Journal of Computational Physics 228 (2009) 8548–8565

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Optimal block-tridiagonalization of matrices for coherent charge transport

Michael Wimmer [*], Klaus Richter

Institut für Theoretische Physik, Universität Regensburg, 93040 Regensburg, Germany

A R T I C L E   I N F O

Article history:
Received 17 June 2008
Received in revised form 12 February 2009
Accepted 3 August 2009
Available online 12 August 2009

MSC:
05C50
05C78

PACS:
72.10.Bg
02.70.−c
02.10.Ox

Keywords:
Coherent quantum transport
Recursive Green's function algorithm
Block-tridiagonal matrices
Matrix reordering
Graph theory

A B S T R A C T

Numerical quantum transport calculations are commonly based on a tight-binding formulation. A wide class of quantum transport algorithms require the tight-binding Hamiltonian to be in the form of a block-tridiagonal matrix. Here, we develop a matrix reordering algorithm based on graph partitioning techniques that yields the optimal block-tridiagonal form for quantum transport. The reordered Hamiltonian can lead to significant performance gains in transport calculations, and allows to apply conventional two-terminal algorithms to arbitrarily complex geometries, including multi-terminal structures. The block-tridiagonalization algorithm can thus be the foundation for a generic quantum transport code, applicable to arbitrary tight-binding systems. We demonstrate the power of this approach by applying the block-tridiagonalization algorithm together with the recursive Green's function algorithm to various examples of mesoscopic transport in two-dimensional electron gases in semiconductors and graphene.

1. Introduction

If the dimensions of a device become smaller than the phase coherence length l_ϕ of charge carriers, classical transport theories are not valid any more. Instead, carrier dynamics is now governed by quantum mechanics, and the wave-like nature of particles becomes important. In general, the conductance/resistance of such a device does not follow Ohm's law.

In the regime of coherent quantum transport, the Landauer–Büttiker formalism [1–3] relates the conductance G of a device to the total transmission probability T of charge carriers through the device,

$$G = \frac{e^2}{h} T = \frac{e^2}{h} \sum_{mn} |t_{mn}|^2, \tag{1}$$

where t_{mn} is the transmission amplitude between different states with transverse quantum numbers n and m in the left and right lead, respectively. A state with a given transverse quantum number n is also called *channel n*.

The problem of calculating the conductance is thus reduced to calculating scattering eigenfunctions ψ for a given energy E:

[*] Corresponding author. Tel.: +49 941 943 2016; fax: +49 941 943 4382.
E-mail address: Michael.Wimmer@physik.uni-regensburg.de (M. Wimmer).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.08.001

$$(E - H)\psi = 0, \tag{2}$$

where H is the Hamiltonian of the system. Alternatively, the transmission probability can be extracted from the retarded Green's function G^{r} that obeys the equation of motion

$$(E - H)G^{\mathrm{r}} = \mathbb{1}. \tag{3}$$

The Fisher–Lee relation [4,5] then allows to calculate the transmission (t_{mn}) and reflection (r_{nm}) amplitudes from G^{r}. In its simplest form, the Fisher–Lee relation reads

$$t_{mn} = -i\hbar\sqrt{v_m v_n}\int_{C_{\mathrm{R}}} dy \int_{C_{\mathrm{L}}} dy' \phi_m(y) G^{\mathrm{R}}(\mathbf{x},\mathbf{x}')\phi_n(y') \tag{4}$$

and

$$r_{mn} = \delta_{mn} - i\hbar\sqrt{v_m v_n}\int_{C_{\mathrm{L}}} dy \int_{C_{\mathrm{L}}} dy' \phi_m(y) G^{\mathrm{R}}(\mathbf{x},\mathbf{x}')\phi_n(y'), \tag{5}$$

where v_n is the velocity of channel n and the integration runs over the cross-section $C_{\mathrm{L}}(C_{\mathrm{R}})$ of the left (right) lead.

The Landauer–Büttiker formalism can also deal with multi-terminal systems, but is restricted to linear response, i.e. small bias voltages. In the general case including external bias, the conductance can be calculated using the non-equilibrium Green's function formalism (see e.g. [6]).

Except for particularly simple examples, solving Eqs. (2) and (3) exactly is not possible, and therefore a numerical computation is often the method of choice. Instead of solving directly a differential equation with its continuous degrees of freedom, such as the Schrödinger equation, numerical computations are usually only attempted within a discrete basis set. The differential equation is replaced by a set of linear equations, and the Hamiltonian H can be written as a matrix. Very often, only few of the matrix elements H_{ij} are nonzero. Such *tight-binding* representations of the Hamiltonian are ubiquitous in quantum transport calculations and can arise from finite differences [7–9], from the finite element method [10], from atomic orbitals in empirical tight-binding [11–13] or Kohn–Sham orbitals within density functional theory [14–16].

When describing transport, the systems under consideration are open and thus extend to infinity. As a consequence, the tight-binding matrix H is infinite-dimensional. However, the conductance calculation can be reduced to a finite problem by partitioning the system into a finite scattering region attached to leads that extend to infinity, as schematically depicted in Fig. 1(a). For the case of two-terminals, the matrix H can be written as

$$H = \begin{pmatrix} H_{\mathrm{L}} & V_{\mathrm{LS}} & 0 \\ V_{\mathrm{SL}} & H_{\mathrm{S}} & V_{\mathrm{SR}} \\ 0 & V_{\mathrm{RS}} & H_{\mathrm{R}} \end{pmatrix}, \tag{6}$$

where $H_{\mathrm{L(R)}}$ is the (infinite) Hamiltonian of the left (right) lead, H_{S} is the Hamiltonian of the scattering region and of finite size. The matrices $V_{\mathrm{SL}} = V^{\dagger}_{\mathrm{LS}}$ and $V_{\mathrm{SR}} = V^{\dagger}_{\mathrm{RS}}$ represent the coupling between the scattering region and the left and right lead, respectively.

In order to reduce the problem size, it is useful to introduce the retarded self-energy $\Sigma^{\mathrm{r}} = \sum_{i=L,R} V_{\mathrm{S}i} g^{\mathrm{r}}_i V_{i\mathrm{S}}$, where $g^{\mathrm{r}}_{\mathrm{L(R)}}$ is the surface Green's function of the left (right) lead, i.e. the value of the Green's function at the interface of the lead disconnected from the scattering region. Then, the Green's function $G^{\mathrm{r}}_{\mathrm{S}}$ of the scattering region can be calculated as [17,18]

$$G^{\mathrm{r}}_{\mathrm{S}} = (E - H_{\mathrm{S}} - \Sigma^{\mathrm{r}})^{-1} \tag{7}$$

reminiscent of Eq. (3) but with an effective Hamiltonian $H_{\mathrm{S}} + \Sigma^{\mathrm{r}}$ of finite size. This treatment is easily extended to multi-terminal systems.

Note that it suffices to know the surface Green's function of the (semi-)infinite leads, as in a tight-binding Hamiltonian the matrices V_{SL} and V_{SR} have only few nonzero entries. For simple systems, the surface Green's function can be calculated ana-

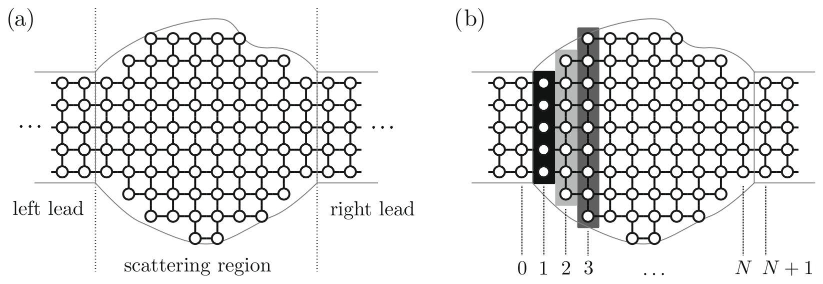

Fig. 1. (a) Schematic view of a finite difference grid in a two-terminal transport setup. (b) Natural ordering of grid points yielding a block-tridiagonal matrix structure. The different matrix blocks are marked in alternating shades of grey.

lytically [17,18], whereas in more complex situations it must be computed numerically, either by iteration [19,20], or by semi-analytical formulas [12,21,16].

The original infinite-dimensional problem has thus been reduced to a finite size matrix problem that can, in principle, be solved straight-forwardly on a computer. However, for any but rather small problems, the computational task of the direct inversion in Eq. (7) is prohibitive. Therefore, for two-terminal transport, many algorithms make use of the *sparsity* of the Hamiltonian matrix in tight-binding representation—in particular that this matrix can be written in block-tridiagonal form:

$$H = \begin{pmatrix}
\ddots & & & & & & & & & & \\
& H_\mathrm{L} & V_\mathrm{L} & & & & & & & & \\
& V_\mathrm{L}^\dagger & H_\mathrm{L} & H_{0,1} & & & \ddots & & & & \\
& & H_{1,0} & H_{1,1} & H_{1,2} & & & 0 & & & \\
& & & H_{2,1} & H_{2,2} & H_{2,3} & & & \ddots & & \\
& & & & H_{3,2} & \ddots & & & & & \\
& & \ddots & & & & \ddots & H_{N-1,N} & & & \\
& & & 0 & & & H_{N,N-1} & H_{N,N} & H_{N,N+1} & & \\
& & & & \ddots & & & H_{N+1,N} & H_\mathrm{R} & V_\mathrm{R} & \\
& & & & & & & & V_\mathrm{R}^\dagger & H_\mathrm{R} & \\
& & & & & & & & & & \ddots
\end{pmatrix}, \tag{8}$$

where the index L(R) denotes the blocks in the left (right) lead, $1, \ldots, N$ the blocks within the scattering region, and $0(N+1)$ the first block in the left (right) lead.

This block-tridiagonal form of the Hamiltonian is the foundation of several quantum transport algorithms for two-terminal systems. The transfer matrix approach applies naturally to block-tridiagonal Hamiltonians, but becomes unstable for larger systems. However, a stabilized version has been developed by Usuki et al. [22,23]. In the decimation technique [24,25], the Hamiltonian of the scattering region is replaced by an effective Hamiltonian between the two leads by eliminating internal degrees of freedom. The contact block reduction method [26] calculates the full Green's function of the system using a limited set of eigenstates. The recursive Green's function (RGF) technique [27–29] uses Dyson's equation to build up the system's Green's function block by block. It has also been adapted to Hall geometries with four-terminals [30] and to calculate non-equilibrium densities [31,32]. Furthermore, the RGF algorithm has been formulated to be suitable for parallel computing [33].

The block-tridiagonal form of H arises naturally, for example, in the method of finite differences, when grid points are grouped into vertical slices according to their x-coordinates, as shown in Fig. 1(b). In fact most of the above mentioned quantum transport algorithms are based on this *natural ordering*. Because of this, these algorithms are in general restricted to systems with two collinear leads. In a more complicated system, e.g. with non-collinear or multiple leads, a block-tridiagonal form is not obvious or involves only a few, very large blocks. In these cases, a direct application of the above mentioned transport algorithms is either very inefficient or even impossible. These restrictions will be lifted in the course of this work.

Of course, there are also other transport techniques not directly based on the block-tridiagonal form of the Hamiltonian matrix, such as extracting the Green's function from wave packet dynamics [34]. Still, such algorithms are not as widely used as the large class of algorithms that are directly based on the block-tridiagonal form of the Hamiltonian. In order to illustrate the typical computational tasks of this class of algorithms, we briefly explain, as a representative example, the RGF algorithm.

The RGF technique is based on Dyson's equation $G^\mathrm{r} = G_0^\mathrm{r} + G_0^\mathrm{r} V G^\mathrm{r}$ (see e.g [29]), where G^r denotes the Green's function of the perturbed system, G_0^r that of the unperturbed system and V the perturbation. Using this equation, the system is built up block by block, as depicted in Fig. 2. Let $G^{\mathrm{r},(i)}$ denote the Green's function for the system containing all blocks $\geqslant i$. Then, at energy E, the Green's function $G^{\mathrm{r},(i-1)}$ is related to $G^{\mathrm{r},(i)}$ by

$$G^{\mathrm{r},(i-1)}_{i-1,i-1} = \left(E - H_{i-1,i-1} - H_{i-1,i} G^{\mathrm{r},(i)}_{i,i} H_{i,i-1}\right)^{-1} \tag{9}$$

and

$$G^{\mathrm{r},(i-1)}_{N+1,i-1} = G^{\mathrm{r},(i)}_{N+1,i} H_{i,i-1} G^{\mathrm{r},(i-1)}_{i-1,i-1}. \tag{10}$$

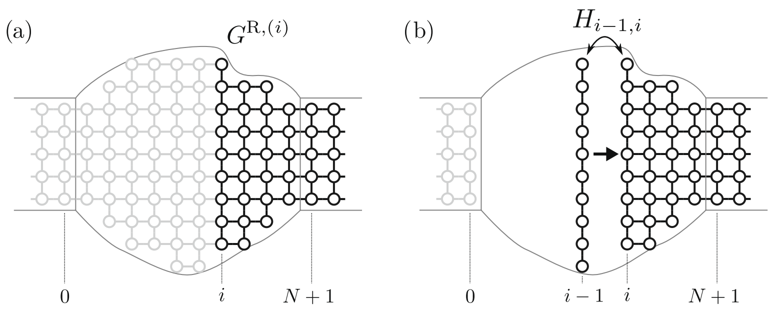

Fig. 2. Schematic depiction of the recursive Green's function algorithm: (a) the Green's function $G^{r,(i)}$ contains all blocks $\geqslant i$ and (b) the Green's function $G^{r,(i-1)}$ is obtained by adding another matrix block.

Starting from $G^{r,(N+1)}_{N+1,N+1} = g^r_R$, the surface Green's function of the right lead, N slices are added recursively, until $G^{r,(1)}$ has been calculated. The blocks of the Green's function of the full system necessary for transport are then given by

$$G^r_{0,0} = \left(\left(g^r{}_L \right)^{-1} - H_{0,1} G^{r,(1)}_{1,1} H_{1,0} \right)^{-1} \tag{11}$$

and

$$G^r_{N+1,0} = G^{r,(1)}_{N+1,1} H_{1,0} G^r_{0,0}, \tag{12}$$

where g^r_L is the surface Green's function of the left lead. $G^r_{0,0}$ and $G^r_{N+1,0}$ are sufficient to calculate transmission and reflection probabilities via the Fisher–Lee relation, Eqs. (4) and (5).

Each step of the algorithm performs inversions and matrix multiplications with matrices of size M_i. Since the computational complexity of matrix inversion and multiplications scales as M_i^3, the complexity of the RGF algorithm is $\propto \sum_{i=0}^{N+1} M_i^3$. Thus, it scales linearly with the "length" N, and cubically with the "width" M_i of the system. This scaling also applies to most of the other transport algorithms mentioned above.

While for particular cases general transport algorithms, such as the RGF algorithm, cannot compete with more specialized algorithms, such as the modular recursive Green's function technique [35,36] that is optimized for special geometries, they are very versatile and easily adapted to two-terminal geometries—provided that the system has only two leads that are arranged collinearly.

The objective of this work is twofold: first, we intend to lift the discussed restrictions of the established quantum transport algorithms. We do this by bringing the Hamiltonian matrix H into a block-tridiagonal form suitable for quantum transport also for complex geometries, such as non-collinear leads or multi-terminal structures, that would otherwise need the development of specialized algorithms. Second, we improve the matrix H such that the block-tridiagonal form is *optimal* for transport. Although the block-tridiagonal structure of H, Eq. (8), that arises naturally in many problems appears to have a small "width" and thus seems to be quite suitable for transport algorithms, optimizing the matrix structure further may lead to significant speed-ups even in the two-terminal case, as we show below.

We achieve these goals by developing a matrix reordering algorithm that brings an arbitrary tight-binding matrix H into a block-tridiagonal form optimal for quantum transport. This algorithm is based on recursive bisection as well as graph and hypergraph partitioning techniques. To this end, the paper is organized as follows: in Section 2 we formulate the matrix reordering problem in the language of graph theory and develop the reordering algorithm. In Section 3 we apply this algorithm to various examples and investigate its performance and the performance of the RGF algorithm for the reordered Hamiltonian H. We conclude in Section 4.

2. Optimal block-tridiagonalization of matrices

2.1. Definition of the problem

2.1.1. Definition of the matrix reordering problem

As shown in the introduction, the typical runtime of transport algorithms is proportional to $\sum_{i=0}^{N+1} M_i^3$, and hence does depend on the particular block-tridiagonal structure of H. Therefore, the runtime of these algorithms can be improved in principle by conveniently reordering H with a permutation P,

$$H' = PHP^{-1}. \tag{13}$$

In order to quantify the typical performance of a transport algorithm for a given matrix structure, we define a weight $w(H)$ associated with a matrix H as

$$w(H) = \sum_{i=0}^{N+1} M_i^3, \tag{14}$$

where M_i is the size of block $H_{i,i}$. Optimizing the matrix for transport algorithms is then equivalent to minimizing the weight $w(H)$. Since $\sum_{i=0}^{N+1} M_i = N_{\text{grid}}$, where N_{grid} is the total number of grid points, $w(H)$ is minimal, if all M_i are equal, $M_i = N_{\text{grid}}/(N+2)$. Therefore, a matrix tends to have small weight, if the number N of blocks is large, and all blocks are equally sized. The numerical experiments of Section 3 will confirm that the weight $w(H)$ is a good measure of the actual performance of a quantum transport algorithm.

The reordering problem of the matrix H is thus summarized as follows:

Problem 2.1. *Matrix reordering problem*: Find a reordered matrix H' such that

1. $H'_{0,0}$ and $H'_{N+1,N+1}$ are blocks given by the left and right leads (as required by transport algorithms).
2. H' is block-tridiagonal ($H'_{i,j} \neq 0$, iff $j = i+1, i, i-1$).
3. The number N of blocks is as large as possible, and all blocks are equally sized.

In principle, this constrained optimization problem could be solved by generic optimization algorithms, such as *Simulated Annealing*. However, for larger problems the optimization could take much more time than the actual transport calculation, rendering the optimization process useless. It is therefore necessary to use heuristics especially designed for the problem at hand. To this end, we formulate the matrix reordering problem in the language of graph theory.

2.1.2. Mapping onto a graph partitioning problem

A *graph* $\mathcal{G}$ is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of *vertices* v and $\mathcal{E}$ a set of ordered pairs of vertices $(v_1, v_2) \in \mathcal{V} \times \mathcal{V}$. Such a pair is called an *edge*. A graph is called *undirected*, if for every edge $(v_1, v_2) \in \mathcal{E}$ also $(v_2, v_1) \in \mathcal{E}$. Two vertices v_1 and v_2 are called *adjacent*, if $(v_1, v_2) \in \mathcal{E}$. In order to simplify the notation, we will also consider a vertex v to be adjacent to itself. An edge (v_1, v_2) is said to be *incident* to a vertex v_i, iff $v_1 = v_i$ or $v_2 = v_i$.

There is a natural one-to-one correspondence between graphs and the structure of sparse matrices. For a given $n \times n$ matrix H, we define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $(v_i, v_j) \in \mathcal{E}$ iff the entry $H_{ij} \neq 0$. Furthermore, the diagonal of H is assumed to be nonzero (as every vertex v is adjacent to itself). This is an appropriate assumption for matrices H encountered in transport problems, as the quantum transport algorithms usually deal with $E - H$, where E is a number (for example, see Eq. (9)).

A graph thus stores information about the *structure* of a matrix, i.e. which entries are nonzero. It does not contain any information about the values of the respective entries, although these may be stored easily along with the graph. However, for the formulation of the quantum transport algorithms, only the block-tridiagonal form, i.e. the structure of the matrix, is relevant. Hermitian matrices that are considered in quantum transport have a symmetric structure of zero and nonzero entries, and therefore the corresponding graphs are undirected.

An example of a matrix and its graph representation is shown in Fig. 3(a) and (b). Here, the graph is depicted by drawing dots for every vertex v, and lines connecting these dots for every edge (v_1, v_2). It should be noted that a graphical representation of a tight-binding grid, such as shown in Fig. 1(a), can be directly interpreted as a representation of a graph and the corresponding matrix structure.

A *hypergraph* $\mathcal{H}$ is an ordered pair $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, where $\mathcal{V}$ is a set of vertices, and $\mathcal{N}$ a set of *nets* n_i (also called *hyperedges*) between them. A net n_i is a set of vertices, i.e. $n_i \subset \mathcal{V}$. A net n_j is said to be *incident* to a vertex v_i, iff $v_i \in n_j$. An undirected graph is a special realization of a hypergraph, where every net contains exactly two vertices.

The structure of a sparse matrix may also be represented by a hypergraph [37,38]. In the *column-net* model, a matrix H is represented by a hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{N})$, such that there is a vertex v_i for every row and a net n_j for every column of the matrix. The net n_j contains the vertices corresponding to the rows that have a nonzero entry in column j, i.e. $v_i \in n_j$ iff $H_{ij} \neq 0$. Alternatively, the matrix H may also be represented in the *row-net* model, where the role of rows and columns is reverted with respect to the column-net model. In the case of structurally symmetric matrices as considered for quantum transport, the column-net and row-net model are identical. In the following we will therefore only employ the term hypergraph model.

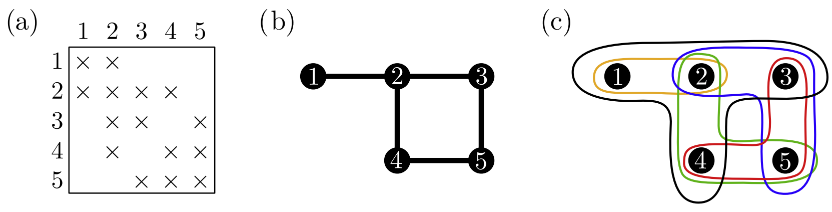

Fig. 3. Simple example of a matrix with a symmetric structure and its graph and hypergraph representations: (a) matrix, (b) graph, and (c) hypergraph (nets are shown in different color).

There is a particularly simply connection between the graph and hypergraph models $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ of a structurally symmetric matrix with nonzero diagonal: in this case, the net n_j contains the vertex v_j and all adjacent vertices,

$$n_j = \{v \in \mathcal{V} | v \text{ is adjacent to } v_j\}, \tag{15}$$

i.e. a net of the hypergraph contains complete information about the adjacent vertices of a given vertex. Fig. 3(a) and (c) shows an example of a matrix and the corresponding hypergraph model.

In principle, the hypergraph models are more general than the graph representation, as they can also describe any sparse matrix, including nonsymmetric rectangular matrices. Since the Hamiltonian matrices encountered in transport calculations are always structurally symmetric and square, using the hypergraph representations may at first glance seem unnecessary, as the simpler graph representation is enough. However, it should be noted that the graph and hypergraph representation carry different information: whereas the edges of the graph model represent *individual* entries, the nets of the hypergraph model represent *entire* columns. This distinction will be of importance in Section 2.2.2, where both the graph and the hypergraph model will be employed simultaneously. Until then, the graph representation will be sufficient.

In terms of graph theory, matrix reordering corresponds to renumbering the vertices of a graph. Since we are only interested in reordering the matrix in terms of matrix blocks (the order within a block should not matter too much), we define a *partitioning* of $\mathcal{G}$ as a set $\{\mathcal{V}_i\}$ of disjoint subsets $\mathcal{V}_i \subset \mathcal{V}$ such that $\bigcup_i \mathcal{V}_i = \mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for $i \neq j$. Using these concepts, we can now reformulate the original matrix reordering problem into a graph partitioning problem:

Problem 2.2. *Partitioning problem*: Find a partitioning $\{\mathcal{V}_0, \ldots, \mathcal{V}_{N+1}\}$ of $\mathcal{G}$ such that:

(i) $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ contain the vertices belonging to left and right leads,
(ii) (a) vertices in $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ are only connected to vertices in $\mathcal{V}_1$ and $\mathcal{V}_N$, respectively,
(b) for $0 < i < N+1$, there are edges between $\mathcal{V}_i$ and $\mathcal{V}_j$ iff $j = i+1, i, i-1$,
(iii) the number $N+2$ of sets $\mathcal{V}_i$ is as large as possible, and all sets $\mathcal{V}_i$ have the same cardinality $|\mathcal{V}_i|$. A partitioning with all $|\mathcal{V}_i|$ equally sized is called *balanced*.

A partitioning obeying requirement 2.2.ii is called a *level set* with *levels* $\mathcal{V}_i$ [39]. Level sets appear commonly as an intermediate step in algorithms for bandwidth reduction of matrices [39–42]. These algorithms seek to find a level set of minimal width, i.e. $\max_{i=0,\ldots,N+1} |\mathcal{V}_i|$ as small as possible which is equivalent to requirement 2.2.iii. The main difference between our graph partitioning problem and the bandwidth reduction problem is requirement 2.2.i: in the partitioning problem, $\mathcal{V}_0$ and $\mathcal{V}_N$ are determined by the problem at hand, while in the bandwidth reduction problem these can be chosen freely. Due to this difference, bandwidth reduction algorithms can be applied successfully to our partitioning problem only for special cases, as we show below.

The term *graph (hypergraph) partitioning* usually refers to the general problem of finding a balanced partitioning $\{\mathcal{V}_i\}$ of a graph (hypergraph) with the objective of minimizing the number of edges (nets) between different parts. Graph and hypergraph partitioning has many applications in various fields such as very-large-scale integration (VLSI) design [43–46], sparse matrix reorderings for LU or Cholesky decompositions [47], or block ordering of sparse matrices for parallel computation [48–53]. In particular, the latter examples also include block-tridiagonal orderings [49,50]. However, as these reorderings are geared towards parallel computation, they obtain a fixed number N of sets $\mathcal{V}_i$ given by the number of processors of a parallel computer, whereas in our block-tridiagonal reordering the number N should be as large as possible. In addition to that, the constraints on the blocks $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ (requirement 2.2.i) are again not present there.

As we cannot directly employ existing techniques to solve the partitioning problem, we will develop an algorithm combining ideas from both bandwidth reduction and graph as well as hypergraph partitioning techniques in the subsequent sections: concepts from bandwidth reduction are employed to construct a level set which is then balanced using concepts from graph theory.

2.2. Matrix reordering by graph partitioning

2.2.1. A local approach—breadth-first-search

A breadth-first-search (BFS) [54] on a graph immediately yields a level set [39–42]. In our particular example, the level set is constructed as follows:

Algorithm 1. Level set construction by breadth-first-search.

```
A. Start from $i = 0$. Then, $\mathcal{V}_i = \mathcal{V}_0$, as the first level is given by the constraints of requirement 2.2.i.
B. If there is a vertex in $\mathcal{V}_i$ that is adjacent to a vertex in $\mathcal{V}_{N+1}$, assign all the remaining unassigned vertices into $\mathcal{V}_{N+1}$ and
   end the algorithm.
C. All vertices adjacent to $\mathcal{V}_i$ not contained in the previous levels $\mathcal{V}_i, \mathcal{V}_{i-1}, \ldots, \mathcal{V}_0$ are assigned to $\mathcal{V}_{i+1}$.
D. Continue at step B with $i = i + 1$.
```

Note that the sets $\{\mathcal{V}_i\}$ form a level set by construction—a set $\mathcal{V}_i$ may only have vertices adjacent to $\mathcal{V}_{i-1}$ and $\mathcal{V}_{i+1}$. The construction by BFS not only obtains the number of levels $N+2$ for a particular realization, but yields a more general information:

Lemma 2.3. *The number of levels $N+2$ in the level set constructed by Algorithm 1 is the maximum number of levels compatible with the constraints on the initial and final level $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ for a graph $\mathcal{G}$.*

This can be seen from the fact that a BFS finds the shortest path in the graph between the initial sets $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$, $(v_0, v_1, \ldots, v_i, \ldots, v_{N+1})$, where $v_0 \in \mathcal{V}_0$ and $v_{N+1} \in \mathcal{V}_{N+1}$. Any vertex on this shortest path can be uniquely assigned to a single level $\mathcal{V}_i$ and it would not be compatible with a larger number of levels than $N+2$.

Algorithm 1 not only yields the maximum number of levels: all vertices contained in the first n levels of the BFS must be contained in the first n levels of *any* other level set:

Lemma 2.4. *Let $\{\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{N+1}\}$ be a level set constructed by Algorithm 1, and $\{\mathcal{V}'_0, \mathcal{V}'_1, \ldots, \mathcal{V}'_{N'+1}\}$ another level set consistent with the requirements of Problem 2.2 with $N' \leqslant N$. Then $\mathcal{V}_0 \cup \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_n \subset \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup \cdots \cup \mathcal{V}'_n$ for $0 \leqslant n \leqslant N'+1$.*

The statement is proved by induction. It is true trivially for $n = 0$ (because of requirement i in Problem 2.2) and for $n = N'+1$ (then the levels cover the whole graph). Suppose now that the statement holds for $n < N'$. Note that for the proof it suffices to show that $\mathcal{V}_{n+1} \subset \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup \cdots \cup \mathcal{V}'_{n+1}$. Consider now the set of all vertices adjacent to $\mathcal{V}_n$, adjacent$(\mathcal{V}_n) = \{v \in \mathcal{V} | v$ is adjacent to some $v' \in \mathcal{V}_n\}$. By construction, $\mathcal{V}_{n+1} \subset$ adjacent$(\mathcal{V}_n)$. Since $\mathcal{V}_n \subset \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup \cdots \cup \mathcal{V}'_n$ and $\{\mathcal{V}'_i\}$ is a level set, all vertices adjacent to $\mathcal{V}_n$ must be contained in the set of vertices including the next level, i.e. adjacent$(\mathcal{V}_n) \subset \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup \cdots \cup \mathcal{V}'_n \cup \mathcal{V}'_{n+1}$. But then also $\mathcal{V}_{n+1} \subset \mathcal{V}'_0 \cup \mathcal{V}'_1 \cup \cdots \cup \mathcal{V}'_{n+1}$, which concludes the proof.

Thus, the vertices contained in the first n levels of the BFS form a minimal set of vertices needed to construct n levels. However, this also implies that the last level, which then covers the remaining vertices of the graph, may contain many more vertices than the average, leading to an unbalanced level set. This is not surprising, since the algorithm does not explicitly consider balancing and only local information is used, i.e. whether a vertex is adjacent to a level or not. An example for this imbalance is shown in Fig. 4, where the BFS construction yields a very large last level.

Note that throughout the manuscript we visualize the graph theoretical concepts using examples of graphs obtained from discretizing a two-dimensional Hamiltonian using the method of finite differences. However, the ideas and algorithms presented here apply to any graph and are not limited to graphs with coordinate information. Two-dimensional graphs have the advantage of being visualized easily. In particular, the BFS search has an intuitive physical analog: wave front propagation of elementary waves emanating from the vertices of the initial level $\mathcal{V}_0$.

The problem that a BFS does not yield a balanced partitioning was also noted in the theory of bandwidth reduction. The Gibbs–Poole–Stockmeyer (GPS) algorithm tries to overcome this deficiency by constructing a level set through the combination of two BFS searches starting from the initial and the final levels. However, there the initial and final levels are sought to be furthest apart, contrary to our problem. In general, the GPS construction only yields a balanced level set if the initial and final level are close to furthest apart, as we will show in Section 3.

2.2.2. A global approach—recursive bisection

In order to obtain a balanced partitioning, graph partitioning algorithms commonly perform a recursive bisection, i.e. successively bisect the graph and the resulting parts until the desired number of parts is obtained [43,45,49,50,55,56]. This approach has the advantage of reducing the partitioning problem to a simpler one, namely bisection. Furthermore, if the resulting parts of every bisection are equally sized, the overall partitioning will be balanced. In addition, bisection is inherently a global approach, as the whole graph must be considered for splitting the system into two equally sized parts. Thus, it can be expected to yield better results than a local approach, such as BFS.

We intend to construct a level set with $N+2$ levels, where $N+2$ is the maximum number of levels as determined by Algorithm 1. To this end we start from an initial partitioning $\{\mathcal{V}_0, \mathcal{V}_1, \mathcal{V}_{N+1}\}$, where $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ contain the vertices of the leads (requirement 2.2.i), and $\mathcal{V}_1$ all other vertices. The level set is then obtained by applying the bisection algorithm recursively to $\mathcal{V}_1$ and the resulting subsets, until N levels are obtained, as shown schematically in Fig. 5. Here bisection means splitting a set $\mathcal{V}_i$ into two sets, $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$, such that $\mathcal{V}_{i_1} \cup \mathcal{V}_{i_2} = \mathcal{V}_i$ and $\mathcal{V}_{i_1} \cap \mathcal{V}_{i_2} = \emptyset$. In oder to be applicable to the partitioning Problem 2.2, the bisection must comply with certain requirements:

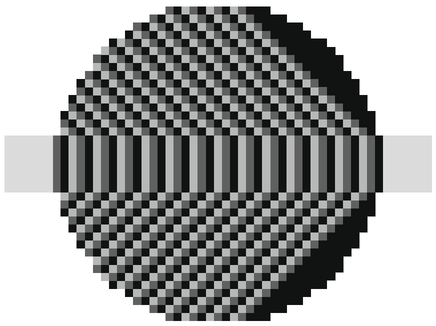

Fig. 4. Level set created by a BFS starting from $\mathcal{V}_0$. Different levels are shown in alternating shades of grey.

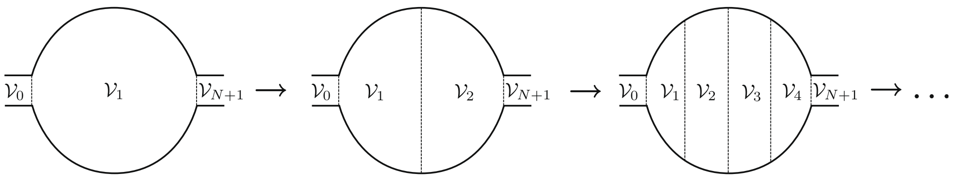

Fig. 5. Schematic depiction of recursive bisection.

Problem 2.5. The bisection algorithm must be

(i) Compatible with a level set with $N + 2$ levels.
(ii) Balanced.
(iii) Performed such that subsequent bisections may lead to a balanced level set.

Requirement 2.5.iii is formulated rather vaguely: usually there are many different choices how to perform a bisection. A particular choice will influence the subsequent bisections (for a similar problem in graph partitioning see [56]), and thus the bisection algorithm must in principle take into account all following bisection steps. Since an exact solution to that problem seems computationally intractable, we will resort to heuristics there.

We start explaining how to comply with requirements 2.5.i and 2.5.ii. In the following we assume that $N > 0$, as $N = -1, 0$ are trivial cases. Then the initial partitioning $\{\mathcal{V}_0, \mathcal{V}_1, \mathcal{V}_{N+1}\}$ forms a level set, and so will the final result of the recursive bisection, if the result of every intermediate bisection yields a level set. For this, consider a set $\mathcal{V}_i$ with vertices adjacent to the sets $\mathcal{V}_{i_\text{left}}$ and $\mathcal{V}_{i_\text{right}}$, where "left" ("right") is defined as being closer to $\mathcal{V}_0(\mathcal{V}_{N+1})$. Then the sets resulting from the bisection, $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ may only have vertices adjacent to $\mathcal{V}_{i_\text{left}}$,$\mathcal{V}_{i_2}$ and $\mathcal{V}_{i_1}$,$\mathcal{V}_{i_\text{right}}$, respectively.

Apart from the condition of forming a level set, requirement 2.5.i also dictates the total number of levels. Due to the nature of the recursive bisection, the number of final levels contained in an intermediate step is always well-defined. If a set $\mathcal{V}_i$ contains N_i levels, then $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ must contain $N_{i_1} = \text{Int}(N_i/2)$ and $N_{i_2} = N_i - \text{Int}(N_i/2)$ levels, respectively. Here, $\text{Int}(\ldots)$ denotes rounding off to the next smallest integer. The bisection is thus balanced, if

$$|\mathcal{V}_{i_1}| \approx \frac{N_{i_1}}{N_i}|\mathcal{V}_i| \quad \text{and} \quad |\mathcal{V}_{i_2}| \approx \frac{N_{i_2}}{N_i}|\mathcal{V}_i|. \tag{16}$$

Note that N_i can take any value, and usually is not a power of two.

From Lemma 2.4 we know that the minimum set of vertices necessary to form n levels is given by a BFS up to level n. Let $\mathcal{V}_{i_1,\text{BFS}}(\mathcal{V}_{i_2,\text{BFS}})$ denote the set of vertices found by a BFS starting from $\mathcal{V}_{i_\text{left}}(\mathcal{V}_{i_\text{right}})$ up to level $N_{i_1}(N_{i_2})$. Then, for any bisection complying with requirement 2.5.i, $\mathcal{V}_{i_1,\text{BFS}} \subset \mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2,\text{BFS}} \subset \mathcal{V}_{i_2}$. These vertices are uniquely assigned to $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ and are consequently marked as *locked*, i.e. later operations may not change this assignment. An example for the vertices found in a BFS is shown in Fig. 6(a). Note that in the initial bisection, $\mathcal{V}_i = \mathcal{V}_1, N_i = N, \mathcal{V}_{i_\text{left}} = \mathcal{V}_0$, and $\mathcal{V}_{i_\text{left}} = \mathcal{V}_{N+1}$.

The remaining unassigned vertices can be assigned to either set, and the bisection will still be compatible with a level set containing $N + 2$ vertices. Thus for complying with requirement 2.5.ii, any prescription obeying the balancing criterion may be used. We choose to distribute the remaining vertices by continuing the BFS from $\mathcal{V}_{i_\text{left}}$ and $\mathcal{V}_{i_\text{right}}$ and assigning vertices to $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ depending on their distance to the left or right neighboring set, while additionally obeying the balancing criterion. This approach—assigning vertices to levels according to their distance from the initial and final set—is rather intuitive and probably the procedure that would be used if the level set were to be constructed "by hand". This procedure may lead to reasonable level sets, however, in general, additional optimization on the sets $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ is needed, as discussed below. If this optimization is used, it can also be useful to distribute the unassigned vertices randomly, as this may help avoiding local minima.

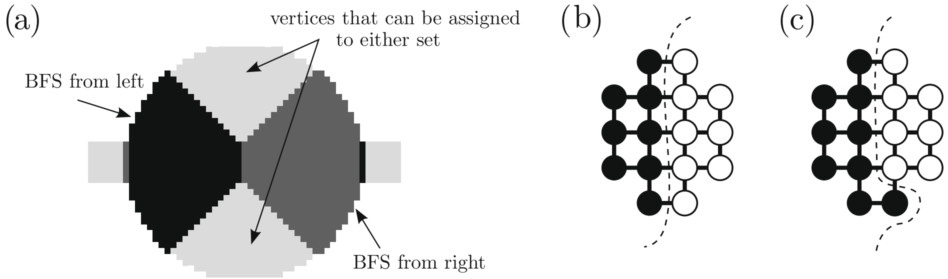

Fig. 6. (a) Example showing for a disk-type geometry the BFS from the left and right neighboring sets that construct the minimal set of vertices $\mathcal{V}_{i_1,\text{BFS}}$ (black) and $\mathcal{V}_{i_2,\text{BFS}}$ (dark grey) that must be contained in $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$, respectively. The remaining vertices (light grey) can be assigned to either set. (b) and (c) Examples illustrating the difference between cut edges and cut nets. The number of cut edges is 5 in both (b) and (c), while the number of cut nets (surface vertices) is 10 in (b) and 9 in (c).

As mentioned above, there is a lot of arbitrariness in distributing the unassigned vertices into $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$. However, the particular choice of the bisection will influence whether a later bisection is balanced or not: if $\mathcal{V}_{i_1(i_2),\text{ BFS}}$ contains more vertices than given by the balance criterion (16), the bisection *cannot* be balanced. Obviously, the BFS that constructs $\mathcal{V}_{i_1(i_2),\text{BFS}}$ depends on the details of the set $\mathcal{V}_i$ and thus on the details of the previous bisection step.

In order to formulate a criterion that may resolve the above mentioned arbitrariness and help to find a balanced level set, it is useful to consider the matrix representation of the graph $\mathcal{G}$. Bisecting a graph means ordering the corresponding matrix into two blocks that are connected by an off-diagonal matrix H_{i_1,i_2}:

$$\left(\begin{array}{c|c} \ddots & \\ & H_{i_1,i_2} \\ \hline H_{i_2,i_1} & \\ & \ddots \end{array}\right). \tag{17}$$

This off-diagonal matrix will be unchanged by further bisections and thus determines the minimum level width that can be achieved. Therefore, the size of the off-diagonal matrices H_{i_1,i_2} and H_{i_2,i_1} should be minimized.

In a bisection, an edge $(v_1, v_2) \in \mathcal{E}$ is said to be *cut*, if v_1 and v_2 belong to different sets, i.e. $v_1 \in \mathcal{V}_{i_1}$ and $v_2 \in \mathcal{V}_{i_2}$ or vice versa. The entries of H_{i_1,i_2} correspond to edges cut by the bisection, and minimizing the number of entries in H_{i_1,i_2} corresponds to minimizing the number of edges cut by the bisection (*min-cut criterion*). The problem of finding a bisection minimizing the number of cut edges is generally referred to as graph partitioning, and commonly used to reorder matrices for parallel computing.

However, the number of entries in H_{i_1,i_2} is not directly related to the size of the matrix: instead of minimizing the number of *entries*, it is better to minimize the number of *columns and rows* of H_{i_1,i_2} which are conveniently represented by the hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ corresponding to the matrix H.

A net n_j of a hypergraph is said to be cut by a bisection, if any two vertices $v_1, v_2 \in n_j$ are contained in different sets $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$. The total number of nonzero columns (and rows, since $H_{i_2,i_1} = H^{\dagger}_{i_1,i_2}$) in the matrices H_{i_1,i_2} and H_{i_2,i_1} is then given by the number of cut nets. Thus, minimizing the number of cut nets (*min-net-cut criterion*) corresponds to minimizing the total number of nonzero columns (and rows) in H_{i_1,i_2} and H_{i_2,i_1}. The problem of finding a bisection minimizing the number of cut nets is generally referred to as hypergraph partitioning. The superiority of hypergraph partitioning over graph partitioning in the context of matrix reordering has first been noted in the context of block-tridiagonalization [49,50] and in the graph partitioning problem for parallel computing [37,51].

In terms of the graph structure, a cut net corresponds to a surface vertex, i.e. a vertex with at least one edge cut by the bisection. Since the vertices in $\mathcal{V}_{i_{1/2},\text{BFS}}$ are determined by a BFS emanating from the surface vertices, minimizing the number of cut nets (and hence the number of surface vertices) will usually also lead to a smaller number of vertices in $\mathcal{V}_{i_{1/2},\text{BFS}}$, leaving more freedom towards achieving a balanced bisection. Fig. 6(b) and (c) shows a comparison of the min-cut and min-net-cut criterion for simple examples. In practice, when minimizing the number of cut nets, we also use the min-cut criterion to break ties between different bisections with the same number of cut nets (*min-net-cut-min-cut criterion*). Hence, in addition to minimizing the total number of columns in H_{i_1,i_2} and H_{i_2,i_1}, we also minimize the number of entries. This helps to avoid wide local minima, that occur frequently in the min-net-cut optimization problem.

Note that minimizing the total number of columns in H_{i_1,i_2} and H_{i_2,i_1} only minimizes the size of the matrix block containing both H_{i_1,i_2} and H_{i_2,i_1}. In order to minimize the individual sizes of H_{i_1,i_2} and H_{i_2,i_1} it is desirable to also equilibrate the individual number of columns (and rows) in H_{i_1,i_2} and H_{i_2,i_1}, and hence balance the number of surface vertices in $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$. In practice, we have, however, found that the typical regular structure of matrices encountered in transport always automatically lead to an approximately equal number of surface vertices in $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$, such that an explicit balancing was not necessary.

Both the min-cut and min-net-cut bisection problem have been shown to be $\mathcal{NP}$-hard [57]. Therefore, only heuristics are available to solve them. These heuristics start from an initial (balanced) bisection, such as constructed by the steps outlined above, and improve upon this initial bisection. Here, we choose to use the Fiduccia–Mattheyses (FM) algorithm [45], as it is suitable for both graph and hypergraph bisection. Furthermore, the FM algorithm can naturally deal with locked vertices that may not be moved between sets, is reasonable fast and its underlying concepts are easy to understand. The FM heuristic is a *pass-based* technique, i.e. it is applied repeatedly to the problem (several *passes* are performed), iteratively improving the bisection. More detailed information about the fundamentals of the Fiduccia–Mattheyses algorithm and the implementation are given in Appendix A.

We now summarize the steps outlined above and formulate an algorithm for bisection:

Algorithm 2. Bisection of set $\mathcal{V}_i$ containing N_i levels, with left (right) neighboring set $\mathcal{V}_{i_\text{left}}(\mathcal{V}_{i_\text{right}})$.

A. Stop, if $N_i = 1$.
B. Do a BFS starting from $\mathcal{V}_{i_\text{left}}$ up to level $N_{i_1} = \text{Int}(N_i/2)$ and a BFS starting from $\mathcal{V}_{i_\text{right}}$ up to level $N_{i_2} = N - \text{Int}(N_i/2)$. The vertices found by the BFS are assigned to $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$, respectively, and are marked as locked.

C. Distribute the remaining unassigned vertices taking into account the balance criterion (16). The vertices may be assigned according to either one of the following prescriptions:
   (a) Continue the BFSs from step B and assign vertices to $\mathcal{V}_{i_1}$, if they are first reached by the BFS from $\mathcal{V}_{i_{\text{left}}}$, and to $\mathcal{V}_{i_2}$, if they are first reached by the BFS from $\mathcal{V}_{i_{\text{right}}}$. If a set has reached the size given by the balance criterion, assign all remaining vertices to the other set.
   (b) Distribute the unassigned vertices randomly to $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$. If a set has reached the size given by the balance criterion, assign all remaining vertices to the other set.

D. Optimize the sets $\mathcal{V}_{i_1}$ and $\mathcal{V}_{i_2}$ by changing the assignment of unlocked vertices according to some minimization criterion. In particular, the following optimizations may be performed:
   (a) No optimization.
   (b) Min-cut optimization using the FM algorithm.
   (c) Min-net-cut optimization using the FM algorithm.
   (d) Min-net-cut-min-cut optimization using the FM algorithm.

Recursive application of the bisection Algorithm 2 then leads to an algorithm for constructing a level set complying with the requirements of the partitioning Problem 2.2, and thus an algorithm for block-tridiagonalizing a matrix.

Algorithm 3. Block-tridiagonalization of matrix H

A. Construct the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to the matrix H, and the sets $\mathcal{V}_0$ and $\mathcal{V}_{N+1}$ corresponding to the leads.
B. Use Algorithm 1 to determine the maximum number of levels $N + 2$. If $N < 1$, stop.
C. Construct $\mathcal{V}_1 = \mathcal{V} \setminus (\mathcal{V}_0 \cup \mathcal{V}_{N+1})$, containing N levels.
D. Apply the bisection Algorithm 2 to $\mathcal{V}_1$ and then recursively on the resulting subsets. Do not further apply if a set only contains one level.

It should be emphasized, that the block-tridiagonalization does not require any other input than the graph structure. In principle, the number of FM passes may affect the result. However, from experience, this number can be chosen as a fixed value, e.g. 10 FM passes, for all situations [45]. Thus, the block-tridiagonalization algorithm can serve as a black box.

In Fig. 7 we show examples of level sets arising from the natural ordering of grid points (Fig. 7(a), *natural level set*) and from the block-tridiagonalization algorithm developed in this work (Fig. 7(b)–(d)) for the case of a disk-type geometry. The level set in Fig. 7(b) arises from recursive bisection, where the vertices were distributed according to a BFS without any optimization. The resulting level set strongly resembles the natural level set. This is due to the highly symmetric structure and the fact that vertices are assigned to levels according to their distance from the leads—only small deviations are present due

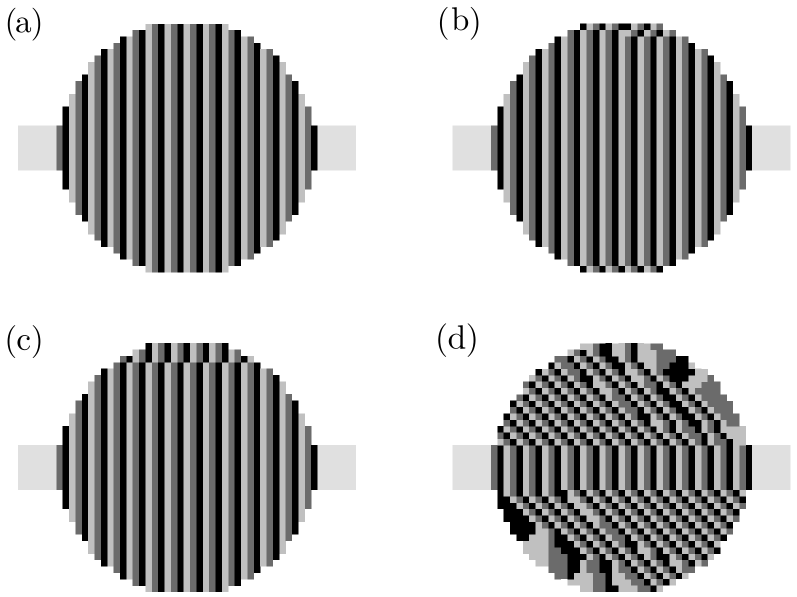

Fig. 7. Examples of level sets arising from (a) the natural ordering of grid points (as in Fig. 1), and application of the block-tridiagonalization Algorithm 3 with distribution of vertices by BFS (Algorithm 2, step C(a)), (b) without further optimization, (c) with min-cut optimization, and (d) with min-net-cut optimization.

to the balance criterion. When the bisection is optimized according to the min-cut criterion, Fig. 7(c), the resulting level set changes little, as the min-cut criterion favors horizontal and vertical cuts for a square lattice, as presented in the example. In contrast, min-net-cut optimization (Fig. 7(d)) yields a new, non-trivial level set that has less symmetry than the underlying structure. Note that the minimization of surface vertices leads to levels in the form of "droplets", analogous to surface tension in liquids.

In fact, we will show in Section 3 that min-net-cut optimization usually leads to level sets and thus block-tridiagonal orderings that are superior to those arising from other methods. In particular, they are better than the natural level sets, leading to a significant speed-up of transport algorithms, as demonstrated in Section 3.1. In addition to that, the reordering algorithms allow one to use conventional two-terminal transport algorithms also for more complicated, even multi-terminal structures (see Sections 3.1 and 3.2).

2.2.3. Computational complexity

We conclude the theoretical considerations with an analysis of the computational complexity of Algorithms 2 and 3.

The bisection algorithm involves a BFS search on $\mathcal{V}_i$, which scales linearly with the number of edges within $\mathcal{V}_i$, and thus has complexity $\mathcal{O}(|\mathcal{E}_i|)$, where $\mathcal{E}_i$ is the set of edges within $\mathcal{V}_i$. In addition to that, a single optimization pass of the FM algorithm scales also as $\mathcal{O}(|\mathcal{E}_i|)$ [45] (for details on the implementation of the FM algorithm, see Appendix A). Usually, a constant number of passes independent of the size of the graph is enough to obtain converged results, and therefore the optimization process of several FM passes is also considered to scale as $\mathcal{O}(|\mathcal{E}_i|)$. Thus, the full bisection algorithm also has complexity $\mathcal{O}(|\mathcal{E}_i|)$.

Usually, the number of edges per vertex is approximately homogeneous throughout the graph. Since the recursive bisection is a divide-and-conquer approach, the computational complexity of the full block-tridiagonalization algorithm is then $\mathcal{O}(|\mathcal{E}|\log|\mathcal{E}|)$ [54]. In typical graphs arising from physics problems, the number of edges per vertex is a constant, and the computational complexity can be written as $\mathcal{O}(N_{\text{grid}}\log N_{\text{grid}})$, where N_{grid} is the number of vertices in $\mathcal{V}$, or the size of the matrix H.

In contrast, many quantum transport algorithms, such as the recursive Green's function technique, scale as $\mathcal{O}(N(N_{\text{grid}}/N)^3) = \mathcal{O}(N_{\text{grid}}^3/N^2)$ in the optimal case of N equally sized matrix blocks (levels) of size N_{grid}/N. Often, the number of blocks (levels) $N \propto N_{\text{grid}}^{\alpha}$. Typically, to name a few examples, $\alpha = 1$ in one-dimensional chains, $\alpha = 1/2$ in two dimensions, and the transport calculation scales as $\mathcal{O}(N_{\text{grid}}^{3-2\alpha})$. Thus, except for the case of a linear chain, where $N = N_{\text{grid}}$ and matrix reordering is pointless anyways, the block-tridiagonalization algorithm always scales more favorably than the quantum transport algorithms. This scaling implies that the overhead of the matrix reordering in the transport calculation will become more negligible, the larger the system size.

3. Examples: charge transport in two-dimensional systems

3.1. Ballistic transport in two-terminal devices

We now evaluate the performance of the block-tridiagonalization algorithm using representative examples from mesoscopic physics. The Schrödinger equation for the two-dimensional electron gas (2DEG) is usually transformed into a tight-binding problem by the method of finite differences [7–9], where the continuous differential equation is replaced by a set of linear equations involving only the values of the wave function on discrete grid points. Commonly, these points are arranged in a regular, square grid. This grid, together with the shape of the particular structure under consideration then defines the structure of the Hamilton matrix and the corresponding graph.

The representative examples considered here are shown in Fig. 8: the circle (Fig. 8(a)) and the asymmetric Sinai billiard (Fig. 8(b)) that are examples of integrable and chaotic billiards in quantum chaos, respectively, the ring (Fig. 8(c)) that may exhibit various interference physics, and the circular cavity with leads that are not parallel (Fig. 8(d)) as an example of a structure that does not have an intuitive, natural block-tridiagonal ordering. For all these structures, we introduce a length scale d_{extent}, given by the outer radius of the circular structures and the side length of the square structure, characterizing the maximum extent. The fineness of the grid, and thus the size of the corresponding graph will be measured in number of grid points per length d_{extent}.

We now apply the block-tridiagonalization algorithm using the various optimization criteria discussed in the previous section, and compare the resulting orderings with the natural ordering and the ordering generated by the GPS algorithm. The weights $w(H)$, Eq. (14), of the different orderings are given in Table 1.

The initial distributions for the bisection algorithm are done in two different ways: the vertices are distributed both in an ordered way—by BFS—and randomly. The outcome after the optimization, however, is always similar for both types of initial distributions which indicates that the resulting weights are close to the global minimum and not stuck in a local minimum. Note that we use twice as many FM passes for a random initial distribution than for an initial distribution by BFS, as convergence is usually slower for a random initial distribution.

In all examples, the min-net-cut criterion yields orderings with the best weights, as expected from the considerations of the previous section. Based on the weight, orderings according to this criterion are expected to give the best performance in transport calculations such as the RGF algorithm. Note that the min-net-cut-min-cut ordering is on average closest to the

[redacted]

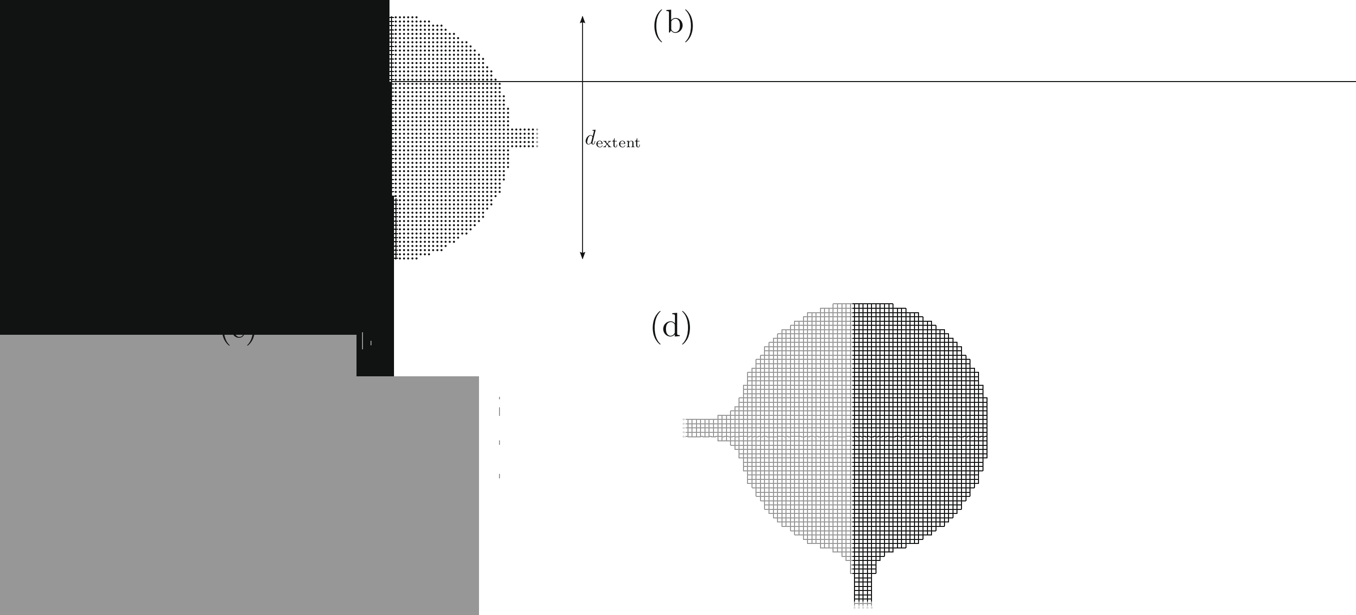

Fig. 8. Typical examples of structures considered in two-dimensional mesoscopic systems: (a) circle billiard, (b) asymmetric Sinai billiard, (c) ring, and (d) circular cavity with perpendicular leads. The tight-binding grid arises from the finite difference approximation to the Schrödinger equation. Note that the number of grid points used here was deliberately chosen ve[redacted] least two orders of magnitude larger. d_{extent} denotes a length characterizing the extent for the different structures.

Table 1
Weights $w(H)$, Eq. (14), for the block-tridiagonal ordering constructed by different algorithms for the examples of Fig. 8. Optimization was done by 10 passes of the FM algorithm, when the initial bisection was constructed by BFS (Algorithm 2, step C(a)), and 20 passes, when the initial bisection was constructed by a random distribution of vertices (Algorithm 2, step C(b)). The minimal weights for each system are printed bold. In all examples, there were 400 grid points per length d_{extent}.

	Circular billiard	Asymmetric Sinai billiard	Ring	Cavity with perp. leads
Natural block-tridiagonal ordering	1.51×10^{10}	1.58×10^{10}	8.72×10^{8}	–
Gibbs–Poole–Stockmeyer	1.15×10^{12}	7.84×10^{11}	2.14×10^{8}	7.05×10^{12}
Distribution by BFS, no optimization	1.51×10^{10}	9.29×10^{9}	2.1×10^{8}	1.69×10^{10}
Distribution by BFS, min-cut	1.51×10^{10}	9.67×10^{9}	2.1×10^{8}	1.59×10^{10}
Random distribution, min-cut	2.22×10^{10}	9.95×10^{9}	2.1×10^{8}	5.13×10^{10}
Distribution by BFS, min-net-cut	1.51×10^{10}	9.46×10^{9}	2.1×10^{8}	$\mathbf{1.18 \times 10^{10}}$
Random distribution, min-net-cut	1.46×10^{10}	$\mathbf{9.0 \times 10^{9}}$	2.09×10^{8}	$\mathbf{1.18 \times 10^{10}}$
Distribution by BFS, min-net-cut-min-cut	$\mathbf{1.26 \times 10^{10}}$	9.28×10^{9}	$\mathbf{2.08 \times 10^{8}}$	1.24×10^{10}
Random distribution, min-net-cut-min-cut	1.27×10^{10}	9.16×10^{9}	2.09×10^{8}	2.02×10^{10}

[redacted] mes suffers from slow convergence, when the algorithm must traverse a wide
[redacted] on helps to break ties and thus avoids these wide local minima.
[redacted] rform well, the GPS algorithm yields weights that are even larger than the
[redacted]

[redacted] gorithm 3 in the combination of initial distribution by BFS and min-net-cut-
[redacted] s with respect to the weight $w(H)$. Experience shows that usually 10 FM passes
[redacted] sequence, we will use this combination exclusively in the rest of this work.
[redacted] asure of the quality of a ordering. Additional insight can be gained from the
[redacted] s/levels. In Fig. 9 we show this distribution before and after reordering. For
[redacted] s, the number of matrix blocks is determined by the number of lattice points
[redacted]). In contrast, the number of matrix blocks after reordering is given by the
[redacted] ads, in terms of the corresponding graph.

[redacted]

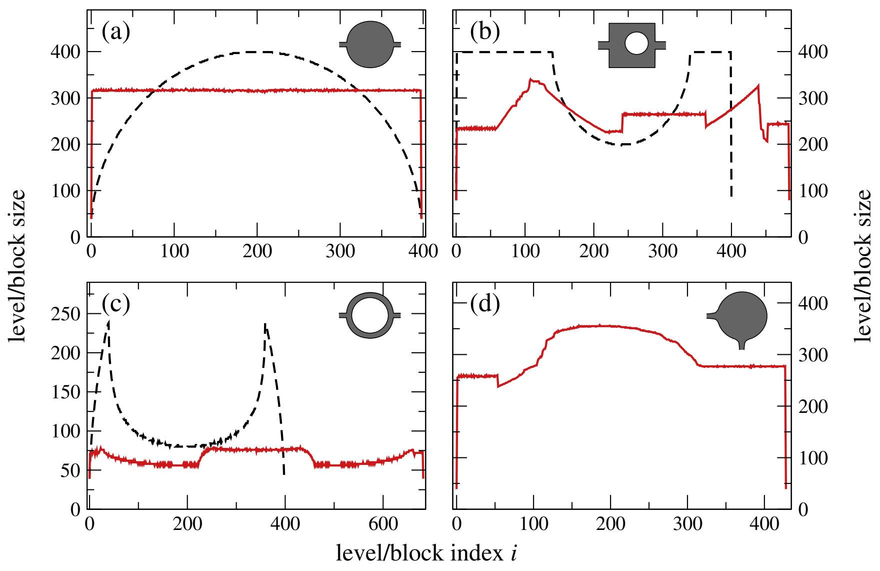

Fig. 9. Level (matrix block) size M_i as a function of the level (matrix block) index i for the natural level set (dashed line) and the min-net-cut-min-cut reordering (solid line), shown for (a) the circle billiard, (b) the asymmetric Sinai billiard, (c) the ring, and (d) the circular cavity with perpendicular leads. Note that for (d), there is no natural ordering. In all examples, there were 400 grid points per length d_{extent}.

In the case of the circle billiard, Fig. 9(a), the number of matrix blocks is the same for the natural ordering and the reordered matrix, as the shortest path between the leads is simply a straight line along the x-coordinate direction. The improvements in the weight originate only from balancing the matrix block sizes: While the matrix block sizes vary for the natural ordering—the lateral size changes along the x-direction—the reordered matrix has equally sized matrix blocks. For this particular example, the result of the block-tridiagonalization algorithm is optimal, as it yields the best solution with respect to the requirements set forth in Problems 2.1 and 2.2. Note that it is not always possible to find a perfectly balanced partitioning, but the circle billiard is such an example.

In contrast, in the case of the asymmetric Sinai billiard and the ring the number of matrix blocks generated by the block-tridiagonalization algorithm is larger than in the natural ordering (see Fig. 9(b) and (c), respectively). In both cases, the obstacle within the scattering region increases the length of the shortest path connecting the two leads. In both examples, this increase in the number of matrix blocks leads to a significantly decreased weight $w(H)$ with respect to the natural ordering, although the partitioning is only approximately balanced. For instance, in the particular case of the ring, the number of matrix blocks after reordering is approximately given by the number of lattice points around half of the circumference. The reordered ring thus has a weight very similar to a straight wire with a width twice as large as the width of one arm of the ring, and a length given by half of the ring circumference.

For the cavity with perpendicular leads, there is no natural ordering, and a specialized transport algorithm would be required. The reordering creates a matrix with approximately balanced block sizes, and allows the direct application of conventional algorithms.

The weight $w(H)$ was introduced as a theoretical concept in order to simulate the computational complexity of a transport calculation. After discussing the influence of the reordering on this theoretical concept, we now demonstrate how the reordering increases the performance of an actual quantum transport calculation.

To this end we use a straight-forward implementation of the well-established recursive Green's function algorithm for two terminals, as described in Ref. [29]. The necessary linear algebra operations are performed using the ATLAS implementation of LAPACK and BLAS [58,59], optimized for specific processors. It should be emphasized that the code that does the actual transport calculation—such as calculation of the Green's function and evaluation of the Fisher–Lee relation—is the same for all examples considered here, including the non-trivial cavity with perpendicular leads. The abstraction necessary for the reordering, i.e. the graph structure and the corresponding level set, allows for a generic computational code applicable to any tight-binding model.

We measure the performance gain through matrix reordering as

$$r_{\text{cpu-time}} = \frac{\text{computing time for natural ordering}}{\text{computing time for reordered matrix}}. \tag{18}$$

Note that during a real calculation, the conductance is usually not only calculated once, but repeatedly as a function of some parameters, such as Fermi energy or magnetic field. Thus, the respective quantum transport algorithm is executed repeatedly, too. In contrast, the block-tridiagonalization has to be carried out again *only* when the structure of the matrix and thus the corresponding graph changes. For the examples considered here this would correspond to changing the grid spacing or the shape of the structure. In such a case, the overhead of matrix reordering must be taken into account for $r_{\text{cpu-time}}$. This overhead can be quantified as

$$r_{\text{matrix reordering}} = \frac{\text{overhead of matrix reordering}}{\text{computing time including reordering}}. \tag{19}$$

In a typical calculation, however, the matrix structure given by the underlying tight-binding grid does not change, and the matrix reordering must be carried out only once. In this common situation, the overhead of matrix reordering is negligible. For example, any change of physical parameters such as Fermi energy, magnetic field or disorder averages does not change the matrix structure.

In Fig. 10 we show the performance gain through matrix reordering, $r_{\text{cpu-time}}$, as a function of grid size for the circle billiard, the asymmetric Sinai billiard, and the ring (Fig. 10(a)–(c), respectively). We include both measurements excluding and including the overhead of matrix reordering, as discussed above. Remember that in the case of the cavity with perpendicular leads, Fig. 8(d), there is no natural ordering and thus a performance comparison is not possible. In fact for this system, only matrix reordering makes a transport calculation possible in the first place.

We find that block-tridiagonalization always increases the algorithmic performance in the typical situation, when the overhead of matrix reordering can be neglected. However, even if the reordering overhead is taken into account, we see a significant performance gain except for small systems—but there the total computing time is very short anyway. In fact, as the system sizes increases, the overhead of reordering becomes negligible, as predicted from the analysis of the computational complexity, and the performance gains including and excluding the reordering overhead converge. This can also be seen in Fig. 10(d), where we show the reordering overhead $r_{\text{matrix reordering}}$ as a function of system size.

Especially for large systems, the total computing time can become very long, and any performance gain is beneficial. Reordering leads to significant performance gains up to a factor of 3 in the case of the ring. The performance gain $r_{\text{cpu-time}}$ can also be estimated from the weights $w(H)$, Eq. (14), of the original matrix (the natural ordering) and the reordered matrix, shown as the dashed line in Fig. 10(a)–(c). The actual, measured performance gain approaches this theoretical value, as the system size increases, demonstrating that $w(H)$ models the actual performance of a quantum transport algorithm appropriately. Note that we do not fully reach the theoretically predicted performance gain in the case of the ring. On modern computer architectures, computing time does not only depend on the number of arithmetic operations [59], and thus the weight $w(H)$ overestimates the performance gain, though the performance still improves significantly.

Finally, we demonstrate the $\mathcal{O}(N_{\text{grid}} \log N_{\text{grid}})$ scaling of the reordering algorithm. Fig. 11 shows the computing times of the block-tridiagonalization algorithm as a function of matrix/graph size N for the geometries considered in this section. For all systems, the computing times scale according to the prediction from the complexity analysis in Section 2.2.3, as apparent from the fit $\propto N_{\text{grid}} \log N_{\text{grid}}$. Note that for large N_{grid}, $\mathcal{O}(N_{\text{grid}} \log N_{\text{grid}})$ scaling is practically indistinguishable from $\mathcal{O}(N_{\text{grid}})$-scaling, as can also be seen in Fig. 11.

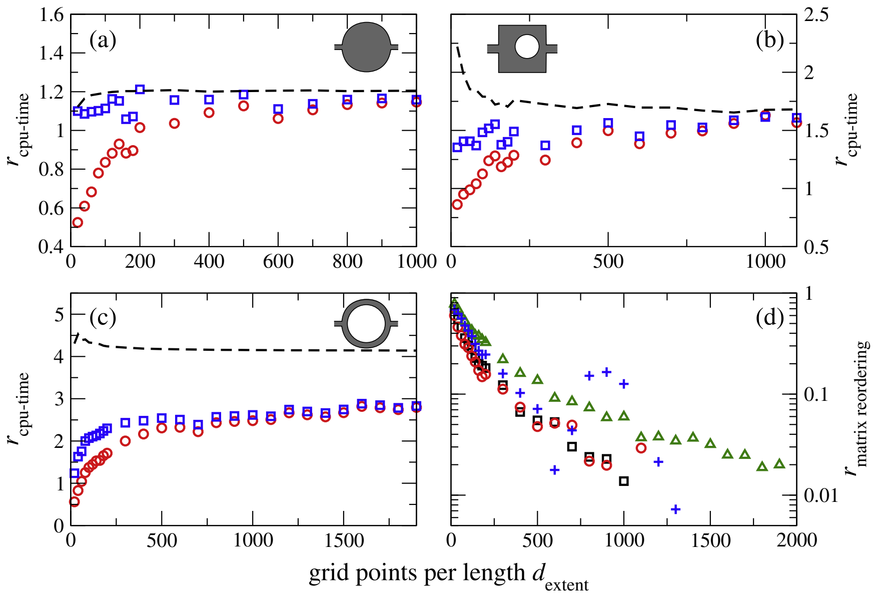

Fig. 10. (a)–(c) Relative gain in computational time $r_{\text{cpu-time}}$, Eq. (18), through the reordering as a function of the grid size for the circular billiard, the asymmetric Sinai billiard, and the ring, respectively. $r_{\text{cpu-time}}$ is shown excluding (□) and including (◯) the overhead of matrix reordering. The estimate for $r_{\text{cpu-time}}$ from the weights $w(H)$ of the different orderings is shown as a dashed line. (d) Fraction of time $r_{\text{matrix reordering}}$, Eq. (19), used for reordering the matrix as a function of the grid size. Data is shown for the circular billiard (□), the asymmetric Sinai billiard (◯), the ring (△), and the circular cavity with perpendicular leads (+). The benchmarks were run on Pentium 4 processor with 2.8 GHz and 2 GBs of memory.

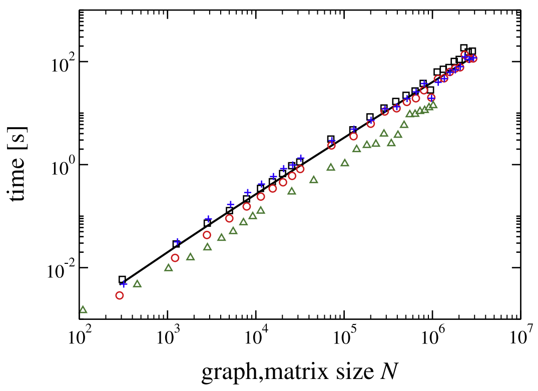

Fig. 11. Time spent for matrix reordering as a function of the total grid (matrix) size N, for the circular billiard (□), the asymmetric Sinai billiard (○), the ring (△), and the circular cavity with perpendicular leads (+). The solid line is a fit to the predicted scaling of the computational complexity, $N \log N$.

In the examples of this section, we considered the pedagogic case of charge transport on a square, finite difference grid. The approach presented here can, however, immediately applied to more complex situations, such as spin transport, as reviewed in Ref. [60]. In addition, extending the transport calculation to a different grid is straight-forward, as any tight-binding grid can be encoded into a graph. The block-tridiagonalization algorithm has already been applied to the case of the hexagonal grid of graphene [61] (for a review on graphene see [62]). A further example of this versatility is shown in the next section, where we apply the block-tridiagonalization algorithm to solve multi-terminal structures involving different tight-binding models.

3.2. Multi-terminal structures

In the previous section, we demonstrated that matrix reordering increases the performance of quantum transport algorithms for two-terminal structures and additionally makes it possible to apply these conventional algorithms to non-trivial structures. Whereas there is a great variety of quantum transport algorithms for systems with two leads, there are only few algorithms that are suitable for multi-terminal structures, and most of them are restricted to rather specific geometries (e.g. Ref. [30]). Only recently algorithms have been developed that claim to be applicable to any multi-terminal structure. The *knitting algorithm* of Ref. [63] is a variant of the RGF algorithm where the system is built up adding every lattice point individually, instead of adding whole blocks of lattice points at a time. Therefore, instead of a matrix multiplication, the central computational step is an exterior product of vectors. Unfortunately, this implies that the knitting algorithm cannot use highly optimized matrix multiplication routines (Level 3 BLAS operations), that are usually much more efficient than their vector counterparts (Level 2 BLAS operations), as discussed in Ref. [59]. Another multi-terminal transport algorithm presented recently [64], is based on the transfer matrix approach. However, it requires the Hamiltonian to be in a specific block-tridiagonal form, and the corresponding level set is set up manually.

Here we show how to employ the block-tridiagonalization algorithm in order to apply the well-established two-terminal quantum transport algorithms to an arbitrary multi-terminal system. The basic idea is sketched in Fig. 12(a): combining several (real) leads into only two virtual leads, the multi-terminal problem is reduced to an equivalent two-terminal problem. After reordering, the resulting problem can then be solved by conventional two-terminal algorithms. Note that in this approach the number of matrix blocks is given by the shortest path between leads in two different virtual leads. If all leads are very close together, this may lead to only few, large blocks in the reordered matrix and, respectively, levels in the graph partitioning, leading to a very large weight $w(H)$. In such a case it is advisable to collect all leads into a single virtual lead. The second virtual lead is then formed by a vertex in the graph, that is furthest away from all leads as depicted in Fig. 12(b). Such a vertex can be found by a BFS search originating from all leads. Thereby the number of matrix blocks/levels is maximized. In fact, this approach yields a block-tridiagonal matrix structure as required by the algorithm of Ref. [64].

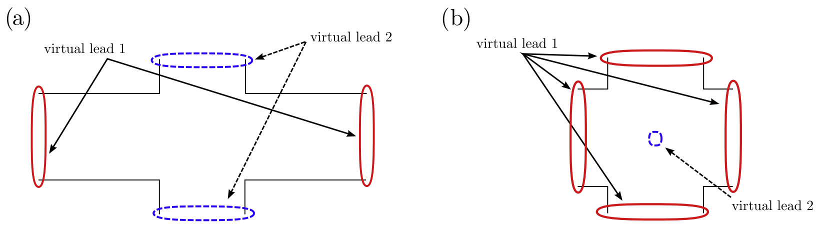

Fig. 12. A multi-terminal structure can be reduced to an equivalent two-terminal structure by collecting all leads in two *virtual leads*. (a) The leads are redistributed into two virtual leads. (b) All leads are combined in a single virtual lead, the second virtual lead is formed by a vertex furthest away.

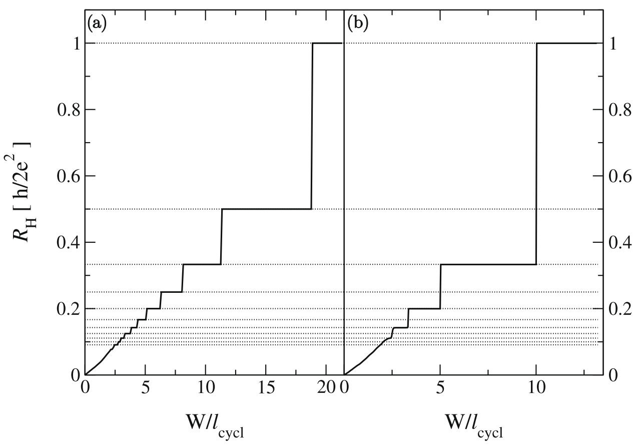

Fig. 13. Example of a four-terminal calculation: quantum Hall effect (a) in a two-dimensional electron gas and (b) in graphene. The Hall resistance R_H is shown as a function of W/l_{cycl}, where W is the width of the Hall bar and l_{cycl} the cyclotron radius in a magnetic field B. Note that $W/l_{cycl} \propto B$. The dotted lines indicate the quantized values of the Hall resistance, $h/2e^2 \times n^{-1}$, where n is a positive integer.

We now demonstrate these strategies on the example of the quantum Hall effect (QHE) in a 2DEG formed in a semiconductor heterostructure [65] and in graphene [66]. For this we use a four-terminal Hall bar geometry as sketched in Fig. 12(a), on top of a square lattice (finite difference approximation to 2DEG) and a hexagonal lattice. Again, it should be emphasized that the code of the actual transport calculation is the same as employed in the two-terminal examples of the previous section. The results of the calculation are shown in Fig. 13, where the integer QHE of the 2DEG and the odd-integer QHE of graphene are clearly visible.

The methods outlined above make it possible to calculate quantum transport in *any* system described by a tight-binding Hamiltonian. This generality is one of the main advantages gained by using the matrix reordering. However, generality also implies that it is difficult to make use of properties of specific systems, such as symmetries, in order to speed-up calculations. Special algorithms developed specifically for a certain system however can, and will usually be faster than a generic approach—at the cost of additional development time.

In the case of the Hall geometry in a 2DEG, such a special algorithm was presented by Baranger et al. [30], and we have implemented a variant of it. Comparing the computing times for the Hall bar geometry in a 2DEG, we find that the special algorithm is only a factor of 1.6–1.7 faster than our generic approach. Although such a performance comparison may depend crucially on the details of the system under consideration, experience shows that the use of the generic approach often does not come with a big performance penalty.

4. Conclusions

We have developed a block-tridiagonalization algorithm based on graph partitioning techniques that can serve as a preconditioning step for a wide class of quantum transport algorithms. The algorithm can be applied to any Hamiltonian matrix originating from an arbitrary tight-binding formulation and brings this matrix into a form that is more suitable for many two-terminal quantum transport algorithms, such as the widely used recursive Green's function algorithm. The advantages of this reordering are twofold: first, the reordering can speed-up the transport calculation significantly. Second, it allows for applying conventional two-terminal algorithms to non-trivial geometries including non-collinear leads and multi-terminal systems. The block-tridiagonalization algorithm scales as $\mathcal{O}(N_{grid} \log N_{grid})$, where N_{grid} is the size of the Hamiltonian matrix, and thus induces only little additional overhead. We have demonstrated the performance of the matrix reordering on representative examples, including transport in 2DEGs and graphene.

The block-tridiagonalization algorithm can operate as a black box and serve as the foundation of a generic transport code that can be applied to arbitrary tight-binding systems. Such a generic transport code is desirable, as it minimizes development time and increases code quality, as only few basic transport routines are necessary, that can be tested thoroughly.

5. Acknowledgments

We acknowledge financial support from DFG within GRK639 and SFB689.

Appendix A. The Fiduccia–Mattheyses algorithm

A.1. Basic idea

The Fiduccia–Mattheyses (FM) algorithm [45] is a variant of the Kernighan–Lin (KL) algorithm [43,44]. Both algorithms are heuristics to improve an existing graph or hypergraph bisection, and are based on the concept of *gain*. The gain of a vertex in a bisection is defined as the change in weight, namely the number of cut edges or nets, that occurs when this vertex is moved to the other part. This gain can also be negative, if such a move increases the number of cut edges or nets. The basic idea of the FM algorithm is to move a vertex with the highest gain from one part to the other, while obeying some balance criterion. In contrast, the KL heuristic is based on swapping pairs of vertices with the highest gain. The fact that the highest gain can be negative, helps the FM algorithm to escape local minima. In particular, the vertex with the highest gain may not be a surface vertex. After moving, the respective vertex is locked in order to avoid an infinite loop, where a single vertex might be swapped back and forth repeatedly. The FM pass ends, when all (free) vertices have been moved, and the best bisection encountered during the pass is returned as result. Further passes can then successively improve on this bisection.

Due to the restriction of moving a single vertex and a clever use of data structures (see Appendix A.2), in the case of graph partitioning a FM pass scales linearly with the number of edges in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{O}(|\mathcal{E}|)$, whereas a KL pass scales cubically with the number of vertices, $\mathcal{O}(|\mathcal{V}|^3)$. Note that in the case of hypergraph bisection, $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, a FM pass scales in principle linearly with the sum of the number of vertices in all nets, $\mathcal{O}(\sum_j |n_j|)$ [45]. However, for hypergraphs arising from structurally symmetric matrices with a nonzero diagonal, $\sum_j |n_j| \sim |\mathcal{E}|$, as seen from Eq. (15). Because of this linear scaling, the FM algorithm is preferable to the KL heuristic.

A.2. Implementation

We have implemented the FM algorithm as described originally by Fiduccia and Mattheyses [45]. In particular, we use a bucket list in order to efficiently choose a vertex with the highest gain. This is possible, as the absolute value of the maximum vertex gain is bounded: for graph partitioning, the vertex gain can take values in the range $-p_{\text{e,max}}, \ldots, p_{\text{e,max}}$, where $p_{\text{e,max}}$ is the maximum number of incident edges to a vertex. For hypergraph partitioning, the range is $-p_{\text{n,max}}, \ldots, p_{\text{n,max}}$, where $p_{\text{n,max}}$ is the maximum number of incident nets to a vertex. Note that for a structurally symmetric matrix with a nonzero diagonal $p_{\text{n,max}} = p_{\text{e,max}} + 1$, as seen from Eq. (15).

For the problem of min-net-cut-min-cut optimization both the graph and the hypergraph structure, and hence the vertex gains both in terms of nets, g_{net}, and in terms of edges, g_{edge}, are considered simultaneously. It is then useful to define a global gain

$$g' = g_{\text{net}} \times (2p_{\text{e,max}} + 1) + g_{\text{edge}} \tag{A.1}$$

which can take values in the range $-p'_{\max}, \ldots, p'_{\max}$, where $p'_{\max} = 2p_{\text{e,max}} \times p_{\text{n,max}} + p_{\text{e,max}} + p_{\text{n,max}}$.

References

[1] R. Landauer, Spatial variation of currents and fields due to localized scatterers in metallic conduction, IBM J. Res. Dev. 1 (3) (1957) 223.
[2] M. Büttiker, Y. Imry, R. Landauer, S. Pinhas, Generalized many-channel conductance formula with application to small rings, Phys. Rev. B 31 (10) (1985) 6207–6215.
[3] A.D. Stone, A. Szafer, What is measured when you measure a resistance? – the Landauer formula revisited, IBM J. Res. Dev. 32 (3) (1988) 384–413.
[4] D.S. Fisher, P.A. Lee, Relation between conductivity and transmission matrix, Phys. Rev. B 23 (12) (1981) 6851–6854.
[5] H.U. Baranger, A.D. Stone, Electrical linear-response theory in an arbitrary magnetic field: a new Fermi-surface formation, Phys. Rev. B 40 (12) (1989) 8169–8193.
[6] H. Haug, A.-P. Jauho, Quantum Kinetics in Transport and Optics of Semiconductors, Springer, Berlin, Heidelberg, 1998.
[7] G.E. Kimball, G.H. Shortley, The numerical solution of Schrödinger's equation, Phys. Rev. 45 (11) (1934) 815–820.
[8] L. Pauling, E.B. Wilson, Introduction to Quantum Mechanics, Dover, New York, 1935.
[9] D. Frustaglia, M. Hentschel, K. Richter, Aharonov–Bohm physics with spin. II. Spin-flip effects in two-dimensional ballistic systems, Phys. Rev. B 69 (15) (2004) 155327.
[10] P. Havu, V. Havu, M.J. Puska, R.M. Nieminen, Nonequilibrium electron transport in two-dimensional nanostructures modeled using green's functions and the finite-element method, Phys. Rev. B 69 (11) (2004) 115325.
[11] R.C. Bowen, G. Klimeck, R.K. Lake, W.R. Frensley, T. Moise, Quantitative simulation of a resonant tunneling diode, J. Appl. Phys. 81 (7) (1997) 3207–3213.
[12] S. Sanvito, C.J. Lambert, J.H. Jefferson, A.M. Bratkovsky, General Green's-function formalism for transport calculations with *spd* Hamiltonians and giant magnetoresistance in Co- and Ni-based magnetic multilayers, Phys. Rev. B 59 (18) (1999) 11936–11948.
[13] M. Luisier, A. Schenk, W. Fichtner, G. Klimeck, Atomistic simulation of nanowires in the $sp^3d^5s^*$ tight-binding formalism: from boundary conditions to strain calculations, Phys. Rev. B 74 (20) (2006) 205323.
[14] M. Brandbyge, J.-L. Mozos, P. Ordejón, J. Taylor, K. Stokbro, Density-functional method for nonequilibrium electron transport, Phys. Rev. B 65 (16) (2002) 165401.
[15] A. Di Carlo, A. Pecchia, L. Latessa, T. Frauenheim, G. Seifert, Tight-binding DFT for molecular electronics (gDFTB), in: G. Cuniberti, G. Fagas, K. Richter (Eds.), Introducing Molecular Electronics, Springer, Berlin, Heidelberg, 2006, pp. 153–184.
[16] A.R. Rocha, V.M. García-Suárez, S. Bailey, C. Lambert, J. Ferrer, S. Sanvito, Spin and molecular electronics in atomically generated orbital landscapes, Phys. Rev. B 73 (8) (2006) 085414.
[17] S. Datta, Electronic Transport in Mesoscopic Transport, Cambridge University Press, Cambridge, 2002.
[18] D.K. Ferry, S.M. Goodnick, Transport in Nanostructures, Cambridge University Press, Cambridge, 2001.
[19] M.P. Lopez Sancho, J.M. Lopez Sancho, J. Rubio, Quick iterative scheme for the calculation of transfer matrices: application to Mo(100), J. Phys. F: Met. Phys. 14 (5) (1984) 1205–1215.
[20] M.P. Lopez Sancho, J.M. Lopez Sancho, J. Rubio, Highly convergent schemes for the calculation of bulk and surface Green functions, J. Phys. F: Met. Phys. 15 (4) (1985) 851–858.
[21] P.S. Krstić, X.-G. Zhang, W.H. Butler, Generalized conductance formula for the multiband tight-binding model, Phys. Rev. B 66 (20) (2002) 205319.
[22] T. Usuki, M. Takatsu, R.A. Kiehl, N. Yokoyama, Numerical analysis of electron-wave detection by a wedge-shaped point contact, Phys. Rev. B 50 (11) (1994) 7615–7625.
[23] T. Usuki, M. Saito, M. Takatsu, R.A. Kiehl, N. Yokoyama, Numerical analysis of ballistic-electron transport in magnetic fields by using a quantum point contact and a quantum wire, Phys. Rev. B 52 (11) (1995) 8244–8255.

[24] C.J. Lambert, D. Weaire, Decimation and Anderson localization, Phys. Stat. Solidi (b) 101 (2) (1980) 591–595.
[25] M. Leadbeater, C.J. Lambert, A decimation method for studying transport properties of disordered systems, Ann. Phys. 7 (5-6) (1998) 498–502.
[26] D. Mamaluy, D. Vasileska, M. Sabathil, T. Zibold, P. Vogl, Contact block reduction method for ballistic transport and carrier densities of open nanostructures, Phys. Rev. B 71 (24) (2005) 245321.
[27] D.J. Thouless, S. Kirkpatrick, Conductivity of the disordered linear chain, J. Phys. C: Solid State Phys. 14 (3) (1981) 235–245.
[28] P.A. Lee, D.S. Fisher, Anderson localization in two dimensions, Phys. Rev. Lett. 47 (12) (1981) 882–885.
[29] A. MacKinnon, The calculation of transport properties and density of states of disordered solids, Z. Phys. B 59 (4) (1985) 385–390.
[30] H.U. Baranger, D.P. DiVincenzo, R.A. Jalabert, A.D. Stone, Classical and quantum ballistic-transport anomalies in microjunctions, Phys. Rev. B 44 (19) (1991) 10637–10675.
[31] R. Lake, G. Klimeck, R.C. Bowen, D. Jovanovic, Single and multiband modeling of quantum electron transport through layered semiconductor devices, J. Appl. Phys. 81 (12) (1997) 7845–7869.
[32] A. Lassl, P. Schlagheck, K. Richter, Effects of short-range interactions on transport through quantum point contacts: a numerical approach, Phys. Rev. B 75 (4) (2007) 045346.
[33] P. Drouvelis, P. Schmelcher, P. Bastian, Parallel implementation of the recursive Green's function method, J. Comput. Phys. 215 (2) (2006) 741–756.
[34] T. Kramer, E.J. Heller, R.E. Parrott, An efficient and accurate method to obtain the energy-dependent green function for general potentials, J. Phys.: Conf. Ser. 99 (2008) 012010.
[35] S. Rotter, J.-Z. Tang, L. Wirtz, J. Trost, J. Burgdörfer, Modular recursive Green's function method for ballistic quantum transport, Phys. Rev. B 62 (3) (2000) 1950–1960.
[36] S. Rotter, B. Weingartner, N. Rohringer, J. Burgdörfer, Ballistic quantum transport at high energies and high magnetic fields, Phys. Rev. B 68 (16) (2003) 165302.
[37] U.V. Çatalyürek, C. Aykanat, Decomposing irregularly sparse matrices for parallel matrix-vector multiplication, Lect. Notes Comput. Sci. 1117 (1996) 75–86.
[38] U.V. Çatalyürek, C. Aykanat, Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication, IEEE Trans. Parallel Distr. Syst. 10 (7) (1999) 673–693.
[39] N.E. Gibbs, J. William, G. Poole, P.K. Stockmeyer, An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal. 13 (2) (1976) 236–250.
[40] E. Cuthill, J. McKee, Reducing the bandwidth of sparse symmetric matrices, in: Proceedings of the 24th National Conference, ACM, New York, 1969, pp. 157–172.
[41] A. George, Computer implementation of the finite element method, Tech. Rep. STAN-CS-71-208, Computer Sci. Dept., Stanford Univ., Stanford, CA, 1971.
[42] W.-H. Liu, A.H. Sherman, Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices, SIAM J. Numer. Anal. 13 (2) (1976) 198–213.
[43] B. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Technol. J. 49 (2) (1970) 291–308.
[44] D.G. Schweikert, B.W. Kernighan, A proper model for the partitioning of electrical circuits, in: DAC'72: Proceedings of the 9th Workshop on Design Automation, ACM, New York, NY, USA, 1972, pp. 57–62.
[45] C.M. Fiduccia, R.M. Mattheyses, A linear-time heuristic for improving network partitions, in: DAC'82: Proceedings of the 19th Conference on Design Automation, IEEE Press, Piscataway, NJ, USA, 1982, pp. 175–181.
[46] G. Karypis, V. Kumar, Multilevel *k*-way hypergraph partitioning, VLSI Des. 11 (3) (2000) 285–300.
[47] B. Hendrickson, E. Rothberg, Improving the run time and quality of nested dissection ordering, SIAM J. Sci. Comput. 20 (2) (1998) 468–489.
[48] J. O'Neil, D.B. Szyld, A block ordering method for sparse matrices, SIAM J. Sci. Stat. Comput. 11 (5) (1990) 811–823.
[49] A. Coon, M. Stadtherr, Generalized block-tridiagonal matrix orderings for parallel computation in process flowsheeting, Comput. Chem. Eng. 19 (1995) 787–805.
[50] K.V. Camarda, M.A. Stadtherr, Matrix ordering strategies for process engineering: graph partitioning algorithms for parallel1976)pp72Kse mConfere840Td(,05M