S}}$ we have

$$C_{int}(S) = 1 - \frac{E_{S,\bar{S}}}{\mu(S)} = 1 - \Phi(S).$$

The conductance Φ of a Markov chain is defined as

$$\Phi = \min_{0<|S|<d, \mu(S) \leq 1/2} \Phi(S) = \min_{0<|S|<d} \max\{\Phi(S), \Phi(\bar{S})\} .$$

This value is sometimes also called the **isoperimetric value** (see e.g. [1]).

Based on the eigenvalues $1 = \beta_1 \geq \beta_2 \geq \ldots \geq \beta_d \geq -1$ of R, Sinclair gave a proof for the bounds

$$1 - 2\Phi \leq \beta_2 \leq 1 - \frac{\Phi^2}{2} .$$

These bounds can be transformed into bounds on the internal cost of a set S, namely

$$1 - 2\sqrt{1 - \beta_2} \leq \max_{0<|S|<d} \min\{C_{int}(S), C_{int}(\bar{S})\} \leq \frac{\beta_2 + 1}{2} .$$

Remark 2. Recently Huisinga [25] has obtained very similar estimates in the situation where the underlying Markov process is induced by different stochastic models for molecular dynamics. The crucial difference to the material presented here lies in the fact that the estimates we are using are valid just for finite approximations of the underlying process.

A discussion of these bounds can also be found in [19].

Now consider the graph $G = (V, E)$ with adjacency matrix A as described in Section 2.1, that is in that notation $A = M \cdot R$. For a vertex $v_i \in V$ denote by

$$\deg(v_i) := \sum_{j=1}^{d} A_{ij}$$

the **degree** of v_i and let D be a diagonal matrix with $D_{ii} = \deg(v_i)$. Then the matrix $L = D - A$ is called the **Laplacian** of G. The **generalized Laplacian** of G is $\mathcal{L} = D^{-1/2} L D^{-1/2}$ with the entries

$$\mathcal{L}_{i,j} = \begin{cases} \frac{\deg(v_i) - A_{ii}}{\mu_i^d}, & \text{if } i = j \\ -\frac{A_{ij}}{\sqrt{\mu_i^d \cdot \mu_j^d}}, & \text{if } i \neq j \text{ and } \mu_i^d \cdot \mu_j^d > 0 \\ 0, & \text{otherwise} . \end{cases}$$

It can be shown that the eigenvalues of $\mathcal{L}$ are nonnegative. We denote them by $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_d$. In fact, it is $\lambda_i = 1 - \beta_i$ for $1 \leq i \leq d$. The multiplicity of 0 as an eigenvalue is equal to the number of connected components in the graph. By our irreducibility assumption on the underlying Markov chain we can conclude that $\lambda_2 > 0$.

We conclude, based on the work by Fiedler [16], that it can be shown that

$$\lambda_2 \leq C_{ext}(S) \leq \lambda_d . \tag{5}$$

Further discussion on spectral graph partitioning is to be found in e.g. [2, 4, 13, 31, 37, 43].

3 Algorithms for Graph Partitioning

In this section we focus on algorithms for partitioning the vertex set of a graph. A number of methods and tools for graph partitioning exist. However, they do not exactly optimize our cost functions mentioned in the previous section. Thus, in this section we give an overview of the most successful graph partitioning methods and implementations and point out the necessary modifications to such tools to optimize our cost functions.

For the remainder of this section we assume that we partition into a given number of p parts. In Section 6 we will show how to identify this number.

We want to calculate a partition of the vertex set V of a graph $G = (V, E)$ into p parts $V = S_1 \cup \ldots \cup S_p$ such that either one of our cost functions of equations (3) or (4) is optimized. However, the calculation of an optimal solution of either cost functions is **NP**-complete. Another widely discussed partitioning problem is to minimize the cut size $cut(\pi) = \sum_{1 \leq i < j \leq p} E_{S_i,S_j}$ of the partition π under the constraint that all parts have an equal (or almost equal) number of vertices. This problem is sometimes called *Balanced Partitioning Problem* and is **NP**-complete, even in the simplest case when a graph with constant vertex and edge weights is to be partitioned into two parts [20].

In recent years, efficient graph partitioning strategies have been developed for a number of different applications. Efficiency and generalizations of graph partitioning methods strongly depend on specific implementations. There are several software libraries, each of which provides a range of different methods. Examples are CHACO [22], JOSTLE [44], METIS [26], SCOTCH [35] or PARTY [38]. The goal of the libraries is to both provide efficient implementations and to offer a flexible and universal graph partitioning interface to applications. These libraries are designed to create solutions to the balanced partitioning problem. We first describe the most significant methods of these tools and then describe how we modified parts of the code in order to optimize the cost functions in the previous section.

The tool PARTY has been developed by the second author and we have used it for partitioning the graphs in this paper. PARTY, like other graph partitioning tools, follows the Multilevel Paradigm (Figure 1). The multilevel graph partitioning strategies have been proven to be very powerful approaches to efficient graph-partitioning [3, 21, 23, 27, 28, 34, 36, 39]. The efficiency of this paradigm is dominated by two parts: graph coarsening and local improvement.

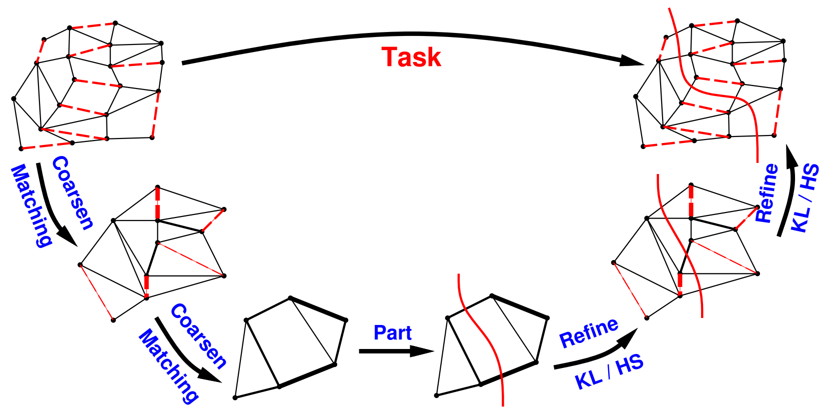

Fig. 1. The multilevel approach with several coarsening steps, a partitioning of the smallest graph and the local refinements.

The graph is coarsened down in several levels until a graph with a sufficiently small number of vertices is constructed. A single coarsening step between two levels can be performed by the use of graph matchings. A matching of a graph is a subset of the edges such that each vertex is incident to at most one matching edge. A matching of the graph is calculated and the vertices incident to a matching edge are contracted to a super-vertex. Experimental results reveal that it is important to contract those vertices which are connected via an edge of a high weight, because it is very likely that this edge does not cross between parts in a partition with a low weight of crossing edges. PARTY uses a fast approximation algorithm which is able to calculate a good matching in linear time [39].

PARTY stops the coarsening process when the number of vertices is equal to the desired number of parts. Thus, each vertex of the coarse graph is one part of the partition. However, it is also possible to stop the coarsening process as soon as the number of vertices is sufficiently small. Then, any standard graph partitioning method can be used to calculate a partition of the coarse graph.

Finally, the partition of the smallest graph is projected back level-by-level to the initial graph. The partition is locally refined on each level. Standard methods for local improvement are Kernighan/Lin [29] type of algorithms with improvement ideas from Fiduccia/Mattheyses [15]. An alternative local improvement heuristic is the Helpful-Set method [11] which is derived from a constructive proof of upper bounds on the bisection width of regular graphs [24, 32, 33].

As mentioned above, the tools are designed for solving the balanced graph partitioning problem. Thus, the optimization criteria is different from our cost functions of Section 2. The coarsening step of the multilevel approach does not consider the balancing of the weights of the super-vertices. It is the local refinement step which not only improves the partition locally but also balances the weights of the parts. Thus, we have to modify the local improvement part of the multilevel approach. We therefore modified the Kernighan/Lin implementation in PARTY such that it optimizes the cost-function C_{int}.

Overall, we use the algorithm of Figure 2 to calculate almost invariant sets.

Partition graph $G_0 = (V_0, E_0)$ into p parts

```
i = 0;
WHILE (|V_i| > p)
    calculate a graph matching M_i ⊂ E_i;
    use M_i to coarse graph G_i = (V_i, E_i) to a graph G_{i+1} = (V_{i+1}, E_{i+1});
    i := i + 1;
END WHILE
let π_i be a p-partition of G_i such that each vertex is one part;
WHILE (i > 0)
    i := i − 1;
    use M_i to project π_{i+1} to a p-partition π_i of G_i;
    modify the partition π_i on G_i locally to optimize C_int(π_i);
END WHILE
output π_0.
```

Fig. 2. Calculating almost invariant sets with the multilevel approach.

As an example to illustrate the partitioning we take the graph of a Pentane molecule from [9]. Figure 3 (left) shows the box collection and all transitions. As we will see in section 6, it is adequate to partition the graph into 5 or 7 parts. These are shown in Figures 3 (center/right). The resulting internal costs are .980 for the 5-partition and 0.963 for the 7-partition.

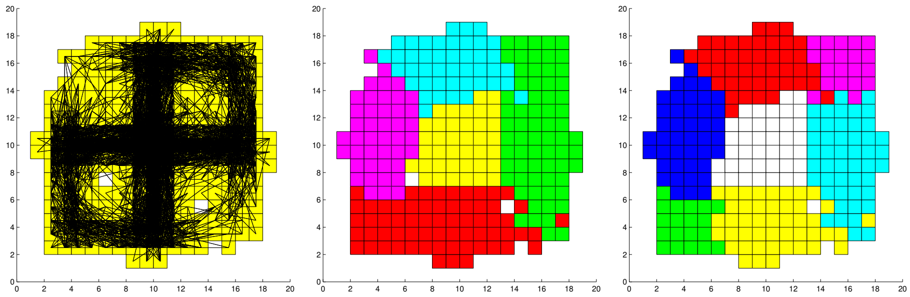

Fig. 3. Partition of the Pentane300. Left: Graph with all transitions. Center: Partition into 5 parts with $C_{int} = .980$. Right: Partition into 7 parts with $C_{int} = .963$.

4 The Congestion

The congestion of a graph can be used to spot bottlenecks in the graph. It is based on an all-to-all routing or a multi-commodity flow on the graph (see e.g. [30, 43]). We are going to make use of the congestion both for bounding the external cost of subgraphs S (Section 5) and for the identification of the number p of almost invariant sets (Section 6).

4.1 The Definition

Let $G = (V, E)$ be an undirected graph with vertex weights $vw(v_i) = \mu_i^d$, for $v_i \in V$ and edge weights $ew(\{v_i, v_j\}) = A_{ij}$ for $\{v_i, v_j\} \in E$ $(i, j = 1, 2, \ldots, d)$.

Let $v_s \in V$ be a source and $v_t \in V$ be a target vertex in G. Let $c \in \mathbb{R}$ be a **single-commodity** which has to flow from v_s to v_t along the edges in G. A **single-commodity flow** f of the commodity c on G is a function $f : V \times V \to \mathbb{R}$ such that

i. $f(v, w) = 0$ for all $\{v, w\} \notin E$ (flow on edges only),
ii. $f(v, w) = -f(w, v)$ for all $v, w \in V$ (symmetry),
iii. $\sum_{w \in V} f(v, w) = 0$ for all $v \in V \backslash \{v_s, v_t\}$ (flow conservation) and
iv. $\sum_{w \in V} f(v_s, w) = \begin{cases} -\sum_{w \in V} f(v_t, w) = c & \text{if } v_s \neq v_t \text{ (from source to target)} \\ 0 & \text{if } v_s = v_t \text{ (within source/target)} \end{cases}$

Now let $c_{s,t} \in \mathbb{R}$, $1 \leq s,t \leq d$, be **multi-commodities**. Each commodity $c_{s,t}$ has to flow from the source $v_s \in V$ to the target $v_t \in V$ along the edges in G. A **multi-commodity flow** F of the commodities $c_{s,t}$ on G is a function $F : V \times V \times V \times V \to \mathbb{R}$ such that $F(s,t,v,w)$ is a single-commodity flow of the commodity $c_{s,t}$ on G for each fixed pair $s,t \in V$.

Remark 3. In our context we use the multi-commodities

$$c_{s,t} = vw(v_s) \cdot vw(v_t)$$

for each pair of vertices $v_s, v_t \in V$ for the definition of the congestion.

The flow F causes a traffic on the edges in G. For an edge $\{v,w\} \in E$ the **edge congestion of** $\{v,w\}$ **in** F on G is

$$cong(\{v,w\},F) = \frac{\sum_{1\leq s,t\leq d} |F(s,t,v,w)|}{ew(\{v,w\})} \ .$$

We will see in the following that high edge congestion occurs at bottlenecks of the graphs.

The **flow congestion of** F on G is

$$cong(F) = \max_{\{v,w\}\in E} cong(\{v,w\},F) \ ,$$

and finally the **congestion** of the graph G is

$$cong(G) = \min_{F \text{ is multi-commodity flow}} cong(F) \ .$$

Intuitively it is clear that a low congestion of the graph ensures that the graph is very compact. For illustration we list the congestion for a couple of specific graphs. The proofs can be found in [30].

Example 1.

a. For the specific case where the graph G is a line consisting of n vertices connected by $n-1$ edges it is obvious that the congestion is given by

$$cong(G) = 2\left\lceil \frac{n}{2} \right\rceil \cdot \left\lfloor \frac{n}{2} \right\rfloor .$$

b. Suppose that the graph G is given by a z-dimensional torus with n^z grid points (n even). Then the congestion is

$$cong(G) = \frac{n^{z+1}}{4}.$$

In particular for the circle we have $z=1$ and obtain $cong(G) = \frac{n^2}{4}$.

c. For the hypercube G of dimension k the congestion is given by

$$cong(G) = 2^k.$$

d. For the complete graph with n vertices – that is a graph where there is an all-to-all coupling between the vertices – the congestion is given by

$$cong(G) = 2.$$

Further discussion of the congestion and a listing of congestions of further graph classes can be found in e.g. [30, 39].

4.2 Heuristics for Calculating a Flow

The computation of the congestion can be costly. We will see in Section 5.1 that the congestion can be used to bound the external cost of a partition. However, that bound holds for the congestion of any flow. Thus, a sub-optimal flow produces a sub-optimal, but still valid bound. In this section we discuss heuristics for calculating a flow with a small flow congestion.

There are some hints of how to construct a flow with a small flow congestion. Clearly, cycles in the flow should be avoided. Furthermore, the flow should - not always, but primarily - go along shortest paths between the pairs of vertices. Here, the length of a path is the sum of the reciprocal values of the edge weights along the path.

A straightforward method is to send the flow along shortest paths only. If more than one shortest paths exist, the flow value can be split among them. If the edge weights are constant, all shortest paths can be calculated in time $O(|V| \cdot |E|)$. This can be done by $|V|$ independent Breath-First searches in time $O(|E|)$ each. If the edge weights are non-negative, all shortest paths can be calculated in time $O(|V| \cdot (|V| \cdot \log |V| + |E|))$, e.g. with $|V|$ runs of the single-source shortest path Dijkstra algorithm using Fibonacci heaps. We refer to [5] for a deeper discussion of shortest paths algorithms.

A different method is to consider n commodities at a time. For each vertex v_s, $1 \leq s \leq n$, consider the commodities $c_{s,t}$, $1 \leq t \leq n$, i.e. all commodities with v_s as the source. For each source v_s we calculate a flow F_s which transports all commodities from v_s to all other vertices, i.e. it replaces n single-commodity flows such that $F_s(v, w) = \sum_{t=1}^{n} F(s, t, v, w)$. F_s is a single-source, multiple-destination commodity flow. Definitions (i.) and (ii.) for the single commodity flow (Section 4.1) remain unchanged whereas the definitions of (iii.) and (iv.) are replaced by

v. $\sum\limits_{w \in V} F_s(v_t, w) = -c_{s,t}$ for all $t \neq s$ (from source s to target t) and

vi. $\sum\limits_{w \in V} F_s(v_s, w) = \sum\limits_{t=1}^{n} c_{s,t} - c_{s,s}$ (from source s to all targets t except to source s itself).

It is left to show how we calculate a single-source flow F_s. We use algorithms from diffusion load balancing on distributed processor networks for this task, see e.g. [10, 12, 14]. Here, the problem is to balance the work load in a distributed processor network such that the volume of data movement is as low as possible.

We use these algorithms in the setting that the vertex v_s models a processor with load $\sum_{t=1}^{n} c_{s,t}$ and all other vertices model a processor with no load. Furthermore, the processors are heterogeneous with a capacity of $c_{s,t}$ for processor v_t [14]. The diffusion algorithms calculate a balancing flow such that each vertex/processor v_t gets a load of $c_{s,t}$. That is exactly what we need in our context. The resulting balancing flow has a nice property: it is minimal in the l_2-norm, i.e. the diffusion algorithms minimize the value $\sqrt{\sum_{1\leq v,w\leq n} |F_s(v,w)|}$. This ensures that there are no cycles in any flow F_s. Furthermore, the flows are not restricted along shortest paths and can avoid high traffic along shortest paths. However, the flows are still favored to be along reasonably short paths. Thus, it is expected that the overall edge congestion of the resulting flow is reasonably small and that the flow congestion is close to the congestion of the graph.

The PARTY library includes efficient code of a variety of diffusion algorithms. We use them to calculate the single-source, multiple-destination flows for each source s and then add up the values to get the multi-commodity flow. We will see in Section 5.2 that the resulting flow congestion is very small.

4.3 An Example: Pentane

We now illustrate the meaning of the congestion in the context of dynamical systems. For this we use data from [9] in which a reversible stochastic matrix has been generated by simulations of an ODE-based model of the n-Pentane molecule. Those computations made use of two dihedral angles as the essential variables. The graph corresponding to this dynamical system is illustrated in Figure 4 (left) and the edges with low (middle) resp. high (right) congestion are also shown.

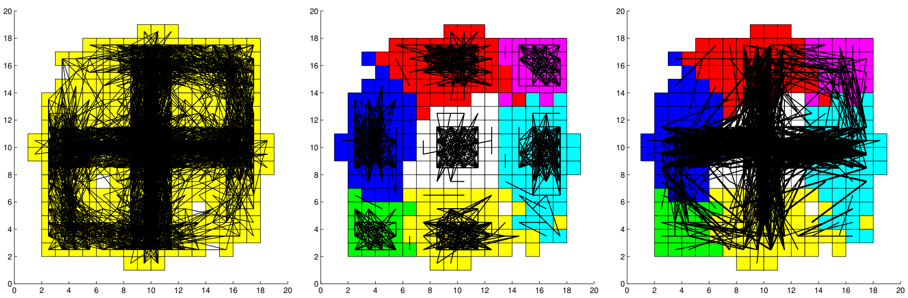

Fig. 4. Congestion of the Pentane. Left: all transitions. Center: only transitions with a low congestion. Right: only transitions with a high congestion.

Furthermore, the boxes in Figure 4 (center) and Figure 4 (right) are colored according to a partition into seven almost invariant sets. It will be shown in Section 6, in which way the number of almost invariant sets has been determined. This decomposition has been calculated using the graph partitioning tool PARTY. Note that the partitioning was calculated independently from the congestion.

It can be observed that – as expected – edges with low congestion can mainly be found inside the almost invariant sets. On the other hand edges between different almost invariant sets have a large congestion. Thus, a high congestion indicates that there are at least two regions in the phase space which are only loosely coupled. As we will see in Section 6, this observation is the basis for using the congestion as an identifier for the number of almost invariant sets which have to be approximated.

5 The Congestion and Almost Invariant Sets

In this section we show how the congestion can be used in order to obtain quantitative estimates on the magnitude of almost invariance for a given set. Moreover we derive estimates on the decrease of the congestion if two almost invariant sets are decoupled. Obviously these estimates have to depend on the structure of the graph inside the almost invariant sets. These results have to be viewed as a first step in using graph theoretic notions for the identification of dynamical properties in multiscale dynamical systems.

5.1 The Congestion Bound

The congestion can be used to derive a lower bound on $C_{ext}(S)$ for any $S \subset I$ – recall the definitions (1) and (2) of the two different cost functions. As before, see Section 4.1, we use multi-commodities $c_{s,t} = vw(v_s) \cdot vw(v_t)$ for each source $v_s \in V$ and each destination $v_t \in V$.

Clearly, at least $\mu(S) \cdot \mu(\bar{S})$ units have to cross the cut from S to $\bar{S}$ and the same amount has to cross in the opposite direction. However, a congestion of *cong* ensures that there are at most $cong \cdot E_{S,\bar{S}}$ units crossing the cut. Thus, $2 \cdot \mu(S) \cdot \mu(\bar{S}) \leq cong \cdot E_{S,\bar{S}}$, and we have shown the following inequality

$$C_{ext}(S) \geq \frac{2}{cong} \ . \tag{6}$$

Obviously, a high and tight lower bound can only be achieved with a small congestion. Although the computation of the congestion can be done in polynomial time, it remains to be very costly. Nevertheless, the congestion can be approximated by the congestion of any flow. Heuristics for calculating a small congestion were discussed in Section 4.2 above.

Remark 4. It is known that an optimal setting for the lower bound (6) occurs if all flows are along shortest paths and the amount of flow is equal on all edges (with respect to their weights). Unfortunately, this 'optimal' case does not exist for arbitrary graphs. Moreover, it can be shown that there are graphs for which such an 'optimal' setting exists, but for which the equation (6) is not tight [39]. Nevertheless, the comparisons in the following section show that the congestion is highly suitable for the type of graphs which are induced by the underlying dynamical systems.

Further discussion of lower bounds based on different variations of multi-commodity flows can be found in [41].

5.2 Comparison of Bounds

In this section we experimentally compare the lower bounds on the external costs of a partition. We compare the bound based on the congestion (see (6)) and the bound based on the second eigenvalue (see (5)).

As mentioned in the previous section, there are examples of graphs for which the congestion bound is not tight although there exists an optimal setting for the calculation of the multi-commodity flow (Remark 4). Furthermore, the examples for which the spectral bound is tight are also very limited, see e.g. [2]. A comparison of the two bounds on some structured graphs can be found in [39]. It is very interesting to know how the two bounds perform on the examples resulting from dynamical systems. We use the heuristic described in the previous section to calculate a flow f which should result in a very low congestion. A comparison of bounds for some examples is shown in Table 1.

The names in the first column in Table 1 refer to the following dynamical systems:

- The first three examples *Pentane200, Pentane300* and *Pentane600* are different models of the n-Pentane molecule with two dihedral angels [9]. An example of *Pentane300* can be found in Figure 3 and in Figure 4.
- *Hexane* is created from data for a model of the Hexane molecule. Again this data is taken from [9]. (An illustration of *Hexane* can be found in Figure 10 below.)
- The example *Logistic* refers to the *Logistic Map* which is given by the transformation
$$T(x) = bx(1-x)$$
on the interval $[0,1]$. We have chosen the parameter value $b=4$.
- *Lorenz* represents the celebrated *Lorenz system* which is given by the ordinary differential equation
$$\begin{aligned}\dot{x} &= -\sigma x + \sigma y\\ \dot{y} &= \rho x - y - xz\\ \dot{z} &= -\beta z + xy.\end{aligned}$$
We have chosen the standard set of parameter values $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$.
- All the examples *Exx* refer to simulations for different energy levels of the Hamiltonian system with the potential
$$V(q) = \left(\frac{3}{2}q_1^4 + \frac{1}{4}q_1^3 - 3q_1^2 - \frac{3}{4}q_1 + 3\right) \cdot (2q_2^4 - 4q_2^2 + 3)\ .$$

Table 1. Example Graphs. The number $|V|$ of vertices is given as the total number of vertices and the number of vertices with nonvanishing discrete measure. The *selfloops* indicate transitions within a box and are listed with total number and total weight. The congestion $cong(f)$ is calculated from a heuristic flow f as described in Section 4.2.

Graph	$\|V\|$	$\|E\|$	selfloops		λ_2	$cong(f)$	$\frac{2}{cong(f)}$
Pentane200	400/181	1127	56	.244	.001591	1168.441628	.001712
Pentane300	400/255	3328	96	.170	.014069	139.939697	.014292
Pentane600	400/330	13016	171	.068	.103959	23.558954	.084893
Hexane	729/397	4258	295	.625	.010174	208.911692	.009573
Logistic	512/512	1501	1	.002	.067301	20.728386	.096486
Lorenz	5025/4842	37284	8	.000	.019535	58.842596	.033989
E32	82/80	676	41	.021	.003164	487.721222	.004101
E33	90/90	731	39	.022	.011117	113.961640	.017550
E34	92/92	760	37	.022	.015016	73.138238	.027345
E35	92/92	766	38	.022	.017796	62.112404	.032200
E36	96/96	795	34	.022	.020236	46.650151	.042872
E37	102/102	858	34	.022	.021510	38.152007	.052422
E38	104/104	892	34	.020	.023039	44.240662	.045207
E39	104/104	917	34	.020	.023933	42.678191	.046862
E40	106/106	937	37	.020	.024319	44.689798	.044753
E41	106/106	948	36	.020	.024771	44.692308	.044750
E42	116/116	1024	35	.020	.024545	45.899904	.043573
E43	124/124	1077	33	.019	.025073	46.176803	.043312
E44	128/128	1139	33	.019	.025813	44.759473	.044683
E45	128/128	1174	32	.019	.025643	44.865978	.044577
Butterfly	888/888	4752	16	.001	.003175	168.883748	.011842

For instance, E45 refers to the case where the sum of the kinetic and the potential energy is equal to $xx = 45$. This example has previously been analyzed in [8]. An illustration of the potential function is shown in Figure 5 (left) and the invariant measure for the corresponding dynamical behavior is shown in Figure 5 (right).

In Figure 6 we illustrate computations of the congestion for the example E45.

– Finally the example *Butterfly* is based on a simulation of the Hamiltonian system given in (7). An illustration of the corresponding graph can be found in Figure 7.

The results of Table 1 show that the congestion bound $\frac{2}{cong(f)}$ is superior to the spectral bound λ_2 in most of the examples. Furthermore, we approximated the congestion $cong(G)$ with a flow congestion $cong(f)$ calculated by the heuristic described above. The computational results indicate that our heuristic approach produces a flow congestion which is at least close to the correct congestion of the graph.

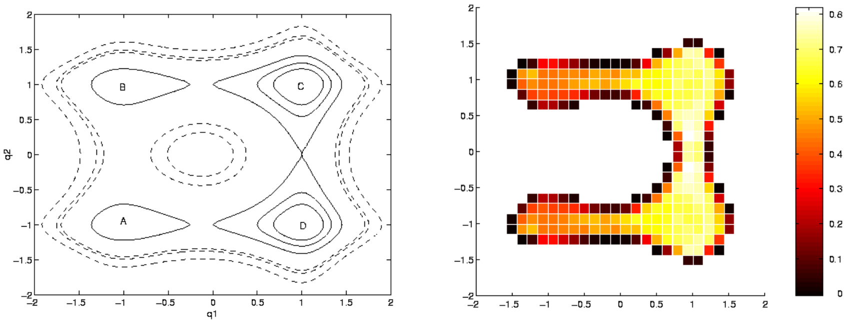

Fig. 5. Left: Contour plot for the potential function of Exx. Right: Typical invariant measure for Exx.

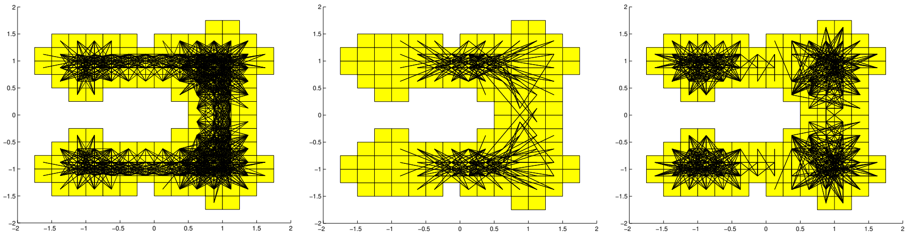

Fig. 6. Example E45 with all edges (left), edges with congestion at least 20 (center) and edges with congestion at most 10 (right).

5.3 Decoupling Almost Invariant Sets

In this section we investigate the change in the congestion when almost invariant sets are decoupled. For simplicity we restrict our attention to graphs where the weight of the vertices are all equal. More refined investigations will be done elsewhere.

For motivational purposes we compute the congestion for the following Hamiltonian system

$$\begin{aligned} \dot{q} &= p \\ \dot{p} &= -4q(q^2-1) \ . \end{aligned} \tag{7}$$

This system has a saddle point in $(q,p) = (0,0)$ and two homoclinic orbits connecting this saddle point with itself. Figure 7 (left) shows the discretized system with all the transitions in the corresponding graph. Figure 7 (center/right) shows a partition of phase space into three parts. We calculated a multi-commodity flow f in the graph with the heuristics described in Section 4.2. The congestion of our flow is $cong(f) = 166.31$. Figure 7 (center) shows only the transitions with a low congestion and Figure 7 (right) shows only the transitions with a high congestion.

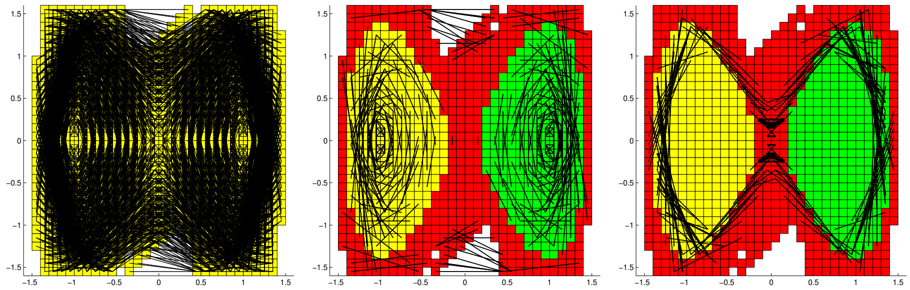

Fig. 7. Partition and congestion of Butterfly. Left: all transitions. Center: only transitions with a low congestion. Right: transitions with a high congestion. The phase space is partitioned into 3 parts.

A more detailed inspection shows that the edges with the highest edge congestion are precisely those which are closest to the saddle point. In fact, this is precisely what has to be expected. Thus, roughly speaking we have the following rule of thumb:

High congestion identifies saddle points.

On the other hand one has to expect that the partition of phase space into almost invariant sets – in particular for Hamiltonian systems – takes place at precisely such saddle points. We now examine the change of the magnitude of the congestion if two invariant sets exhibiting complicated dynamics are coupled and become almost invariant. For this the following observation is crucial.

Proposition 1. *Let G be a graph and denote by H the graph consisting of two copies of G which are connected by m additional edges. Let n be the number of vertices in G. Then we have the following estimate on the congestion of H:*

$$cong(H) \geq \frac{2n^2}{m}$$

Proof. The result follows from the fact that at least $2n^2$ commodities have to cross the m connecting edges. □

Let us apply Proposition 1 to the particular cases which have already been listed in the Examples 1

Corollary 1.

a. *Suppose that G is a z-dimensional torus with n vertices. Then $cong(G) = \frac{n^{(z+1)/z}}{4}$ and the coupling of two such graphs leads to an increase of the congestion by a factor of at least $\frac{8}{m}n^{(z-1)/z}$.*
b. *Suppose that G is a k-dimensional hypercube with $n = 2^k$ vertices. Then $cong(G) = n$ and the coupling leads to an increase of the congestion by a factor of at least $\frac{2n}{m}$.*

c. *If G is a complete graph with n vertices then cong$(G) = 2$. Thus, the congestion of the coupled graph will increase by a factor of at least $\frac{n^2}{4m}$.*

A couple of remarks are in order.

Remark 5.

a. Observe that in applications typically m will be very small in comparison to the number of vertices of G.
b. From the application point of view Corollary 1 can be interpreted in the following way. Suppose that the dynamics inside two almost invariant sets can adequately be modeled by a z-dimensional torus, a k-dimensional hypercube or a complete graph. Then the congestion will roughly drop by a factor of $\frac{8}{m}n^{(z-1)/z}$, by a factor of $\frac{2n}{m}$ or by a factor of $\frac{n^2}{4m}$ if two almost invariant sets are separated by a cut through m edges. Thus, this result has to be viewed as a tool for the identification of the internal dynamics inside almost invariant sets.
c. Observe that the torus and the hypercube are realistic models for the description of complicated dynamics inside the almost invariant sets. However, in the computations in the following section we sometimes even observe a more drastic decrease in the congestion than predicted by Corollary 1. This indicates that in those cases a model with an (almost) all-to-all coupling for the dynamical behavior, i.e. the complete graph, would be even more appropriate.

6 Identification of the Number of Almost Invariant Sets

We want to partition the system into almost invariant sets. In this section we discuss the problem of identifying the number of almost invariant sets.

Again, we cover the relevant dynamics by a box collection and deal with the graph $G = (V, E)$ represented by the transition matrix as described in Section 2. We want to determine a number $p \in \mathbb{N}$ such that there is a partition π of V into p parts $V = S_1 \cup \ldots \cup S_p$ with the internal cost of π being high. In order to explore the macroscopic dynamics of the system, π should not contain a part which could further be partitioned into sub-parts while maintaining or improving the internal cost. In other words, any further partition of any part of π should lower the internal cost of π. We will refer to this characteristic of a part as being *compact*, i.e. we want to find all compact almost invariant parts in the graph. This leads us to a strategy of how to determine the number of compact parts. It can be phrased as a general method:

Recursively bisect the graph into 2 parts until all parts are compact.

One needs to solve two tasks in order to follow this strategy. Firstly, one has to define the compactness of a graph and has to be able to decide whether a part is compact or not. Secondly, one needs a bisection method which bisects a non-compact part into two sub-parts such that the bisection preserves compact areas within a sub-part, i.e. the cut does not intersect compact areas of the graph.

Many graph partitioning methods were originally designed as graph bisection heuristics. Some of them are easy generalizable to partition the graph into any number p of parts. However, some methods are very difficult to generalize. It is a common approach to use bisection heuristics recursively to get a partition into more than two parts. Several experiments have been performed to compare the direct p-partitioning approach with the recursive bisection approach. An analyses on this comparison is to be found in [42]. We will use the methods described in Section 3 to recursively calculate bisections of a graph.

There are several graph theoretical areas which define some sort of compactness of a part. Thus, there are several criteria which can be used as indicators for a compact part. As some examples, a part is called compact if

1. it has a large internal cost (equation 1) and there is no bisection of the part into 2 sub-parts with a similar or higher internal cost.
2. the sub-graph representing the part has a large gap between the first and the second eigenvalue (e.g. [17]).
   In this case, equation (5) indicates that the external cost of any further cut is large and, thus, a further bisection would decrease the internal cost.
3. the Markov chain representing the part has a high conductance.
   The conductance is defined in Section 2.2. It is also referred to as the *isoperimetric value* of the graph (see e.g. [1]). Clearly, if the part has a high conductance, any further bisection of the graph would lead to a high connectivity of the two sub-parts.
4. **small congestion**.
   As stated in Section 5.1, a small congestion ensures a high external cost of any sub-part. Thus, any further bisection will decrease the internal cost of the partition.

Clearly, if a graph is considered compact with respect to one measure, it is also considered compact with respect to the other measures.

These characteristics are not only important in the analyses of dynamical systems, but they are also important in the explicit construction of compact graphs. Compact graphs have been studied for the construction of efficient topologies of communication networks for multi-processor systems. The main objective here is to construct a network such that the information flow in the network is as fast as possible and that there is no bottleneck within the network which could slow down the information flow.

We now illustrate the identification of the number of almost invariant sets on some examples.

Figure 8 illustrates the partitioning of the Pentane molecule from [9] which we already used in Section 3. As we can observe in the left most picture, the first bisection results in one part of 43 boxes and a very low congestion of 0.88. However, the other part consisting of 212 boxes has a high congestion of 168.82 . We continue to bisect parts with a congestion of more than 5. Thus, after a total of 4 bisection levels we get a partition into 7 parts and the highest congestion of any part is 3.67 .

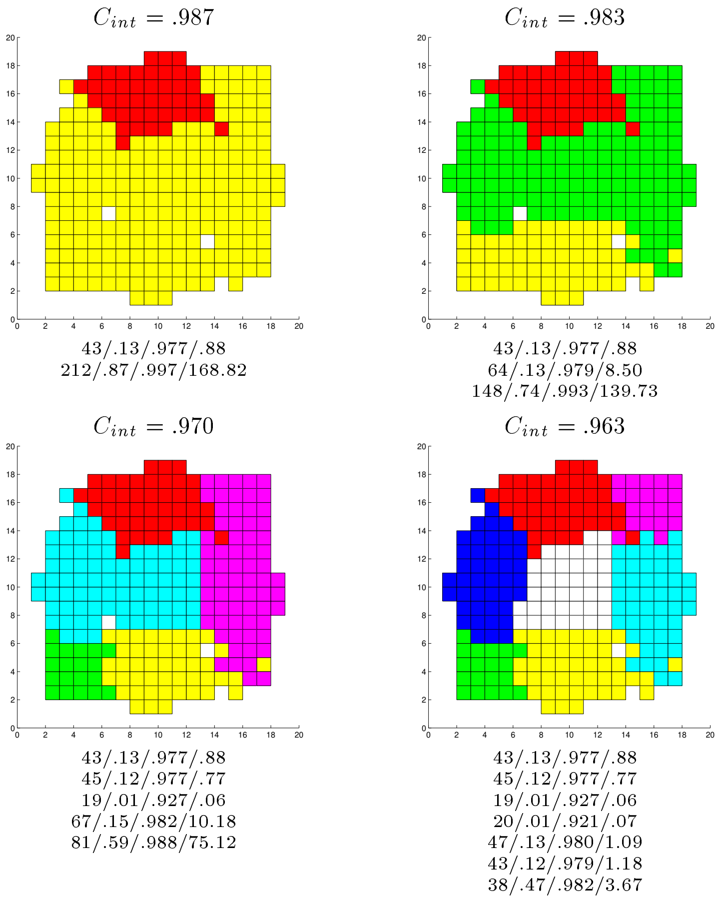

Fig. 8. Recursive bisection of the Pentane300. The values indicate: number of boxes / invariant measure / internal cost C_{int} / congestion of subgraph. The graph has 255 vertices and a congestion of 139.67.

A different method to derive the number of reasonable parts is to look at the sorted list of the eigenvalues of the Laplacian. A large gap in the value of two neighbored eigenvalues λ_i and λ_{i+1} is an indication to partition the graph into i parts. Figure 9 (left) shows the 20 smallest eigenvalues of the Pentane200, Pentane300 and Pentane600 graphs. As one can observe, there is a large gap between λ_7 and λ_8. This suggests to partition the Pentane300 into 7 parts and is conform with our calculation above using the congestion.

Figure 9 (right) shows the 20 smallest eigenvalues of the Laplacian of 'Hexane'. The large gap between λ_{17} and λ_{18} suggests to partition the graph into 17 parts. Table 2 shows the recursive bisection of Hexane by using the congestion as bisection criteria. Again, we recursively bisected the parts of the partition until all parts have a congestion of at most 5. This was achieved after 5 levels

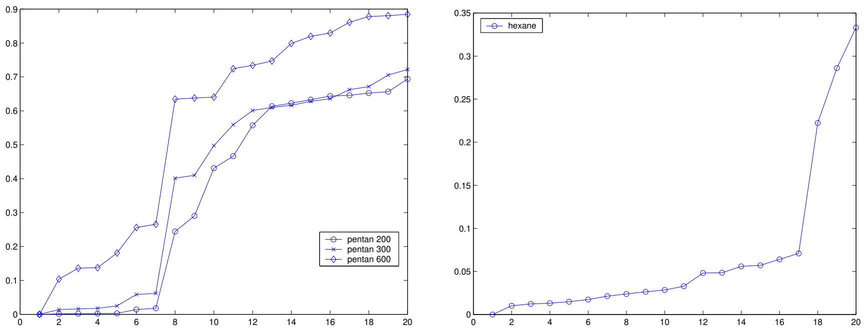

Fig. 9. Laplacian eigenvalues of Pentane200, Pentane300 and Pentane600 (left) and Hexane (right).

of recursion. The result is a partition into 17 parts with 1.65 being the highest congestion of any part. As we can see, the lowest internal cost of any of the 17 parts is .93.

Figure 10 shows a partition of Hexane into 17 parts. The example is 3-dimensional and the 3-dimensional boxes are drawn with a smaller size in order to be able to get a better visualization.

7 Conclusion

The macroscopic behavior of a dynamical system can be analyzed by an identification of almost invariant sets in phase space. The calculation of almost invariant sets requires an appropriate partitioning of the phase space. We have addressed this topic by an application of graph partitioning theory in this context and use graph partitioning algorithms to calculate almost invariant sets of the system. We have shown that the congestion of the system is an appropriate measure of the quality of a partition. Furthermore, the congestion can be used to identify the number of almost invariant sets.

We view the results of this paper as an important step towards the application of graph theoretical results and algorithms in the field of dynamical systems exhibiting multiscale dynamics. So far, only a limited number of aspects are considered. For instance, it would be an interesting question to develop methods for the identification of appropriate graph models representing the dynamics inside the almost invariant sets. Then one could obtain even stronger statements about the change in congestion by an application of Proposition 1.

Finally the algorithmic methods presented eventually have to be combined with multilevel subdivision techniques which are used for the construction of the box coverings. This way a more efficient identification of the macroscopic structure of the dynamical behavior can be accomplished.

Table 2. Partitioning of Hexane into 17 parts with recursive bisection using the congestion as bisection criteria. The values in the first 6 columns indicate: number of boxes / congestion. The last two columns list the invariant measure and the internal cost of each of the resulting parts.

level 0	level 1	level 2	level 3	level 4	level 5	$\mu(S)$	$C_{int}(S)$
397/208.91	162/174.51	84/145.80	47/7.61	15/.05		.009	.95
				32/.63		.086	.98
			37/.71			.087	.98
		78/5670.31	51/1044.50	19/.12		.023	.98
				32/216.01	24/.21	.021	.98
					8/.01	.001	.93
			27/1.65			.320	.98
	235/506.16	90/139.35	38/214.00	10/.02		.002	.94
				28/.68		.092	.98
			52/8.23	17/.05		.010	.94
				35/.73		.090	.98
		145/141.11	46/5.88	29/.76		.085	.98
				17/.04		.009	.94
			99/30.13	59/16.68	26/.18	.026	.97
					33/1.32	.100	.98
				40/569.80	21/.12	.010	.94
					19/.15	.030	.98

Acknowledgments

We thank Christof Schütte and his group at the Freie Universität Berlin for the very fruitful cooperation. In particular our treatment of the examples of the Pentane and Hexane molecules would not have been possible if Alexander Fischer and Wilhelm Huisinga would not have provided the corresponding data. We also thank Anthony Baker for patient discussions on the contents of this article.

References

1. N. Alon. On the edge-expansion of graphs. *Combinatorics, Probability and Computing*, 6:145–152, 1997.
2. S.L. Bezroukov, R. Elsässer, B. Monien, R. Preis, and J.-P. Tillich. New spectral lower bounds on the bisection width of graphs. In *Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS 1928, pages 23–34, 2000.
3. N. Bouhmala. Impact of different graph coarsening schemes on the quality of the partitions. Technical Report RT98/05-01, University of Neuchatel, Department of Computer Science, 1998.
4. F.R.K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional conference series in mathematics*. American Mathematical Society, 1997.
5. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

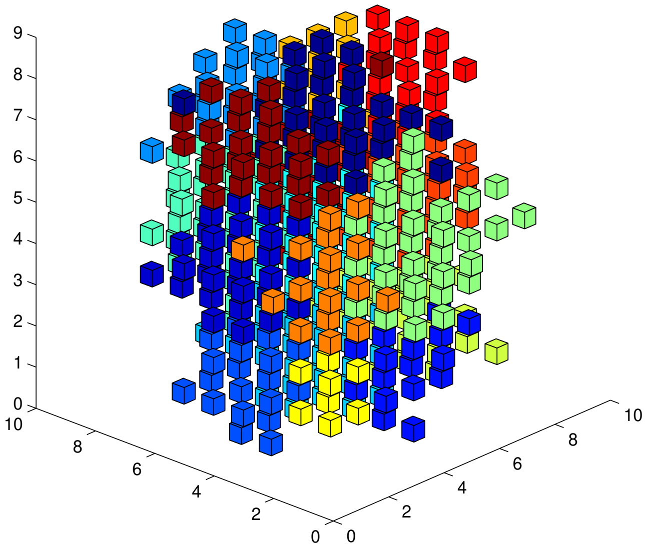

Fig. 10. Partition of Hexane into 17 parts

6. M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75:293–317, 1997.
7. M. Dellnitz and O. Junge. On the approximation of complicated dynamical behavior. *SIAM J. Numer. Anal.*, 36(2):491–515, 1999.
8. P. Deuflhard, M. Dellnitz, O. Junge, and Ch. Schütte. Computation of essential molecular dynamics by subdivision techniques. In Deuflhard et al., editor, *Computational Molecular Dynamics: Challenges, Methods, Ideas*, LNCSE 4, 1998.
9. P. Deuflhard, W. Huisinga, A. Fischer, and Ch. Schütte. Identification of almost invariant aggregates in reversible nearly uncoupled markov chains. *Lin. Alg. Appl.*, 315:39–59, 2000.
10. R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(7):789–812, 1999.
11. R. Diekmann, B. Monien, and R. Preis. Using helpful sets to improve graph bisections. In D.F. Hsu, A.L. Rosenberg, and D. Sotteau, editors, *Interconnection Networks and Mapping and Scheduling Parallel Computations*, volume 21 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 57–73. AMS, 1995.
12. R. Elsässer, A. Frommer, B. Monien, and R. Preis. Optimal and alternating-direction loadbalancing schemes. In P. Amestoy et al., editor, *Euro-Par'99 Parallel Processing*, LNCS 1685, pages 280–290, 1999.
13. R. Elsässer, T. Lücking, and B. Monien. New spectral bounds on k-partitioning of graphs. In *Proc. of the Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 255–270, 2001.

14. R. Elsässer, B. Monien, and R. Preis. Diffusive load balancing schemes on heterogeneous networks. In *12th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 30–38, 2000.
15. C.M. Fiduccia and R.M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proc. IEEE Design Automation Conf.*, pages 175–181, 1982.
16. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical J., Praha*, 25(100):619–633, 1975.
17. A. Fischer, Ch. Schütte, P. Deuflhard, and F. Cordes. Hierarchical uncoupling-coupling of metastable conformations. In Schlick and Gan, editors, *Proc. of the 3rd International Workshop on Algorithms for Macromolecular Modeling*, LNCSE 24, 2002.
18. G. Froyland and M. Dellnitz. Detecting and locating near-optimal almost-invariant sets and cycles. Technical report, DFG-Priority Program 1095: "Analysis, Modeling and Simulation of Multiscale Problems", 2001.
19. G. Froyland and M. Dellnitz. μ almost-invariant sets and adaptive boundary refinement. manuscript, 2003.
20. M.R. Garey and D.S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* Freemann, 1979.
21. A. Gupta. Fast and effective algorithms for graph partitioning and sparse matrix reordering. *IBM J. of Research and Development*, 41:171–183, 1997.
22. B. Hendrickson and R. Leland. The chaco user's guide: Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1994.
23. B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proc. Supercomputing '95.* ACM, 1995.
24. J. Hromkovič and B. Monien. The bisection problem for graphs of degree 4 (configuring transputer systems). In Buchmann, Ganzinger, and Paul, editors, *Festschrift zum 60. Geburtstag von Günter Hotz*, pages 215–234. Teubner, 1992.
25. W. Huisinga. *Metastability of Markovian systems.* Phd thesis, Freie Universität Berlin, 2001.
26. G. Karypis and V. Kumar. *METIS Manual, Version 4.0.* University of Minnesota, Department of Computer Science, 1998.
27. G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. of Parallel and Distributed Computing*, 48:96–129, 1998.
28. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. on Scientific Computing*, 20(1), 1999.
29. B.W. Kernighan and S. Lin. An effective heuristic procedure for partitioning graphs. *The Bell Systems Technical J.*, pages 291–307, 1970.
30. F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, 1992.
31. B. Mohar. Some applications of laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications*, NATO ASI Ser. C 497, pages 225–275, 1997.
32. B. Monien and R. Diekmann. A local graph partitioning heuristic meeting bisection bounds. In *8th SIAM Conf. on Parallel Processing for Scientific Computing*, 1997.
33. B. Monien and R. Preis. Bisection width of 3- and 4-regular graphs. In *26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, LNCS 2136, pages 524–536, 2001.
34. B. Monien, R. Preis, and R. Diekmann. Quality matching and local improvement for multilevel graph-partitioning. *Parallel Computing*, 26(12):1609–1634, 2000.
35. F. Pellegrini. SCOTCH 3.1 user's guide. Technical Report 1137-96, LaBRI, University of Bordeaux, 1996.

36. R. Ponnusamy, N. Mansour, A. Choudhary, and G.C. Fox. Graph contraction for mapping data on parallel computers: A quality-cost tradeoff. *Scientific Programming*, 3:73–82, 1994.
37. A. Pothen, H.D. Simon, and K.P. Liu. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. on Matrix Analysis and Applications*, 11(3):430–452, 1990.
38. R. Preis. *The PARTY Graphpartitioning-Library, User Manual - Version 1.99.* Universität Paderborn, Germany, 1998.
39. R. Preis. *Analyses and Design of Efficient Graph Partitioning Methods.* Heinz Nixdorf Institut Verlagsschriftenreihe, 2000. Dissertation, Universität Paderborn, Germany.
40. Ch. Schütte. *Conformational Dynamics: Modelling, Theory, Algorithm, and Application to Biomolecules.* Habilitation thesis, Freie Universität Berlin, 1999.
41. N. Sensen. Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows. In *Proc. European Symposium on Algorithms (ESA)*, LNCS 2161, pages 391–403, 2001.
42. H.D. Simon and S.-H. Teng. How good is recursive bisection? *SIAM J. on Scientific Computing*, 18(5):1436–1445, 1997.
43. A. Sinclair. *Algorithms for Random Generation & Counting: A Markov Chain Approach.* Progress in Theoretical Computer Science. Birkhäuser, 1993.
44. C. Walshaw. *The Jostle user manual: Version 2.2.* University of Greenwich, 2000.