Lower Bounds and Exact Algorithms for the Graph Partitioning Problem using Multicommodity Flows

Norbert Sensen*
University of Paderborn
Department of Mathematics and Computer Science
Fürstenallee 11, D 33102 Paderborn
sensen@upb.de

Abstract

In this paper new and generalized lower bounds for the graph partitioning problem are presented. These bounds base on the well known lower bound of embedding a clique into the given graph with minimal congestion which is equivalent to a multicommodity flow problem where each vertex sends a commodity of size one to every other vertex. Our new bounds use arbitrary multicommodity flow instances for the bound calculation, the critical point for the lower bound is the guaranteed cut flow of the instances. Furthermore, a branch&bound procedure basing on these bounds is presented. Finally, upper bounds of the lower bounds are shown which demonstrate the superiority of the presented generalizations; and the new bounds are applied to the Butterfly and Beneš network.

1 Introduction

Graph Partitioning is the problem of partitioning a set of vertices of a graph into disjoint subsets of a given maximal size such that the number of edges with end points in different subsets is minimized. Graph Partitioning is a very common problem and has a large number of applications. For example circuit layout, compiler design, and load balancing are typical applications in which Graph Partitioning problems appear. Unfortunately, the Graph Partitioning problem is a NP-hard problem. So in the last years a lot of effort has been spent in the development of fast and good heuristics for the problem, a recent survey is given in [7]. These heuristics often can handle rather large graphs with more than a million vertices and deliver good solutions. In contrast to the development of heuristics only a little expense has been done in the development of exact algorithms. From the NP-hardness fact it is clear that generally only relatively small graphs can be solved exactly. Nevertheless, exact solutions are of interest for applications and for the validation of heuristics.

In this paper we present new lower bounds for the Graph Partitioning problem and a branch & bound algorithm for the exact solution of the Graph Partitioning problem using these new lower bounds. The new bounds base on a well known method for proving lower bounds of the graph bisection problem (a special case of the graph partitioning problem with two equally sized partitions): If there is an embedding of the Clique graph with n vertices (K_n) into the given graph G with a congestion C, then each bisection of the graph G cuts at least $\frac{n^2}{4C}$ edges. The computation of an embedding with minimal congestion (i.e. the lower bound is maximal) is equivalent to a multicommodity flow problem: Every vertex sends a commodity of size one to every other vertex. Realizing this flow with minimal congestion provides us with the identical lower bound for the graph bisection problem.

The above lower bound is often used for theoretical analysis of the bisection width of a given graph (e.g. see [12]) and it delivers convenient bounds if the graph is quite regular. But the bound is impractical for the construction of a branch & bound algorithm if the graph is more irregular as it is often the fact in practice. In this paper we present generalizations of this bound. The generalizations base on the observation that not

*This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

every vertex has to send a commodity of identical size to every other vertex. In fact we can compute lower bounds on the graph partitioning problem for every possible combination of commodities. We only have to know the $guaranteed\ CutFlow$, i.e. the amount of flow which crosses the cut of every feasible partitioning of the graph. Furthermore, we have generalized the idea such that it is applicable not only for the graph bisection problem with exactly two equally sized partitions, but also for the general graph partitioning problem with a given number of partitions and a given maximal size for every partition.

In the last years there have been presented a number of different approaches for solving the graph partitioning problem exactly. The most recent approach is presented in [10] by S.E. Karisch, F. Rendl, and J. Clausen. They use a semidefinite relaxation for the computation of a lower bound on the Graph Partitioning problem as the core of a branch & bound algorithm. They address the Graph Partitioning problem with two partitions, edge weights, and a maximal partition size; their approach does not handle vertex weights or more than two partitions. In [6] Ferreira et al. present a branch-and-cut algorithm basing on a variety of separation heuristics. They address the general Graph Partitioning problem with vertex weights, edge weights, an arbitrary number of partitions, and a maximal partition size. In [1] L. Brunetta, M. Conforti, and G. Rinaldi present another branch-and-cut algorithm. They start with a linear program defining the convex hull of all solutions and use several separation procedures to add cuts to the linear program. They address the bisection problem with edge weights. In [9] E. Johnson, A. Mehrotra, and G. Nemhauser present a column generation algorithm for the Graph Partitioning problem. There, the generation of additional columns itself is NP-hard, so they present efficient strategies for the generation. They address the general Graph Partitioning problem with vertex and edge weights, an arbitrary number of partitions and a maximal partition size.

Also related to our paper is the work of F. Shahrokhi and L. Szekely in [16]. They examine bounds basing on the embedding of the clique K_n. They apply this bound to general graphs with a small number of equivalence classes of vertices and use only shortest paths for the flow. Putting this together they get lower bounds on the crossing number, bisection width and edge and vertex expansion of graphs. Their bounds are especially good if there is only one equivalence class of vertices.

In the next section we give basic definitions for the Graph Partitioning problem and Multicommodity Flows. Then in section three we present the new lower bounds for the Graph Partitioning problem basing on more flexible multicommodity flows. In section four the branch & bound algorithm using this bound is described; inside the algorithm we can use the solution of the multicommodity flow instance to force vertices to stay in the same partition. Finally, in section five we give upper bounds on the new lower bounds and apply the new lower bounds to the butterfly network.

2 Definitions

2.1 Graph Partitioning

There have been a couple of slightly different definitions of the Graph Partitioning problem. Inside this paper we are talking about a graph with vertex and edge weights, a given number of partitions and a maximal size for each partition. More formally:

Definition 1
The Graph Partitioning problem *has given an undirected graph $G = (V, E)$, vertex weights $g : V \to \mathbb{N}$, edge weights $f : E \to \mathbb{N}$, the number of partitions $p \in \mathbb{N}$, and a maximal size $M \in \mathbb{N}$. The problem is to find a partition of the vertices V into p disjoint sets $V_1, \dots, V_p$ with $\forall i : \sum_{v \in V_i} g(v) \leq M$ and minimal $CutSize := \min_{V=\bigcup V_i} CutSize(V_1, \dots, V_p)$, where*

$$CutSize(V_1, \dots, V_p) := \sum_{v \in V_i, w \in V_j, i<j} f(\{v, w\}).$$

In the following we use $n = |V|$ and $N = \sum_{v \in V} g(v)$. The above definition is a very general one. If $p = 2$ is used we obtain the well known bisection problem. Often equally sized partitions are requested, i.e. $M = \lceil \frac{N}{p} \rceil$. Notice, that in the case of $p > 2$ this is a less strong requirement than the requirement $\forall i, j : |V_i| - |V_j| \leq 1$ which is also used sometimes.

2.2 Multicommodity Flows

As mentioned above we want to use Multicommodity Flows in order to compute lower bounds on the Graph Partitioning problem. In the Multicommodity Flow problem there are commodities of specific sizes with a source vertex and a destination vertex. The goal is to fulfill the given set of commodities with minimal congestion. More formally:

Definition 2
The Multicommodity Flow problem has given an undirected graph $G = (V, E)$, edge weights $f : E \to \mathbb{N}$, and for each pair $(v, w) \in V^2$ the size $d_{v,w} \in \mathbb{R}_{\geq 0}$ with $\forall v \in V : d_{v,v} = 0$ of the commodity which has to flow from vertex v to vertex w. A flow $h : V \times V \times V \to \mathbb{R}_{\geq 0}$ is sought with $h(u, v, w) = 0 \Leftarrow \{v, w\} \notin E$, and

$$\forall v, w \in V, v \neq w : \sum_{u \in V} h(v, u, w) - h(v, w, u) = d_{v,w}.$$

The congestion $c : E \to \mathbb{R}$ of each edge is $c(\{v, w\}) := \frac{1}{f(\{v,w\})} \sum_{u \in V} h(u, v, w) + h(u, w, v)$. A flow h is searched with minimal total congestion $C := \max_{e \in E} c(e)$.

The Multicommodity Flow problem is a well known problem. It can be represented as a linear program of polynomial size, so it is solvable in polynomial time. Fast or approximating algorithms for the Multicommodity Flow problem are subject of current research activities, see e.g. [11, 15, 17].

3 Multicommodity Bounds

3.1 General Idea

The main idea for the generalization of the known lower bound on the graph bisection problem into equally sized partitions is the following: The known bound bases on the embedding of a K_n into the given graph with minimal congestion C'. Any bisection of a K_n has a $CutSize$ of at least $\frac{n^2}{4}$. Since each edge of the given graph is used by at most C' edges of the K_n, a $CutSize$ of at least $\frac{n^2}{4C'}$ of the given graph is unavoidable in order to cut the minimal number of edges of the K_n. Computing the minimal congestion of this embedding is equivalent to calculating the minimal congestion C of a Multicommodity Flow problem with $\forall v, w \in V : d_{v,w} = 1$. In this case we can argue equivalently to get a lower bound for the Graph Partitioning problem: Any bisection of the graph has a flow of $\frac{n^2}{2}$ between vertices of different partitions. Since each vertex transports at most C commodities (assuming $\forall e \in E : f(e) = 1$) at least $\frac{n^2}{2C}$ crossing edges are necessary, i.e. $CutSize \geq \frac{n^2}{2C}$.

This consideration can be generalized to any Multicommodity Flow instance:

Theorem 1
Given a Graph Partitioning problem with a graph G which has to be divided into p partitions. If a Multicommodity Flow with this graph G, sizes d, and congestion C exists with a $CutFlow \in \mathbb{R}$ such that for any optimal partition $V = \bigcup_{i=1}^{p} V_i$ $\sum_{v \in V_i, w \in V_j, i \neq j} d_{v,w} \geq CutFlow$ holds. Then

$$CutSize \geq \frac{CutFlow}{C}.$$

Proof:
Let $V = \bigcup V_i$ be an optimal solution of the Graph Partitioning problem. Then

$$CutFlow \leq \sum_{v \in V_i, w \in V_j, i \neq j} d_{v,w} \leq \sum_{v \in V_i, w \in V_j, i < j} C \cdot f(\{v, w\}) = C \cdot CutSize(V_1, \ldots, V_p)$$

Since we have assumed that $V = \bigcup V_i$ is optimal, $CutSize \geq \frac{CutFlow}{C}$ follows.. ■

3.2 Different Instantiations

So in principle we could use any Multicommodity Flow instance for the computation of a lower bound on the Graph Partitioning problem. The bound is the better the bigger the guaranteed $CutFlow$ and the smaller the congestion C are. In the following we introduce three different general Multicommodity Flow instances with a different degree of freedom for their choice of the sizes of the commodities. The first one corresponds to the known bound where every vertex sends a commodity of size one to every other vertex. We have adapted it to consider vertex weights, only:

Definition 3
The 1-1-MC *is a Multicommodity Flow instance with a graph* $G = (V, E)$*, vertex weights* $g : V \to I\!N$*, and default sizes of the commodities:*

$$\forall v, w \in V : \quad d_{v,w} = g(v) \cdot g(w)$$

The next Multicommodity Flow instance allows a variable source strength for each vertex. The idea behind this is that more central vertices generate less total load on the edges such that we hope to get a larger $CutFlow$ with a smaller congestion:

Definition 4
The VarMC *is a Multicommodity Flow instance with a graph* $G = (V, E)$*, vertex weights* $g : V \to I\!N$*, and a free source strength* $s : V \to I\!R_{\geq 0}$ *such that*

$$\forall v, w \in V : \quad d_{v,w} = g(w) \cdot s(v).$$

For the computation of a good lower bound on the graph partitioning problem the free source strengths have to be adapted to the given graph such that the bound is maximized. The selecting of the source strengths is not done by hand but can be included into a linear program which also solves the Multicommodity Flow problem. Finally, we introduce a Multicommodity Flow instance in which all sizes of the commodities are free:

Definition 5
The MVarMC *is a Multicommodity Flow instance with a graph* $G = (V, E)$*, vertex weights* $g : V \to I\!N$*, and a free source strength* $s : V \times V \to I\!R_{\geq 0}$ *with* $s(v, v) = 0 \; \forall v \in V$ *such that*

$$\forall v, w \in V : \quad d_{v,w} = g(w) \cdot s(v, w).$$

The critical point for the use of a Multicommodity Flow instance when computing a lower bound on the graph partitioning problem is the guaranteed $CutFlow$. The following Theorem delivers correct values for the MVarMC instance:

Theorem 2
We have given a Graph Partitioning problem with a graph $G = (V, E)$*, vertex weights* g*, and the constants* p *and* M*. A Multicommodity Flow instance MVarMC guarantees a flow over each feasible partition of*

$$CutFlow \geq \sum_{v \in V} \left(\sum_{w \in V} s(v, w) \cdot g(w) - \bar{s}_v (M - g(v)) \right) + \tilde{s} R (M - R)$$

with $\bar{s}_v := max_{w \in V} s(v, w)$*,* $\tilde{s} := min_{v \in V} \frac{\bar{s}_v}{g(v)}$*, and* $R := N - M \cdot \lfloor \frac{N}{M} \rfloor$*.*

Proof:
We are looking for a guaranteed $CutFlow$ of the commodities with sizes $s(v, w)$. A $CutFlow$ is guaranteed if for any possible partition according to the parameters M and p the actual flow between the partitions is at least as large as $CutFlow$. So let us assume any feasible partition $V = V_1 \cup \ldots \cup V_p$ with $N_i = \sum_{v \in V_i} g(v)$. Then

$$CutFlow \;\; = \;\; \sum_{i=1}^{p} \sum_{v \in V_i} \sum_{w \in V \setminus V_i} s(v, w) \cdot g(w)$$

$$
\begin{aligned}
&= \sum_{i=1}^{p} \sum_{v \in V_i} \left(\sum_{w \in V} s(v,w) \cdot g(w) \; - \sum_{w \in V_i} s(v,w) \cdot g(w) \right) \\
&\geq \sum_{i=1}^{p} \sum_{v \in V_i} \left(\sum_{w \in V} s(v,w) \cdot g(w) \; - \; \bar{s}_v \sum_{w \in V_i \setminus \{v\}} g(w) \right) \\
&= \sum_{i=1}^{p} \sum_{v \in V_i} \left(\sum_{w \in V} s(v,w) \cdot g(w) \; - \; (N_i - g(v)) \cdot \bar{s}_v \right) \\
&= \sum_{v \in V} \left(\sum_{w \in V} s(v,w) g(w) \; - \; (M - g(v)) \cdot \bar{s}_v \right) \quad + \sum_{i=1}^{p} (M - N_i) \sum_{v \in V_i} \bar{s}_v \\
&\geq \sum_{v \in V} \left(\sum_{w \in V} s(v,w) g(w) \; - \; (M - g(v)) \cdot \bar{s}_v \right) \quad + \sum_{i=1}^{p} (M - N_i) \sum_{v \in V_i} g(v) \tilde{s} \\
&= \sum_{v \in V} \left(\sum_{w \in V} s(v,w) g(w) \; - \; (M - g(v)) \cdot \bar{s}_v \right) \quad + \; \tilde{s} \cdot \sum_{i=1}^{p} (M - N_i) N_i
\end{aligned}
$$

To prove the Theorem it remains to show that $\sum_{i=1}^{p}(M - N_i)N_i \geq R(M - R)$ for every feasible partition, i.e. $\forall i : N_i \leq M \;\wedge\; \sum_{i=1}^{p} N_i = N$:

$$
\sum_{i=1}^{p}(M - N_i)N_i = MN - \sum_{i=1}^{p} N_i^2 \geq MN - \lfloor \frac{N}{M} \rfloor M^2 - R^2 = R(M - R)
$$

In the above formulations we have used $\sum N_i^2 \leq \lfloor \frac{N}{M} \rfloor M^2 + R^2$. This is true since the sum $\sum N_i^2$ is maximal if as many N_i's as possible have the maximal value M, i.e. we have $\lfloor \frac{N}{M} \rfloor$-times the value M and one of the N_i's has the value R. We can see that this selection of the parameters N_i leads to the maximal sum by the following consideration: Let us assume two of the parameters are less than M, i.e. without loss of generality $0 < N_2 \leq N_1 < M$. We could then increase N_1 and decrease N_2 and get a bigger value for the sum $\sum N_i^2$ since: $(N_1 + 1)^2 + (N_2 - 1)^2 = N_1^2 + N_2^2 + 2(N_1 - N_2) \geq N_1^2 + N_2^2 \quad \Leftrightarrow \quad N_1 \geq N_2$. ■

Notice that the term $R(M - R)$ corresponds to the problem definition of a maximal size for every partition. If the restriction $\forall i, j : |V_i| - |V_j| \leq 1$ is used, a different (bigger) term is possible. Basing on the $CutFlow$ of the MVarMC instance it is easy to specify the guaranteed $CutFlow$ of the VarMC and 1-1-MC instance:

Lemma 1
The VarMC instance guarantees a $CutFlow$ of

$$
CutFlow \geq (N - M) \sum_{v \in V} s(v) + \tilde{s} R(M - R)
$$

with $\tilde{s} := \min_{v \in V} \frac{s(v)}{g(v)}$. The 1-1-MC instance guarantees a $CutFlow$ of

$$
CutFlow \geq N(N - M) + R(M - R).
$$

Proof:
Using $\forall v, w \in V : \; s(v,w) = s(v)$ or $\forall v, w \in V : \; s(v,w) = g(v)$, respectivly, the Lemma follows from the $CutFlow$ of the MVarMC instance. ■

So altogether we have introduced three different instances with a different degree of freedom. It is clear that the bound of the MVarMC instance is at least as good as the bound of the VarMC instance which is at least as good as the bound of the 1-1-MC instance. On the other hand the computation of the MVarMC instance should last longer since more variables have to be specified. For practical usage of the bounds we have to compare the quality of the bounds and their running times.

3.3 Experimental Results

All three proposed Multicommodity Flow instances can be represented as a linear program. The free source strengths in the VarMC and MVarMC instances are realized by additional variables inside the linear program. An arbitrary but constant maximal congestion is given and the goal is the maximization of the guaranteed $CutFlow$. Thus, solving the linear program contains the selection of the optimal source strengths. The following linear program corresponds to the MVarMC instance:

$$\begin{array}{lll} \text{Maximize} & \sum_{v\in V}\left(\sum_{w\in V} s(v,w)\cdot g(w) - \bar{s}_v(M-g(v))\right) + \tilde{s}R(M-R) & \\ \text{subject to} & \forall\{v,w\}\in E: & \sum_{u\in V} h(u,v,w)+h(u,w,v) \leq f(\{v,w\}) \\ & \forall v,w\in V, v\neq w: & \sum_{u\in V} h(v,u,w)-h(v,w,u) \geq s(v,w)\cdot g(w) \\ & \forall v,w\in V, v\neq w: & \bar{s}_v \geq s(v,w) \\ & \forall v\in V: & g(v)\cdot\tilde{s} \leq \bar{s}_v \end{array}$$

Of course the constraints for $\tilde{s}$ can be omitted if $R = 0$. To solve this linear program we use the PCx package, an open implementation of the interior point method, see [4, 5]. We have also made some experiments with the CPLEX package ([8]), but decided to use PCx since on the one hand the running times for computing the exact bound are comparable and on the other hand PCx is an open package such that we are able to change the code. For further details see subsection 4.4. All experiments have been performed on a Sun Enterprise 450 Model 4400 machine with 1 GB main memory and a Sun UltraSPARC-II 400 MHz processor.

We have done a large amount of tests for different Graph Partitioning problems. Here we show the results of a small representative set of graphs in order to show the typical behavior of the bounds. The “SEd” graph is a shuffle exchange graph of dimension d, the “DBd” graph is a DeBruijn graph of dimension d, “Star 50” is a simple star graph with 50 vertices, “Grid 5x10” is a two-dimensional grid. “ex36a” and “ex36d” are introduces in [10], both having 36 vertices. In the “a” version each edge is in the graph with probability 0.5, in the “d” version each edge has a random weight between 0 and 10. “Rand 0.1” and “Rand 0.05” are random graphs from ourselves, both having 60 vertices; each possible edge is contained in the graph with probability of 0.1 or 0.05, respectively. “RandPlan” is a random maximal planar graph with 100 vertices, constructed from routines of the LEDA library ([13]). “m4”, “mc”, and “m8” are real world instances from a finite elements method, see [3], also used in [6, 1, 10]. “cb30” and “cb61” are real world instances from compiler design problems, also used in [6, 10], and finally the “wcb$x$” graphs are the “cb$x$” graphs with vertex weights.

The problems in Table 1 with $p = 2$ and $M = \lceil \frac{N}{p} \rceil$ corresponds to the classical bisection problem with equally sized partitions. For each of the three Multicommodity Flow instances the lower bound and the computation time (format hh:mm:ss) is given. The “Opt” column gives the exact result, or in braces the best known upper bound if the exact solution is unknown. The graph with missing 1-1-MC and VarMC results are not connected, so it is not possible for any vertex to send something to all other vertices. Furthermore, results are presented with $p = 2$, $M = \lceil \frac{2}{3}N \rceil$ and $p = 4$, $M = \lceil \frac{1}{4}N \rceil$.

Conclusions from the experiments

From the experimental results the following observations can be concluded:

- The 1-1-MC instance is generally faster than the two other instances. The VarMC and MVarMC instances are equally fast.
- The 1-1-MC bounds are generally worse than the bounds from the two other instances, but with increasing M the gap of the bounds of the 1-1-MC and the VarMC gets smaller.
- For $p = 2$ the VarMC and MVarMC delivers often the same bound while with increasing p the gap between the VarMC and MVarMC bounds gets bigger.

4 Branch & Bound Algorithm

4.1 Branching realization

In a branch & bound algorithm a given subproblem, which cannot be bounded, has to be divided into at least two new restricted subproblems. We do this restriction by determining if two vertices $v, w \in V$ stay in the

Graph	1-1-MC		VarMC		MVarMC		Opt
	Bound	Time	Bound	Time	Bound	Time	
$p = 2$, $M = \lceil \frac{N}{p} \rceil$							
SE7	14.26	28	15.12	3:47	15.12	3:38	16
DB7	27.52	56	28.98	4:46	28.98	4:22	30
Star 50	12.76	1	25.00	1	25.00	1	25
Grid 5x10	5.00	1	5.00	2	5.00	2	5
ex36a	92.58	15	104.17	54	104.17	45	117
ex36d	1277.49	48	1385.59	1:20	1385.59	1:25	1426
Rand 0.1	15.25	10	30.00	1:19	37.83	40	42
Rand 0.05	-	-	-	-	9.79	12	10
RandPlan	21.37	12	26.79	2:02	26.79	1:43	28
m4	-	-	-	-	5.83	1	6
mc	3.69	3	5.10	17	5.67	17	6
m8	7.00	1:07	7.00	3:42	7.00	3:25	7
cb30	52.23	1	97.50	0	213.00	1	213
cb61	170.50	8	330.00	1:12	1906.40	34	2177
$p = 2$, $M = \lceil \frac{2}{3} N \rceil$							
SE6	7.39	3	7.42	16	7.44	16	8
DB6	13.98	5	13.99	26	13.99	31	14
Star 50	11.10	0	16.00	3	16.00	4	16
Grid 5x10	4.35	1	4.35	5	4.35	6	5
ex36a	82.29	19	86.18	1:02	88.48	1:20	101
ex36d	1135.54	53	1181.32	2:18	1212.00	3:00	1246
Rand 0.1	13.56	10	20.00	2:48	27.30	43	33
Rand 0.05	-	-	-	-	6.00	17	6
RandPlan	18.90	15	20.59	2:11	20.64	1:40	21
m4	-	-	-	-	4.69	1	6
mc	3.23	4	3.38	23	3.45	19	4
m8	6.20	1:09	6.20	4:34	6.20	4:06	7
cb30	46.43	0	65.00	1	114.00	1	114
cb61	150.33	9	220.00	1:17	717.00	46	717
wcb30	38.85	0	53.54	0	126.10	2	136
wcb61	114.31	8	165.92	1:30	826.08	1:02	867
$p = 4$, $M = \lceil \frac{N}{p} \rceil$							
SE6	12.58	3	13.37	14	15.15	13	18
DB6	23.79	5	25.44	22	27.17	21	32
Star 50	19.10	0	37.00	1	37.00	2	37
Grid 5x10	7.49	1	7.49	5	13.95	5	16
ex36a	138.86	18	156.25	59	159.00	35	(186)
ex36d	1916.23	52	2078.38	1:25	2098.00	1:27	(2192)
Rand 0.1	22.88	10	45.00	1:23	58.51	42	(69)
Rand 0.05	-	-	-	-	16.56	14	18
RandPlan	32.05	14	40.20	5:46	53.19	3:37	60
m4	-	-	-	-	11.45	1	12
mc	5.53	4	7.59	18	13.25	36	14
m8	10.50	1:00	10.50	3:11	20.70	10:04	22
cb30	78.00	0	143.00	0	430.20	1	436
cb61	255.20	9	495.00	1:00	4487.88	59	4565

Table 1: Lower bounds and computation time with the three Multicommodity Flow instances

same partition or are separated into different partitions. We call these two possibilities “join” or “split”. A join is performed by creating a new graph from the original graph: the two vertices $v, w \in V$, which have to stay in the same partition, are joined into one vertex $\tilde{v}$ with $g(\tilde{v}) = g(v) + g(w)$. The new edge weights are generated accordingly: $\forall u \in V \setminus \{v, w\} : f(\{u, \tilde{v}\}) = f(\{u, v\}) + f(\{u, w\})$. It is obvious that this new graph has the same $CutSize$ as the original graph with the restriction that v and w have to be in the same partition.

A split of two vertices $v, w \in V$ is performed by removing the commodity with source v and destination w from the general computation of the $CutFlow$. Instead, this particular flow can be counted into the $CutFlow$ completely. Let $S \subset V \times V$ be the set of pairs of vertices which are split. Then we get the following formula for the $CutFlow$ of the MVarMC instance:

$$\begin{aligned} CutFlow \quad \geq \quad & \sum_{\{v,w\} \in S} s(v,w) \cdot g(w) + s(v,w) \cdot g(v) \\ & + \sum_{v \in V} \left(\sum_{w \in V, \{v,w\} \notin S} s(v,w) \cdot g(w) - \bar{s}_v (M - g(v)) \right) + \tilde{s} R (M - R) \end{aligned}$$

with $\bar{s}_v := max_{w \in V, \{v,w\} \notin S} s(v,w)$, $\tilde{s} := min_{v \in V} \frac{\bar{s}_v}{g(v)}$, and $R := N - M \cdot \lfloor \frac{N}{M} \rfloor$. So in fact in case of the MVarMC instance the split means only a reinterpretation of $\bar{s}_v$, while in case of the VarMC and 1-1-MC instance an additional variable $s(v,w)$ is included into the linear program.

4.2 Branching Strategy

A crucial point for a good branch & bound algorithm is the branching strategy: In our application this means the decision, which pair of vertices should be used for the join and split in a given situation. We have tried two different strategies. Firstly, we have used the two vertices that are adjacent to that edge, which corresponds to the constraint of the linear program with the maximal dual value. This idea follows the principle that the constraints with maximal dual value are the ones which restrict the solution mostly. Especially by joining this two vertices, the specific edge disappears from the graph and the bound can increase.

The other strategy follows a simple upper bound for the $CutFlow$ of the 1-1-MC instance. We select these two vertices for the branching which increase the simple upper bound mostly.

Theorem 3
Given a Graph Partitioning problem with graph G, weights g and f, and constants M and p. We use $D(v,w)$ as the length of the shortest path between vertices v and w regarding the edge weights $\frac{1}{f(e)}$. Then the maximal lower bound of the 1-1-MC instance is

$$\frac{N(N-M) + R(M-R)}{\frac{2}{|E|} \sum_{\{v,w\} \in V^2} g(v) \cdot g(w) \cdot D(v,w)}.$$

For our branching strategy we are looking for a pair of vertices which minimizes the term $\frac{2}{|E|} \sum_{\{v,w\} \in V^2} g(v) \cdot g(w) \cdot D(v,w)$ in the case of a join of the two vertices. Following this strategy the new subproblem created by the join should have a good lower bound. But we have not minded on the split subproblem. Neverthelss, experiments have shown that in the case of $p = 2$ this branching strategy yields small search tree. In the case of $p \geq 3$ it is better to restrict the set of candidates to those pairs of vertices which are adjacent. Altogether experiments have shown that the branching strategy based on Theorem 3 yields smaller search trees compared to the strategy basing on the dual values.

4.3 Forcing Moves

Following the subproblems in the search tree, there are some situations where pairs of vertices can be forced to be split or joined. Firstly, if $\exists v \in V : g(v) = M$ this vertex v can be split from all other vertices, since a vertex with weight M corresponds to a partition with maximal size. Secondly, if $\exists v, w \in V, \{v,w\} \notin S : N + g(v) - g(w) - \sum_{u \in V, \{v,u\} \in S} g(u) < N - (p-1)M$ then we can join the vertices v and w, since otherwise vertex v has no possibility to become part of a correct partition. Thirdly, looking at the graph $G_S := (V, S)$: if this graph has two clique subgraphs with p vertices which match in exactly $p-1$ vertices, then the two remaining vertices can be joined. The last possibility for forcing a join bases on a given solution of a Multicommodity Flow instance:

Lemma 2
Given a Graph Partitioning problem with graph G and edge weights f and a Multicommodity Flow problem with this graph, congestions $c : E \to I\!R$, maximal congestion C and CutFlow CF. An improving solution for the Graph Partitioning problem with CutSize at most L is searched. Then the vertices v, w with $\{v, w\} = e \in E$ can be joined if

$$\frac{CF + (C - c(e)) \cdot f(e)}{C} > L.$$

This possibility for forcing a join is extremely helpfully if the bounds are close to a given feasible solution and the graph is somehow irregular such that there are edges which are not loaded with the maximal congestion.

4.4 Experiments

As there are a lot of good heuristics for the Graph Partitioning problem, it is easy to get a good feasible solution at the beginning of the branch & bound procedure. We use the Party library [14] for this purpose. As in most cases the solution from the heuristic is optimal, we do not matter about any best first search but use simple depth first search. The PCx code is used for the solution of the linear programs. We have adapted the code such that the interior point method stops if the primary solution is good enough to bound the actual subproblem or if the dual solution shows that we cannot bound the actual subproblem.

In Table 2 a comparison of our branch & bound algorithm with the results from Karisch et al. [10], for our knowledge the best actual code, are presented. The results in [10] are performed on a HP 9000/735 system. The “#B” column gives the number of search nodes in the branch&bound tree, the time is given in hh:mm:ss. The missing values for some graphs of the 1-1-MC and VarMC instances result from runs we have stoped after a time overflow. For the problems with the DeBruijn graphs we have utilized two specific properties: Firstly, every bisection of the DeBruijn graph has an even CutSize; Karisch has used this for the DB7, also. Secondly, symmetrical parts of the search tree, which result from four automorphisms of the DeBruijn graph, are cut of; this decreases the search tree of the DB8 by a factor of about two. Finally, we want to remark that our approach has solved the bisection problem of the DeBruijn graph with dimension 8 for the first time at all.

Conclusions from the experiments

- The MVarMC instance is definitely the best one for the branch & bound algorithm from the three Multicommodity Flow instances.
- Concerning the tested real world applications our approach delivers equally results compared with the results from Karisch. For the DeBruijn graphs our approach is in orders of magnitude better than Karischs approach. This is the other way around for the randomly generated dense “ex” problems.

So we conclude that our approach is quite good for more “sparse” and “regular” graphs while it is less good for “dense” graphs. In most applications of the Graph Partitioning problem, e.g. circuit layout or load balancing, the graphs are relatively “sparse” and “regular”, in fact. So our approach is well suited for these applications.

5 Theoretical Analyses

5.1 Upper Bounds on the Multicommodity Flow Bounds

In order to support the statement that the VarMC and MVarMC instances deliver better bounds than the known 1-1-MC instance, we have examined the maximal possible bound of graph bisection problems with these instances, if an infeasible partition of the graph with a relative small CutSize exists. It is obvious that the lower bounds basing on Multicommodity Flows have a problem if there exists e.g. a partition into two partitions with sizes $\frac{n}{4}$ and $\frac{3}{4}n$ with a $CutSize$ of one. This single edge forms a bottleneck for the flows an the congestion will be quite large. But we will see that the three different instances can react differently upon such a situation:

Graph	1-1-MC		VarMC		MVarMC		Karisch	
	Time	#B	Time	#B	Time	#B	Time	#B
m4	-	-	-	-	1	1	1	1
ma	-	-	-	-	3	1	3	1
me	1	1	3	1	5	1	4	1
m6	10	13	9	1	21	1	37	1
mb	4	1	10	1	14	1	28	1
mc	2:31	49	12	1	17	1	46	1
md	4	1	14	1	18	1	29	1
mf	5	1	23	1	26	1	24	1
m1	-	-	-	-	2:54	5	18:15	15
m8	1:02	1	2:13	1	2:47	1	5:21	1
cb30	32	82	6	34	0	1	2	1
cb45	6:52	766	8:20	461	4	1	7	1
cb47_99					12	8	10	1
cb47_101					31	9	1:52	35
cb61					2:20	16	20	1
DB5	0	1	0	1	1	1	6	3
DB6	8	11	10	1	17	1	7:49	55
DB7	4:15:14	226	2:40	1	4:17	1	8:53:20	195
DB8					85:15:10	72		
ex36a			1:43:51	245	1:39:36	245	3	1
ex36d			1:35:33	1111	1:43:31	1111	11	3

Table 2: Results of the branch & bound algorithms

Theorem 4
We have a Graph Partitioning problem with graph G, node weights g, edge weights f and parameters $p = 2$, $M = \frac{N}{2}$. Using the 1-1-MC instance for each partition $V = V_1 \cup V_2$ with $N_i = \sum_{v \in V_i} g(v)$ we get a maximal lower bound of

$$Bound \leq \frac{N^2}{4N_1N_2} CutSize(V_1, V_2).$$

In contrast to the simple 1-1-MC instance the VarMC instance can react on a bottleneck such that the flow across these bottleneck is smaller:

Theorem 5
We have a Graph Partitioning problem with graph G, node weights g, edge weights f and parameters $p = 2$, $M = \frac{N}{2}$. Using the VarMC instance for each partition $V = V_1 \cup V_2$ with $N_i = \sum_{v \in V_i} g(v)$ we get a maximal lower bound of

$$Bound \leq \frac{N}{2 \cdot \min\{N_1, N_2\}} CutSize(V_1, V_2).$$

These upper bounds show that the VarMC instance can react better onto the given graph. The difference is the bigger the smaller $\min\{N_1, N_2\}$ is. Examining the MVarMC instance with a graph of a known infeasible partition no upper bound can be shown at all since the MVarMC instance has the freedom to send nothing across a specific infeasible partition.

5.2 Butterfly and Beneš Network

In this section we show that the new VarMC bound and MVarMC bound techniques can also be used for theoretical analyses of lower bounds on the Graph Partitioning problem and can give improved lower bounds compared with the known 1-1-MC. Here we consider the butterfly network without wraparound which has been studied extensively. The butterfly network of dimension d has a simple bisection with $CutSize = 2^d$. In [2] it has been shown that the bisection width is $2(\sqrt{2} - 1)2^d + o(2^d) \approx 0.83 \cdot 2^d$. Here we will show that the known 1-1-MC can prove an asymptotic lower bound of $\frac{1}{2}2^d$ while the VarMC can prove an asymptotic lower bound of $\frac{2}{3}2^d$.

Lemma 3
Solving a Multicommodity Flow instance with a butterfly of dimension d with $(d+1)2^d$ vertices. All vertices of level l sends one commodity to every other vertex of the graph. We use $Load(k,l)$ as the sum of flows over all edges of level k. We define $X_{d,k} := d + k + 2 - 2(k+1)2^{k-d}$ and $Y_{d,k} := 2d - k + 1 - (d-k)2^{-k}$. Then

$$Load(k,l) = 2^{2d} \cdot \begin{cases} X_{d,k} & \text{if } k \geq l \\ Y_{d,k} & \text{else} \end{cases}$$

if the flow is realized using only shortest paths and cannot be improved by any other realization.

Using the above Lemma we can show the following Theorems:

Theorem 6
The 1-1-MC instance delivers a lower bound on the graph bisection problem of a butterfly network with dimension d without wraparound edges of $(\frac{1}{2} + o(1))2^d$.

Theorem 7
The VarMC instance delivers a lower bound on the graph bisection problem of a butterfly network with dimension d without wraparound edges of $(\frac{2}{3} + o(1))2^d$.

The butterfly network is an example of a quite regular network where the VarMC instance gives a better lower bound on the bisection problem with equally sized partitions than the known 1-1-MC. Furthermore, it is remarkable that the above analysis can be adapted to the Beneš network, a kind of back-to-back butterfly. The Beneš network has a similar simple bisection with $CutSize = 2^{d+1}$. To our knowledge the asymptotically exact bisection width is unknown. Using the 1-1-MC we get a lower bound of $\frac{1}{2}2^{d+1}$ while the VarMC instance where only the two most outside levels of vertices and the most inside level of vertices send commodities delivers a lower bound of $\frac{2}{3}2^{d+1}$.

6 Conclusion and further Work

We have introduces a generalized lower bound on the Graph Partitioning problem. The bound bases on Multicommodity Flow instances with arbitrary sizes of commodities for every pair of source and destination. To get correct lower bounds from a flow instance the guaranteed $CutFlow$ is used. By inserting the sizes of the commodities as variables into the linear program, we get the best selection of these sizes for the given graph automatically. We have compared three different types of Multicommodity Flow instances with a different degree of freedom for the sizes of the commodities. Experiments show the superiority of the instance with the biggest degree of freedom.

Basing on these bounds a branch & bound algorithm has been presented which computes exact solutions for the Graph Partitioning problem. The comparison with other approaches shows that for a lot of graphs the presented approach delivers very good results. For example the DeBruijn graph of dimension eight has been solved exactly for the first time. On the other hand there are graphs, for example the quite dense and random instances introduced by Karisch, where former approaches are better suited for. Finally, is has been shown that the generalized bounds can also be used for theoretical analyses of graphs and can deliver new lower bounds. So altogether the new branch & bound algorithm is of importance for applications, since the algorithm can solve problems, which are unsolved until now. And furthermore, the generalized bounds offers new instruments for theoretical analyses.

For a further speed-up of the branch & bound algorithm three improvements look promising: Firstly, we could use the resulting primal and dual solution of a node of the search tree a starting point for the interior point algorithm of the next nodes in the search tree. Secondly, the branch & bound algorithm could be parallelized. Thirdly, we could use specialized algorithms for the Multicommodity Flow problem instead of using the general tool of an interior point algorithm. But our usage of Multicommodity Flow problems does not correspond to known ones since the sizes of the commodities have to be selected. So existing algorithm must be adapted.

Apart of this it is also interesting to use the VarMC and MVarMC instance for theoretical analyses of lower bounds on graphs. For the graph bisection problem the VarMC instance is promising using with quite “regular” graph which are not vertex symmetric. For a partition into more than two partitions the MVarMC approach is promising, even for vertex symmetric graphs.

References

[1] L. Brunetta, M. Conforti, and G. Rinaldi. A branch-and-cut algorithm for the equicut problem. *Mathematical Programming*, 78:243–263, 1997.

[2] C. Bornstein, A. Litman, B. Maggs, R. Sitaraman, and T. Yatzkar. On the Bisection Width and Expansion of Butterfly Networks. In *Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (IPPS/SPDP-98)*, pages 144–150. IEEE Computer Society, 1998.

[3] C. Souza, R. Keunings, L. A. Wolsey, and O. Zone. A new approach to minimising the frontwidth in finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, 111:323–334, 1994.

[4] J. Czyzyk, S. Mehrotra, M. Wagner, and S.J. Wright. PCx user guide (Version 1.1). Technical report, OptimizationTechnology Center, Aragone National Laboratory and Northwestern University, 1996.

[5] J. Czyzyk, S. Mehrotra, M. Wagner, and S.J. Wright. PCx: An interior-point code for linear programming. *Optimization Methods and Software*, 11:397–430, 1999.

[6] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, and L. A. Wolsey. The node capacitated graph partitioning problem: a computational study. *Mathematical Programming*, 81:229–256, 1998.

[7] P.-O. Fjällström. Algorithms for graph partitioning: A survey. Linköping Electronic Articles in Computer and Information Science, 1998.

[8] ILOG. *CPLEX 7.0 Reference Manual*, 2000.

[9] E. Johnson, A. Mehrotra, and G. Nemhauser. Min-cut clustering. *Mathematical Programming*, 62:133–151, 1993.

[10] S. E. Karisch, F. Rendl, and J. Clausen. Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing*, 12(3):177–191, 2000.

[11] L. K. Fleischer. Approximating Fractional Multicommodity Flow Independent of the Number of Commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

[12] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufman, 1992.

[13] K. Mehlhorn and S. Näher. LEDA, a library of efficient data types and algorithms. Technical report, University of Saarland, 1989.

[14] R. Preis and R. Dieckmann. The PARTY Partitioning – Library User Guide – Version 1.1. SFB 376 tr-rsfb-96-024, University of Paderborn, 1996.

[15] R. D. McBride. Progress made in solving the multicommodity flow problem. *SIAM Journal on Optimization*, 8:947–955, 1998.

[16] F. Shahrokhi and L. Szekely. On canonical concurrent flows, crossing number and graph expansion. *Combinatorics, Probability and Computing*, 3:523–543, 1994.

[17] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas. Fast Approximation Algorithms for Multicommodity Flow Problems. *Journal of Computer and System Sciences*, 50(2):228–243, 1995.

Appendix

Proof of Theorem 3

Proof:
The guaranteed $CutFlow$ of the 1-1-MC is $N(N-M)+R(M-R)$, see Lemma 1. So it remains to show that the congestion cannot be smaller than $\frac{2}{|E|}\sum_{\{v,w\}\in V^2} g(v)\cdot g(w)\cdot D(v,w)$. Now we look at the commodity which goes from a source $v\in V$ to a destination $w\in V$. For each edge $e\in E$ which is used by the commodity, a load relative to the edge weight of $\frac{1}{f(e)}$ is generated. So the minimal sum of loads for this commodity is $D(v,w)$. Summing up the loads of all commodities, we get a minimal load of $2\sum_{\{v,w\}\in V^2} g(v)\cdot g(w)\cdot D(v,w)$. In the best case the load is divided equally on all available edges, so the total congestion is at least $\frac{2}{|E|}\sum_{\{v,w\}\in V^2} g(v)\cdot g(w)\cdot D(v,w)$. ∎

Proof of Lemma 2

Proof:
Let us assume that the vertices v,w are split. Then we can adapt the given Multicommodity Flow instance to this split: Additionally to the given commodities we introduce a commodity from source v and destination w. This commodity can have a size of at least $(C-c(e))\cdot f(e)$ using the edge $\{v,w\}$ without increasing the total congestion of the instance. So the total $CutFlow$ is increased by the term $(C-c(e))\cdot f(e)$ and the new lower bound is at least $\frac{CF+(C-c(e))\cdot f(e)}{C}$. Since we are looking for a partition with CutSize at most L, the Lemma is proven. ∎

Proof of Theorem 4

Proof:
From the construction of the 1-1-MC instance it follows that commodities of size at least $2N_1N_2$ have to cross the cut of the two partitions V_1 and V_2. I.e. the congestion of these cut edges is at least $\frac{2N_1N_2}{CutSize(V_1,V_2)}$ and with $Bound=\frac{CutFlow}{Congestion}$ the Theorem follows. ∎

Proof of Theorem 5

Proof:
The minimal size of commodities which crosses the cut of the partitions V_1 and V_2 is created if only the vertices from the larger partition sends something. So the edges from the specific cut have a congestion of at least $\frac{\sum_{v\in V_1} s(v)\min\{N_1,N_2\}}{CutSize(V_1,V_2)}$. Using the formula for the $CutFlow$ of Lemma 1 we get $Bound=\frac{CutFlow}{Congestion}\leq\frac{N}{2\min\{N_1,N_2\}}CutSize(V_1,V_2)$. ∎

Proof of Lemma 3

Proof:
We have $\forall k\in\{0,\dots,d-1\}, l\in\{0,\dots,d\}:\quad Load(k,l):=\sum_{e=\{v,w\}\in E_k}\sum_{u\in V_l} h(u,v,w)+h(u,w,v)$ with E_k are the edges between vertices of level k and level $k+1$ and V_l are the vertices of level l. Let us start with $k<l$. All commodities of a vertex $u\in V_l$ with a destination $v\in V_h, h<k$ have to cross the edges of level k. This are $(k+1)2^d$ commodities of one sender. Furthermore, the commodities with a destination $v\in V_h, h\geq k$ have to cross the edges of level k two times, if the row of the destination vertex differs in a bit b

with $b \geq k$. These are exactly $(d-k)(2^d - 2^{d-k-1})$ vertices. Considering all sending vertices of level l we get

$$\begin{aligned} Load(k,l) &= 2^d((k+1)2^d + 2(d-k)(2^d - 2^{d-k-1})) \\ &= 2^{2d}(2d - k + 1 - (d-k)2^{-k}) \end{aligned}$$

if $l > k$. Now look a the case $k \geq l$: All commodities of a vertex $u \in V_l$ with a destination $v \in V_h, h \geq k$ have to cross the edges of level k. This are $(d-k)2^d$ commodities of one sender. Furthermore, the commodities with a destination $v \in V_h, h < k$ have to cross the edges of level k two times, if the row of the destination vertex differs in a bit b with $b \geq k$. These are exactly $(k+1)(2^d - 2^{d-k})$ vertices. Considering all sending vertices of level l we get

$$\begin{aligned} Load(k,l) &= 2^d((d-k)2^d + 2(k+1)(2^d - 2^{d-k})) \\ &= 2^{2d}(d + k + 2 - 2(k+1)2^{k-d}) \end{aligned}$$

if $l \leq k$. The above considerations are a lower bound on the load of the edges. It is easy to see that these bounds match exactly if all commodities are routed according shortest paths. ∎

Proof of Theorem 6

Proof:
Using Lemma 3 the congestion of the edges of level k is $\frac{1}{2^{d+1}} \sum_{l=0}^{d} Load(k,l)$, which follows from the symmetry of the butterfly. Assuming that the congestion of these edges is maximal we would get a lower bound for the Graph Partitioning problem of $Bound_k = \frac{(d+1)^2 2^d}{(k+1)X_{d,k} + (d-k)Y_{d,k}}$. In order to prove the Theorem, firstly we show that $\forall k \in \{0, \cdots, d-1\} : Bound_k \geq \frac{1}{2}2^d$, and secondly we show that $\exists k \in \{0, \cdots, d-1\} : Bound_k \leq (\frac{1}{2} + o(1))2^d$.

For the first task let us look at the denominator of $Bound_k$:

$$\begin{aligned} & (k+1)X_{d,k} + (d-k)Y_{d,k} \\ = \; & (d+1)^2 + (k+1)^2(1 - 2^{k+1-d}) + (d-k)^2(1 - 2^{-k}) \\ \leq \; & (d+1)^2 + (k+1)^2 + (d-k)^2 \\ \leq \; & 2(d+1)^2 \end{aligned}$$

Using this we get $\forall k : \quad Bound_k \geq \frac{(d+1)^2 2^d}{2(d+1)^2} = \frac{1}{2}2^d$. For the second task we examine $k = \log d$. Again, analysing the denominator we get $(k+1)X_{d,k} + (d-k)Y_{d,k} \geq 2d^2 + o(d^2)$, and the Theorem is proven. ∎

Proof of Theorem 7

Proof:
We can use Lemma 3 again and get a congestion of the edges of level k of $2^{d-1}(X_{d,k} + Y_{d,k})$ if every vertex of level 0 and level d sends a commodity of size 1 to every other vertex. According to the proof of Theorem 6 we get $Bound_k = \frac{2(d+1)}{X_{d,k}+Y_{d,k}}2^d$. Looking again at the denominator we get $X_{d,k} + Y_{d,k} \leq 3(d+1)$. So $\forall k : \; Bound_k \geq \frac{2}{3}2^d$. Using now $k = \frac{d}{2}$ we get $X_{d,k}+Y_{d,k} \geq 3(d+1) - o(d)$, so $Bound_{\frac{d}{2}} \leq (\frac{2}{3} + o(1))2^d$. ∎