Abusing a hypergraph partitioner for unweighted graph partitioning

B. O. Fagginger Auer and R. H. Bisseling

Mathematics Institute, Utrecht University,
Budapestlaan 6, 3584 CD, Utrecht, the Netherlands
B.O.FaggingerAuer@uu.nl
R.H.Bisseling@uu.nl

Abstract. We investigate using the Mondriaan matrix partitioner for unweighted graph partitioning in the communication volume and edge-cut metrics. By converting the unweighted graphs to appropriate matrices, we measure Mondriaan's performance as a graph partitioner for the 10th DIMACS challenge on graph partitioning and clustering. We find that Mondriaan can effectively be used as a graph partitioner: w.r.t. the edge-cut metric, Mondriaan's average results are within 21% of the best known results as listed in Chris Walshaw's partitioning archive.

1 Introduction

In this paper, we use the Mondriaan matrix partitioner [21] to partition the graphs from the 10th DIMACS challenge on graph partitioning and clustering [1]. In this way, we can compare Mondriaan's performance as a graph partitioner with the performance of the state-of-the-art partitioners participating in the challenge.

An *undirected graph* G is a pair (V, E), with vertices V, and edges E that are of the form $\{u, v\}$ for $u, v \in V$ with possibly $u = v$. For vertices $v \in V$, we denote the set of all of v's *neighbours* by

$$V_v := \{u \in V \mid \{u, v\} \in E\}.$$

Note that vertex v is a neighbour of itself precisely when the self-edge $\{v, v\} \in E$.

Hypergraphs are a generalisation of undirected graphs, where edges can contain an arbitrary number of vertices. A *hypergraph* $\mathcal{G}$ is a pair $(\mathcal{V}, \mathcal{N})$, with vertices $\mathcal{V}$, and nets (or hyperedges) $\mathcal{N}$; nets are subsets of $\mathcal{V}$ that can contain any number of vertices.

Let $\epsilon > 0$, $k \in \mathbf{N}$, and $G = (V, E)$ be an undirected graph. Then a valid solution to the *graph partitioning problem* for partitioning G into k parts with imbalance ϵ, is a partitioning $\Pi : V \to \{1, \ldots, k\}$ of the graph's vertices into k parts, each part $\Pi^{-1}(\{i\})$ containing at most

$$|\Pi^{-1}(\{i\})| \leq (1 + \epsilon) \left\lceil \frac{|V|}{k} \right\rceil, \qquad (1 \leq i \leq k) \tag{1}$$

vertices.

To measure the quality of a valid partitioning we use two different metrics. The *communication volume metric*[1] [1] is defined by

$$\mathrm{CV}(\Pi) := \max_{1 \le i \le k} \sum_{\substack{v \in V \\ \Pi(v)=i}} |\Pi(V_v) \setminus \{\Pi(v)\}|. \tag{2}$$

For each vertex v, we determine the number of different parts $\pi(v)$ in which v has neighbours, except $\Pi(v)$. Then, the communication volume is given by the maximum over i, of the sum of all $\pi(v)$ for vertices v belonging to part i.

The *edge-cut metric* [1], defined as

$$\mathrm{EC}(\Pi) := |\{\{u, v\} \in E \mid \Pi(u) \neq \Pi(v)\}|, \tag{3}$$

measures the number of edges that exist between different parts of the partitioning Π.

Name	Ref.	Graph/ hypergraph	Sequential/ parallel
Chaco	[13]	graph	sequential
Metis	[14]	graph	sequential
Scotch	[18]	graph	sequential
Jostle	[22]	graph	parallel
ParMetis	[16]	graph	parallel
PT-Scotch	[10]	graph	parallel
hMETIS	[15]	hypergraph	sequential
ML-Part	[6]	hypergraph	sequential
Mondriaan	[21]	hypergraph	sequential
PaToH	[8]	hypergraph	sequential
Par*k*way	[20]	hypergraph	parallel
Zoltan	[12]	hypergraph	parallel

Table 1. Overview of available software for partitioning (hyper)graphs from [2].

There exist a lot of different (hyper)graph partitioners, which are summarised in Table 1 from [2]. All partitioners follow a multi-level strategy [5], where the (hyper)graph is coarsened by generating a matching of the (hyper)graph's vertices and contracting matched vertices to a single vertex. Doing this recursively creates a hierarchy of increasingly coarser approximations of the original (hyper)graph. After this has been done, an initial partitioning is generated on the coarsest (hyper)graph in the hierarchy, i.e. the one possessing the smallest number of vertices. This partitioning is subsequently propagated to the finer (hyper)graphs in the hierarchy and refined at each level (e.g. using the Kernighan–Lin algorithm [17]), until we reach the original (hyper)graph and obtain the final partitioning.

[1] We forgo custom edge and vertex weights and assume they are all equal to one, because Mondriaan's hypergraph partitioner does not support net weights.

2 Mondriaan

Mondriaan has been designed to partition the matrix and the vectors for a parallel sparse matrix–vector multiplication, where a sparse matrix A is multiplied by a dense input vector $\mathbf{v}$ to give a dense output vector $\mathbf{u} = A\,\mathbf{v}$ as the result. First, the matrix partitioning algorithm is executed to minimise the total communication volume $\mathrm{LV}(\Pi)$ of the partitioning, defined below, and then the vector partitioning algorithm is executed with the aim of balancing the communication among the processors. The matrix partitioning itself does not aim to achieve such balance, but it is not biased in favour of any processor part either.

Name	Ref.	$\mathcal{V}$	$\mathcal{N}$
Column-net	[7]	$\{r_1, \ldots, r_m\}$	$\{\{r_i \mid 1 \le i \le m, a_{i\,j} \ne 0\} \mid 1 \le j \le n\}$
Row-net	[7]	$\{c_1, \ldots, c_n\}$	$\{\{c_j \mid 1 \le j \le n, a_{i\,j} \ne 0\} \mid 1 \le i \le m\}$
Fine-grain	[9]	$\{v_{i\,j} \mid a_{i\,j} \ne 0\}$	$\underbrace{\{\{v_{i\,j} \mid 1 \le i \le m, a_{i\,j} \ne 0\} \mid 1 \le j \le n\}}_{\text{column nets}}$ $\cup \underbrace{\{\{v_{i\,j} \mid 1 \le j \le n, a_{i\,j} \ne 0\} \mid 1 \le i \le m\}}_{\text{row nets}}$

Table 2. Available representations of an $m \times n$ matrix $A = (a_{i\,j})$ by a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$ in Mondriaan.

Mondriaan uses recursive bipartitioning to split the matrix or its submatrices repeatedly into two parts, choosing the best of the row or column direction in the matrix. The current submatrix is translated into a hypergraph by the column-net or row-net model, respectively (see Table 2). Another possibility is to split the submatrix based on the fine-grain model, and if desired the best split of the three methods can be chosen. The outcome of running Mondriaan is a two-dimensional partitioning of the sparse matrix (i.e., a partitioning where both the matrix rows and columns are split). The number of parts is not restricted to a power of two, as Mondriaan can split parts according to a given ratio such as 2:1. After each split, Mondriaan adjusts the weight balancing goals of the new parts obtained, as the new part that receives the largest fraction of the weight will need to be stricter in allowing an imbalance during further splits than the part with the smaller fraction.

If the input vector and output vector can be partitioned independently, the vector partitioning algorithm usually has enough freedom to achieve a reasonable communication balancing. If the matrix is square, and both vectors must be partitioned in the same way, then there is usually little freedom. Sometimes, the total communication volume must even be increased because of the identical vector partitioning. If the matrix diagonal has only nonzero elements, however, the vector partitioning can be achieved without incurring additional communication by assigning vector components v_i and u_i to the same processor as the diagonal matrix element a_{ii}. More details on the matrix and vector partitioning can be found in [21]; improved methods for vector partitioning are given in [4], see also [3].

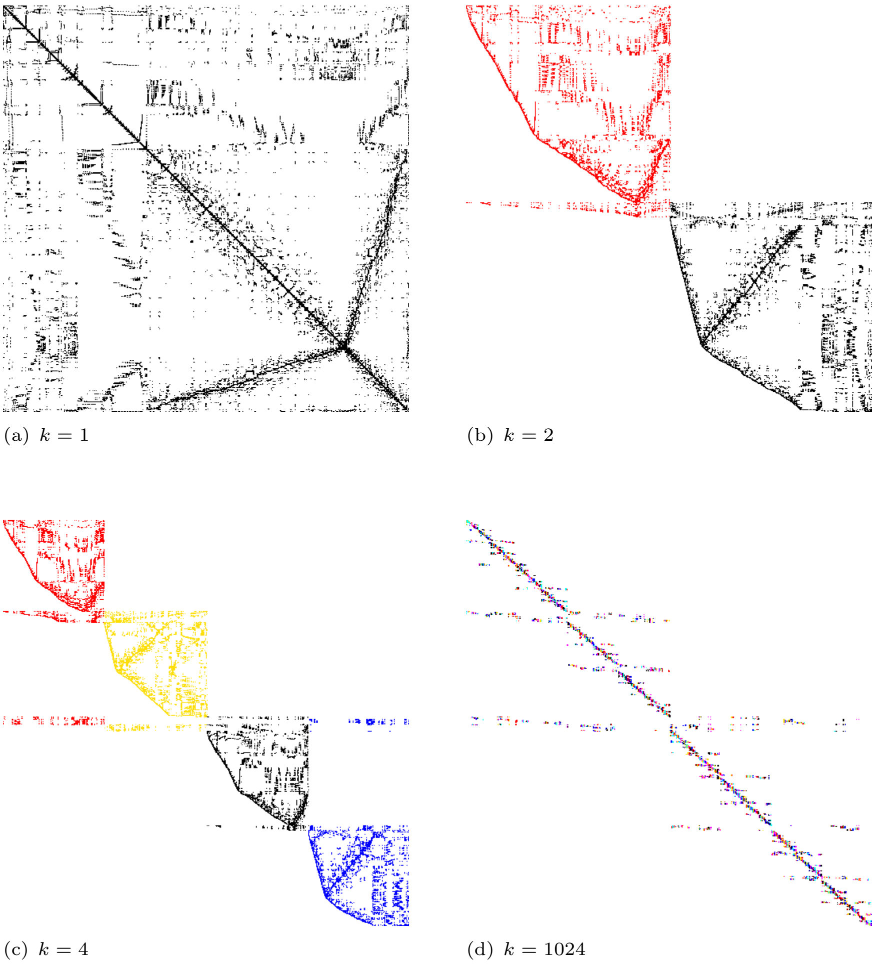


Fig. 1. Mondriaan 1D column partitioning of the graph `fe_tooth`, modelled as a sparse matrix cf. Thm. 1, into $k = 1, 2, 4, 1024$ parts with imbalance $\epsilon = 0.03$. The rows and columns of the matrices have been permuted for $k > 1$ to Separated Block Diagonal form, see [23].

Here, we will use Mondriaan as a hypergraph partitioner, which can be done by choosing the column direction in all splits, so that columns are vertices and rows are nets. This means that we use Mondriaan in one-dimensional mode, as only rows will be split. Fig. 1 illustrates this splitting procedure. Mondriaan has the option to use its own, native hypergraph bipartitioner, or link to the external partitioner PaToH [8]. In the present work, we use the native partitioner.

For the graph partitioning challenge posed by DIMACS, we try to fit the existing software to the aims of the challenge. One could say that this entails abusing the software, as it was designed for a different purpose, namely matrix and hypergraph partitioning. Using a hypergraph partitioner to partition graphs will be at the cost of some additional, unnecessary overhead. Still, it will be interesting to see how the Mondriaan software performs in this unforeseen mode, and to compare the quality of the generated partitionings to the quality of partitionings generated by other software, in particular by graph partitioning packages.

In the situation of the challenge, we can only use the matrix partitioning of Mondriaan and not the vector partitioning, as the vertex partitioning of the graph is already completely determined by the column partitioning of the matrix. The balance of the communication will then solely depend on the balance achieved by the matrix partitioning.

Internally, Mondriaan's hypergraph partitioner solves the following problem. For a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$ with vertex weights $\zeta : \mathcal{V} \to \mathbf{N}$, an imbalance factor $\epsilon > 0$, and a number of parts $k \in \mathbf{N}$, Mondriaan's partitioner produces a partitioning $\Pi : \mathcal{V} \to \{1, \ldots, k\}$ such that

$$\zeta(\Pi^{-1}(\{i\})) \leq (1 + \epsilon) \left\lceil \frac{\zeta(\mathcal{V})}{k} \right\rceil, \qquad (1 \leq i \leq k), \tag{4}$$

where the partitioner tries to minimise the $(\lambda - 1)$-*volume*

$$\mathrm{LV}(\Pi) := \sum_{n \in \mathcal{N}} (|\Pi(n)| - 1). \tag{5}$$

We will now translate the DIMACS partitioning problems from Sec. 1 to the hypergraph partitioning problem that Mondriaan is designed to solve, by creating a suitable hypergraph $\mathcal{G}$, encoded as a sparse matrix A in the row-net model.

2.1 Minimising communication volume

Let $G = (V, E)$ be a given graph, $k \in \mathbf{N}$, and $\epsilon > 0$. Our aim will be to construct a matrix A from G such that minimising eq. (5) subject to eq. (4) enforces minimisation of eq. (2) subject to eq. (1).

To satisfy eq. (1), we need to create one column in A for each vertex in V, such that the hypergraph represented by A in the row-net model will have $\mathcal{V} = V$. This is also necessary to have a direct correspondence between partitionings of the vertices V of the graph and the vertices $\mathcal{V}$ of the hypergraph. Setting the weights ζ of all vertices/matrix columns to 1 will then ensure that eq. (1) is satisfied if and only if eq. (4) is satisfied.

It is a little more tricky to match eq. (2) to eq. (5). Note that because of the maximum in eq. (2), we are not able to create an equivalent formulation. However, as

$$\mathrm{CV}(\Pi) \leq \sum_{i=1}^{k} \sum_{\substack{v \in V \\ \Pi(v)=i}} |\Pi(V_v) \setminus \{\Pi(v)\}| = \sum_{v \in V} |\Pi(V_v) \setminus \{\Pi(v)\}|, \tag{6}$$

we can provide an upper bound, which we can use to limit $\mathrm{CV}(\Pi)$. We need to choose the rows of A, corresponding to nets in the row-net hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$, such that eq. (6) and eq. (5) are in agreement.

For a net $n \in \mathcal{N}$, we have that $n \subseteq \mathcal{V} = V$ is simply a collection of vertices of G, so $|\Pi(n)|$ in eq. (5) equals the number of different parts in which the vertices of n are contained. In eq. (6) we count, for a vertex $v \in V$, all parts in which v has a neighbour, except $\Pi(v)$. Note that this number equals $|\Pi(V_v) \setminus \{\Pi(v)\}| = |\Pi(V_v \cup \{v\})| - 1$.

Hence, we should pick $\mathcal{N} := \{V_v \cup \{v\} \mid v \in V\}$ as the set of nets, for eq. (6) and eq. (5) to agree. In the row-net matrix model, this corresponds to letting A be a matrix with a row for every vertex $v \in V$, filled with nonzeros $a_{v\,v}$ and $a_{v\,w}$ for all $w \in V_v \setminus \{v\}$. Then, for this hypergraph $\mathcal{G}$, we have by eq. (6) that $\mathrm{CV}(\Pi) \leq \mathrm{LV}(\Pi)$. Note that since the communication volume is defined as a maximum, we also have that $k\,\mathrm{CV}(\Pi) \geq \mathrm{LV}(\Pi)$.

Theorem 1. *Let $G = (V, E)$ be a given graph, $k \in \mathbf{N}$, and $\epsilon > 0$. Let A be the $|V| \times |V|$ matrix with entries*

$$a_{v\,w} := \begin{cases} 1 & \text{if } \{v, w\} \in E \text{ or } v = w, \\ 0 & \text{otherwise,} \end{cases}$$

for $v, w \in V$, and $\mathcal{G} = (\mathcal{V}, \mathcal{N})$ the hypergraph corresponding to A in the row-net model with vertex weights $\zeta(v) = 1$ for all $v \in \mathcal{V}$.

Then, for every partitioning $\Pi : V \to \{1, \ldots, k\}$, we have that Π satisfies eq. (1) if and only if Π satisfies eq. (4), and

$$\frac{1}{k}\,LV(\Pi) \leq CV(\Pi) \leq LV(\Pi). \tag{7}$$

2.2 Minimising edge cut

We will now follow the same procedure as in Sec. 2.1 to construct a matrix A such that minimising eq. (5) subject to eq. (4) is equivalent to minimising eq. (3) subject to eq. (1).

As in Sec. 2.1, the columns of A should correspond to the vertices V of G to ensure that eq. (4) is equivalent to eq. (1).

Eq. (3) simply counts all of G's edges that contain vertices belonging to two parts of the partitioning Π. Since every edge contains vertices belonging to at least one part, and at most two parts, this yields

$$\mathrm{EC}(\Pi) = \sum_{e \in E} (|\Pi(e)| - 1).$$

Choosing $\mathcal{N} := E$ will therefore give us a direct correspondence between eq. (5) and eq. (3).

Theorem 2. *Let $G = (V, E)$ be a given graph, $k \in \mathbf{N}$, and $\epsilon > 0$. Let A be the $|E| \times |V|$ matrix with entries*

$$a_{e\,v} := \begin{cases} 1 & \text{if } v \in e, \\ 0 & \text{otherwise,} \end{cases}$$

for $e \in E$, $v \in V$, and $\mathcal{G} = (\mathcal{V}, \mathcal{N})$ the hypergraph corresponding to A in the row-net model with vertex weights $\zeta(v) = 1$ for all $v \in \mathcal{V}$.

Then, for every partitioning $\Pi : V \to \{1, \ldots, k\}$, we have that Π satisfies eq. (1) if and only if Π satisfies eq. (4), and

$$EC(\Pi) = LV(\Pi). \tag{8}$$

With Thm. 1 and Thm. 2, we know how to translate a given graph G to a hypergraph that Mondriaan can partition to obtain solutions to the DIMACS partitioning challenges.

3 Results

We measure Mondriaan's performance as a graph partitioner by partitioning graphs from the `walshaw/` [19] and `matrix/` [11] categories of the DIMACS test bed [1], see Table 3. This is done by converting the graphs to matrices, as expressed by Thm. 1 and Thm. 2, and partitioning these matrices with an updated version of Mondriaan 3.11, using the `onedimcol` splitting strategy (since the matrices represent row-net hypergraphs) with the `lambda1` metric (cf. eq. (5)). The imbalance is set to $\epsilon = 0.03$, the number of parts to $k = 2, 4, \ldots, 1024$, and we average the recorded communication volumes and edge cuts over 10 (`walshaw/`) or 5 (`matrix/`) runs (as Mondriaan uses random tie-breaking) of the Mondriaan partitioner. Note that we did not take the best result of the set of runs, as we are interested in the average performance of Mondriaan. All results were recorded on a dual quad-core AMD Opteron 2378 system with 32GiB of main memory and can be found in Tables 4–6 and Figures 2 and 3.

Results for graphs from the `walshaw/` category for the edge-cut metric, Table 5, can directly be compared with the best known partitionings with 3% imbalance from `http://staffweb.cms.gre.ac.uk/~wc06/partition/` [19]. Compared to the results retrieved on November 2, 2011, we find that Mondriaan performs rather well, except for the graph `add32`. If we take the average of the relative edge cuts over all graphs in `walshaw/` and all values $k = 2, 4, \ldots, 64$, then Mondriaan performs 21% worse than the best results from [19], and only 16% worse if `add32` is excluded. It should be noted that we compare the average edge cut obtained by Mondriaan to the best known edge cuts from [19].

G	$\|V\|$	$\|E\|$
add20	2,395	7,462
data	2,851	15,093
3elt	4,720	13,722
uk	4,824	6,837
add32	4,960	9,462
bcsstk33	8,738	291,583
whitaker3	9,800	28,989
crack	10,240	30,380
wing_nodal	10,937	75,488
fe_4elt2	11,143	32,818
vibrobox	12,328	165,250
bcsstk29	13,992	302,748
4elt	15,606	45,878
fe_sphere	16,386	49,152
cti	16,840	48,232
memplus	17,758	54,196
cs4	22,499	43,858
bcsstk30	28,924	1,007,284
bcsstk31	35,588	572,914
fe_pwt	36,519	144,794
bcsstk32	44,609	985,046
fe_body	45,087	163,734
t60k	60,005	89,440
wing	62,032	121,544
brack2	62,631	366,559
finan512	74,752	261,120
fe_tooth	78,136	452,591
fe_rotor	99,617	662,431
598a	110,971	741,934
fe_ocean	143,437	409,593
144	144,649	1,074,393
wave	156,317	1,059,331
m14b	214,765	1,679,018
auto	448,695	3,314,611

G	$\|V\|$	$\|E\|$
af_shell9	504,855	8,542,010
audikw1	943,695	38,354,076
ldoor	952,203	22,785,136
ecology2	999,999	1,997,996
ecology1	1,000,000	1,998,000
thermal2	1,227,087	3,676,134
af_shell10	1,508,065	25,582,130
G3_circuit	1,585,478	3,037,674
kkt_power	2,063,494	6,482,320
nlpkkt120	3,542,400	46,651,696
cage15	5,154,859	47,022,346
nlpkkt160	8,345,600	110,586,256
nlpkkt200	16,240,000	215,992,816

Table 3. Graphs $G = (V, E)$ from the 10th DIMACS challenge [1] from the `walshaw/` (left) and `matrix/` (right) categories.

G	2	4	8	16	32	64	128	256	512	1024
add20	80	116	142	163	208	-	-	-	-	-
data	66	92	95	87	74	-	-	-	-	-
3elt	47	68	69	74	60	87	-	-	-	-
uk	21	32	42	40	34	26	-	-	-	-
add32	16	32	35	29	27	27	-	-	-	-
bcsstk33	494	734	796	817	635	495	384	375	-	-
whitaker3	65	132	112	107	82	68	-	-	-	-
crack	101	115	131	115	84	71	53	101	121	69
wing_nodal	460	688	564	494	395	273	194	150	-	-
fe_4elt2	66	97	113	103	84	91	52	-	-	-
vibrobox	1,075	1,155	1,047	962	713	560	-	568	-	-
bcsstk29	187	384	398	365	273	245	287	-	-	-
4elt	74	115	106	106	108	81	64	-	-	-
fe_sphere	204	223	192	152	119	92	69	128	-	-
cti	272	560	574	431	319	221	147	151	-	-
memplus	2,608	1,850	1,113	792	732	662	561	692	-	-
cs4	330	520	442	326	244	171	112	79	102	-
bcsstk30	317	675	665	787	738	639	558	489	425	-
bcsstk31	406	564	582	562	515	439	338	293	267	-
fe_pwt	120	145	173	174	182	147	113	157	102	-
bcsstk32	602	785	885	794	629	505	389	349	298	-
fe_body	124	212	234	212	173	145	128	99	90	134
t60k	74	158	173	154	137	113	82	62	46	62
wing	726	987	801	644	480	337	227	148	218	-
brack2	238	661	836	739	619	499	398	309	259	-
finan512	94	123	155	156	88	175	175	206	188	149
fe_tooth	1,299	1,393	1,429	1,224	946	807	582	398	275	-
fe_rotor	574	1,467	1,454	1,297	1,028	813	640	445	379	-
598a	657	1,557	1,570	1,563	1,230	940	740	541	368	283
fe_ocean	274	918	1,172	1,184	971	714	514	354	242	164
144	1,719	2,785	2,340	1,743	1,608	1,325	1,007	675	448	303
wave	2,505	3,266	2,954	2,411	1,790	1,333	906	605	412	284
m14b	955	2,338	2,404	2,234	1,670	1,317	1,087	827	583	390
auto	2,580	5,059	5,177	4,406	3,287	2,581	1,775	1,182	800	545

Table 4. Average communication volume, eq. (2), over 10 Mondriaan runs, for graphs from the `walshaw/` category, Table 3, divided into $k = 2, 4, \ldots, 1024$ parts with imbalance $\epsilon = 0.03$. A '-' indicates that Mondriaan was unable to generate a partitioning satisfying the balancing requirement, eq. (1).

G	2	4	8	16	32	64	128	256	512	1024
`add20`	701	1,239	1,860	2,328	2,742	-	-	-	-	-
`data`	210	428	770	1,303	2,080	-	-	-	-	-
`3elt`	92	221	391	662	1,108	2,008	-	-	-	-
`uk`	22	53	113	190	318	544	-	-	-	-
`add32`	48	109	201	298	493	800	-	-	-	-
`bcsstk33`	10,082	22,289	39,695	59,426	84,167	115,601	150,047	197,525	-	-
`whitaker3`	130	395	731	1,213	1,879	2,815	-	-	-	-
`crack`	197	407	767	1,223	1,915	2,846	4,211	8,861	27,546	28,206
`wing_nodal`	1,750	3,864	6,078	9,206	13,142	17,653	23,074	30,739	-	-
`fe_4elt2`	130	356	658	1,140	1,842	2,866	4,144	-	-	-
`vibrobox`	11,456	21,559	31,780	39,980	48,792	56,222	70,224	100,660	-	-
`bcsstk29`	3,009	8,719	17,760	27,286	41,233	61,076	98,161	-	-	-
`4elt`	144	361	630	1,100	1,819	2,964	4,724	-	-	-
`fe_sphere`	426	844	1,308	2,006	2,902	4,173	5,918	11,041	-	-
`cti`	346	1,011	1,882	3,075	4,400	6,136	8,479	12,184	-	-
`memplus`	5,788	9,829	12,433	14,345	16,081	18,052	21,185	23,664	-	-
`cs4`	402	1,082	1,717	2,484	3,448	4,688	6,234	8,267	15,904	-
`bcsstk30`	6,483	17,522	38,673	80,951	128,679	189,268	272,306	381,846	527,735	-
`bcsstk31`	2,880	7,935	15,805	27,376	44,942	67,652	98,515	140,666	199,171	-
`fe_pwt`	367	796	1,577	3,179	6,112	9,305	12,993	18,544	26,003	-
`bcsstk32`	5,657	11,674	25,134	43,940	69,459	105,490	154,161	225,289	317,952	-
`fe_body`	294	735	1,331	2,184	3,631	5,835	9,403	14,507	22,128	48,374
`t60k`	82	259	548	994	1,632	2,547	3,813	5,565	8,152	12,087
`wing`	912	1,921	2,966	4,551	6,628	9,044	12,010	15,890	24,206	-
`brack2`	713	2,972	7,594	12,697	19,722	29,070	42,271	60,056	84,770	-
`finan512`	162	510	1,125	1,872	2,896	11,089	22,030	39,294	57,481	75,316
`fe_tooth`	4,140	7,892	13,284	20,226	28,577	39,601	53,141	71,917	96,280	-
`fe_rotor`	2,119	8,012	14,289	23,311	35,628	52,139	73,975	102,663	140,333	-
`598a`	2,457	8,343	17,031	28,841	44,104	63,806	88,039	118,654	157,227	209,672
`fe_ocean`	329	1,918	4,720	8,832	14,401	22,074	30,787	42,182	57,226	75,811
`144`	6,812	17,251	28,688	43,192	63,041	87,927	119,688	161,019	214,115	280,451
`wave`	9,206	20,999	34,177	49,883	69,756	95,549	128,205	169,570	220,677	282,531
`m14b`	3,973	13,688	27,861	48,599	75,484	109,861	154,290	212,062	286,225	381,217
`auto`	10,364	28,026	52,424	89,759	134,990	193,265	267,282	360,509	476,470	621,578

Table 5. Average edge cut, eq. (3), over 10 Mondriaan runs, for graphs from the `walshaw/` category, Table 3, divided into $k = 2, 4, \ldots, 1024$ parts with imbalance $\epsilon = 0.03$. A '-' indicates that Mondriaan was unable to generate a partitioning satisfying the balancing requirement, eq. (1).

G	2	4	8	16	32	64	128	256	512	1024
af_shell9	1,156	1,372	2,008	1,918	1,353	1,196	846	640	465	344
audikw1	5,822	12,702	14,279	15,248	11,351	9,043	6,449	4,292	3,034	2,085
ldoor	1,985	2,295	2,660	2,493	1,980	1,809	1,286	1,038	755	619
ecology2	1,031	1,097	1,384	1,087	821	632	467	324	244	171
ecology1	1,017	1,117	1,332	1,086	846	668	490	334	253	169
thermal2	620	1,033	1,050	1,085	1,020	730	594	447	308	228
af_shell10	2,862	3,400	4,013	3,394	2,708	2,068	1,629	1,129	841	599
G3_circuit	1,215	2,025	2,104	1,686	1,361	1,136	911	642	463	343
kkt_power	5,000	7,281	7,951	7,573	6,738	6,459	5,451	4,472	3,327	2,017
nlpkkt120	36,989	35,432	28,552	28,807	22,846	16,719	11,340	7,357	4,824	3,381
cage15	299,452	348,641	316,500	226,511	178,795	123,046	84,141	57,786	38,618	24,727
nlpkkt160	64,842	69,401	51,616	54,384	41,661	30,506	19,552	13,086	8,543	5,603
nlpkkt200	104,129	105,787	83,647	85,272	64,572	43,799	30,925	20,150	13,586	8,950
af_shell9	9,820	23,740	49,135	93,665	145,215	226,575	339,645	493,067	711,203	1,005,116
audikw1	105,949	339,874	805,979	1,379,132	2,087,455	2,994,228	4,240,882	5,845,790	7,760,475	10,047,213
ldoor	25,146	50,516	93,933	171,086	280,695	434,669	666,308	991,939	1,473,505	2,139,374
ecology2	1,280	2,481	4,831	7,779	12,441	18,050	25,487	36,396	51,295	71,290
ecology1	1,258	2,516	4,813	7,639	11,900	17,856	25,638	36,111	51,050	71,974
thermal2	1,049	3,284	7,482	12,837	21,106	31,731	46,753	68,578	98,363	139,836
af_shell10	28,695	60,955	115,525	181,535	284,235	421,786	629,854	905,889	1,282,815	1,817,534
G3_circuit	1,455	3,465	6,146	10,180	14,947	23,839	38,850	57,729	82,776	118,561
kkt_power	21,099	40,750	79,321	134,342	215,321	341,772	503,334	655,129	779,904	873,420
nlpkkt120	303,056	617,142	993,142	1,401,120	1,967,164	2,654,406	3,446,856	4,479,385	5,690,697	7,211,457
cage15	879,668	1,501,558	2,235,240	3,072,493	3,890,359	4,751,166	5,724,787	6,840,355	8,042,563	9,409,982

Table 6. Average communication volume, eq. (2) (top) and edge cut, eq. (3) (bottom), over 5 Mondriaan runs, for graphs from the `matrix/` category, Table 3, divided into $k = 2, 4, \ldots, 1024$ parts with imbalance $\epsilon = 0.03$. For `nlpkkt160` and `nlpkkt200`, the test system ran out of memory while it was partitioning the edge cut matrix.

The strange dip in the communication volume for `finan512` in Table 4 for $k = 32$ parts can be explained by the fact that the graph `finan512` consists of 32 densely connected parts with few connections between them, see the visualisation of this graph in [11], such that there is a natural partitioning with very low communication volume in this case.

In Fig. 2, we plot the time required by Mondriaan to create a partitioning for both communication volume and edge cut. The number of nonzeros in the matrices from Thm. 1 and Thm. 2 equals $2\,|E| + |V|$ and $2\,|E|$, respectively. However, the matrix sizes are equal to $|V| \times |V|$ and $|E| \times |V|$, respectively. Therefore, even though the number of nonzeros in matrices from Thm. 2 is smaller, the larger number of nets (typically $|E| > |V|$, e.g. `nlpkkt200`) will lead to higher processing times and increased memory requirements for the edge-cut matrices, as can be seen when comparing Fig. 2(b) to Fig. 2(a).

We have also investigated the communication volume imbalance, defined for a partitioning Π of G into k parts as

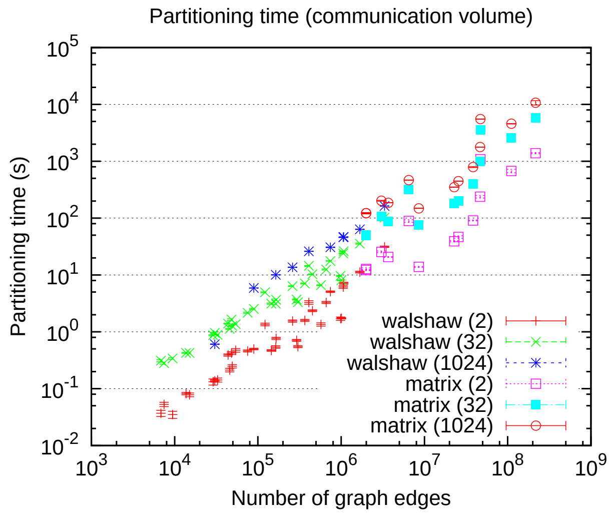
$$\frac{\mathrm{CV}(\Pi)}{\mathrm{LV}(\Pi)/k} - 1. \tag{9}$$

Eq. (9) measures the imbalance in communication volume and can be compared to the factor ϵ for vertex imbalance in eq. (1). We plot eq. (9) as a percentage for a selection of graphs in Fig. 3, where we see that the deviation of the communication volume $\mathrm{CV}(\Pi)$ from perfect balance, i.e. from $\mathrm{LV}(\Pi)/k$, is no more than 140% (for `cage15`, $k = 1024$). Compared to the theoretical upper bound for the imbalance of $k-1$ (via eq. (7)), this is very good. This also means that at most a factor of 2.4 in communication volume per processor can still be gained by improving the communication balance. Therefore, as the number of parts increases, the different parts of the partitionings generated by Mondriaan are not only balanced in terms of vertices, cf. eq. (1), but also in terms of communication volume.

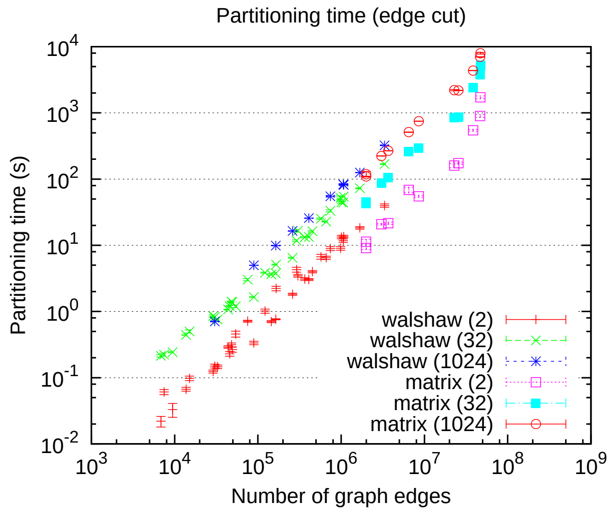
4 Conclusion

We have shown that it is possible to use the Mondriaan matrix partitioner as a graph partitioner by constructing appropriate matrices of a given graph for either the communication volume or edge-cut metric. Mondriaan's performance was measured by partitioning graphs from the 10th DIMACS challenge on graph partitioning and clustering, as well as comparing obtained edge cuts with the best known results from [19]: here Mondriaan's average edge cut was, on average, 21% higher than the best known. From these results we find that Mondriaan can effectively be used to perform graph partitioning.

To our surprise, the partitionings generated by Mondriaan are reasonably balanced in terms of communication volume, as shown in Fig. 3, even though Mondriaan does not perform explicit communication volume balancing during matrix partitioning. We attribute the observed balancing to the fact that the



(a)



(b)

Fig. 2. The average partitioning time required by the Mondriaan partitioner to generate the partitionings from Table 4, 6, (a), and Table 5, 6, (b).

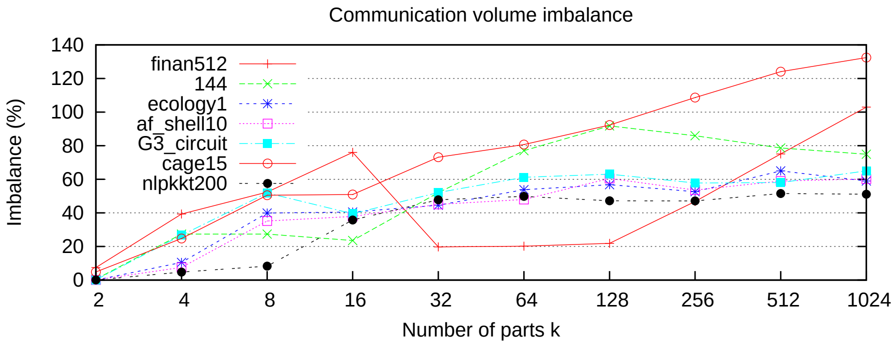


Fig. 3. The communication volume imbalance given by eq. (9), plotted as a percentage for several graphs.

Mondriaan algorithm performs random tie-breaking, without any preference for a specific part of the partitioning.

These tests also indicate the value of extending Mondriaan to take hypergraph net weights into account for the $(\lambda - 1)$-metric, eq. (5), because we could only perform unweighted graph partitioning due to the absence of this feature. We intend to add this feature in a next version of Mondriaan.

References

1. Bader, D.A., Sanders, P., Wagner, D., Meyerhenke, H., Hendrickson, B., Johnson, D.S., Walshaw, C., Mattson, T.G.: 10th DIMACS implementation challenge - graph partitioning and graph clustering (2012), `http://www.cc.gatech.edu/dimacs10/index.shtml`
2. Bisseling, R.H., Fagginger Auer, B.O., Yzelman, A.N., van Leeuwen, T., Çatalyürek, Ü.V.: Two-dimensional approaches to sparse matrix partitioning. In: Naumann, U., Schenk, O. (eds.) Combinatorial Scientific Computing. CRC Press (2012)
3. Bisseling, R.H.: Parallel Scientific Computation: A Structured Approach using BSP and MPI. Oxford University Press, Oxford, UK (2004)
4. Bisseling, R.H., Meesen, W.: Communication balancing in parallel sparse matrix-vector multiplication. Electronic Transactions on Numerical Analysis 21, 47–65 (2005), special Issue on Combinatorial Scientific Computing
5. Bui, T., Jones, C.: A heuristic for reducing fill-in in sparse matrix factorization. In: Proceedings Sixth SIAM Conference on Parallel Processing for Scientific Computing. pp. 445–452. SIAM, Philadelphia, PA (1993)
6. Caldwell, A.E., Kahng, A.B., Markov, I.L.: Improved algorithms for hypergraph bipartitioning. In: Proceedings Asia and South Pacific Design Automation Conference. pp. 661–666. ACM Press, New York (2000)
7. Çatalyürek, Ü.V., Aykanat, C.: Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Transactions on Parallel and Distributed Systems 10(7), 673–693 (1999)

8. Çatalyürek, Ü.V., Aykanat, C.: PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0. Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey. PaToH is available at `http://bmi.osu.edu/~umit/software.htm` (1999)
9. Çatalyürek, Ü.V., Aykanat, C.: A fine-grain hypergraph model for 2D decomposition of sparse matrices. In: Proceedings Eighth International Workshop on Solving Irregularly Structured Problems in Parallel (Irregular 2001). p. 118. IEEE Press, Los Alamitos, CA (2001)
10. Chevalier, C., Pellegrini, F.: PT-Scotch: a tool for efficient parallel graph ordering. Parallel Computing 34(6-8), 318–331 (2008)
11. Davis, T.A., Hu, Y.: The university of florida sparse matrix collection. ACM Transactions on Mathematical Software 38(1), 1–25 (2011)
12. Devine, K.D., Boman, E.G., Heaphy, R.T., Bisseling, R.H., Catalyurek, U.V.: Parallel hypergraph partitioning for scientific computing. In: Proceedings IEEE International Parallel and Distributed Processing Symposium 2006. IEEE Press (2006)
13. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing 16(2), 452–469 (1995)
14. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20(1), 359–392 (1998)
15. Karypis, G., Kumar, V.: Multilevel k-way hypergraph partitioning. In: Proceedings 36th ACM/IEEE Conference on Design Automation. pp. 343–348. ACM Press, New York (1999)
16. Karypis, G., Kumar, V.: Parallel multilevel k-way partitioning scheme for irregular graphs. SIAM Review 41(2), 278–300 (1999)
17. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 29, 291–307 (1970)
18. Pellegrini, F., Roman, J.: Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In: Proceedings High Performance Computing and Networking Europe. Lecture Notes in Computer Science, vol. 1067, pp. 493–498. Springer (1996)
19. Soper, A.J., Walshaw, C., Cross, M.: A combined evolutionary search and multilevel optimisation approach to graph-partitioning. J. of Global Optimization 29, 225–241 (2004)
20. Trifunović, A., Knottenbelt, W.J.: Parallel multilevel algorithms for hypergraph partitioning. Journal of Parallel and Distributed Computing 68(5), 563–581 (2008)
21. Vastenhouw, B., Bisseling, R.H.: A two-dimensional data distribution method for parallel sparse matrix–vector multiplication. SIAM Review 47(1), 67–95 (2005)
22. Walshaw, C., Cross, M.: JOSTLE: Parallel Multilevel Graph-Partitioning Software – An Overview. In: Magoules, F. (ed.) Mesh Partitioning Techniques and Domain Decomposition Techniques, pp. 27–58. Civil-Comp Ltd. (2007)
23. Yzelman, A.N., Bisseling, R.H.: Cache-oblivious sparse matrix–vector multiplication by using sparse matrix partitioning methods. SIAM Journal on Scientific Computing 31(4), 3128–3154 (2009)