New heuristics for the minimum fundamental cut basis problem

A. J. Perez Tchernov[*], A. M. Schwahn[†]

July 24, 2007

Abstract

Given an undirected connected network $G = (V, E)$, $n = |V|$, $m = |E|$, and a weight function $w : E \mapsto \mathbb{R}_+$, finding a basis of the cut space with minimum sum of the cut weights is termed *Minimum Cut Basis Problem.* This problem can be solved, e.g., by the algorithm of Gomory and Hu [GH61]. If, however, *fundamentality* is required, i.e., the basis is induced by a spanning tree T in G, the problem becomes NP-hard. Theoretical and numerical results on that topic can be found in Bunke et al. [BHMM07] and in Bunke [Bun06]. In the following we present heuristics with complexity $O(m \log n)$ and $O(mn)$ which obtain upper bounds on the aforementioned problem and in several cases outperform the heuristics of Schwahn [Sch05].

Keywords: minimum fundamental cut basis, fundamental cut, cut, heuristic, algorithm, data structure, NP

1 Introduction

In this article we consider a finite undirected connected simple graph $G = (V, E)$, where V and E are the sets of vertices and of edges, respectively, $|V| = n$ and $|E| = m$. A *cut* of a graph is a set of edges C such that its deletion, $G - C$, disconnects the graph. The set of all cuts of a graph and the operation of symmetric difference constitute a vector space on the field F_2. A basis $B = \{C_1, \ldots, C_n\}$ of a cut space is called *fundamental* if each cut C_i contains at least one edge that does not belong to any other cut C_j, where $j \neq i$.

Let a spanning tree $T = (VT, ET)$ of a graph G be given and let $e \in ET$ be fixed. The deletion of e from T induces two connected components T_1^e and T_2^e; the edges of the graph having their endpoints in different components constitute a *fundamental cut* denoted by δ_T^e. It follows from definition that any fundamental basis of a cut space is determined by the fundamental cuts δ_T^e, where T is a spanning tree of G and $e \in ET$.

[*]Belarusian State University; E-mail: alex.pereztchernov@gmail.com

[†]Technische Universität Kaiserslautern, Germany; E-mail: schwahn@mathematik.uni-kl.de

Given a *weight function* $\omega : E \mapsto \mathbb{R}_+$, the weight of a subgraph (set of edges) A is defined as $\omega(A) = \sum_{e \in A} \omega(e)$, and the weight of a fundamental cut basis as $\omega(\delta T)$, where some edges are summed up more than once.

An *edge swap* $\pi = (e, f)$ of edges e and f is an operation constructing a new spanning tree πT from a spanning tree T by deleting edge $e \in ET$ and adding edge $f \in \delta_T^e$.

The problem of finding a fundamental cut basis with minimum weight (MinFDB) is NP-hard [BHMM07]. Optimal solutions can not be found, but for the smallest instances. Therefore, we need heuristics to find a near-optimal solution in reasonable time.

Dealing with a "tree based" problem we can work within the following frame for heuristics:

1) *Starting point phase:* Choose the vertex $a \in V$ to start with; let $T_0 = a$ initially.

2) *Strategy phase:* Expand the trivial one-vertex tree T_0 to a spanning tree T_n edge by edge using a special strategy to examine the edge to be added to the tree T_k to construct $T_{k+1} = T_k + e$.

3) *Swapping phase:* Swap edges to traverse different spanning trees until a *local minimum* is found, i.e., a spanning tree where no better tree can be reached by k edge swaps, where $k > 0$ and k is usually small (< 3) when dealing with large graphs due to practical reasons.

4) *Metaheuristic phase:* Try to escape a local optimum since it is not necessarily a global one. Execute a certain number of random edge swaps and apply the local search to the modified spanning tree. If the result is better than the incumbent, the former optimum is replaced by the new one and the procedure is iterated.

The above heuristic is meant to position the following results within the common outline. Our heuristics, named *P-min* and *P-max*, concern the strategy phase of the above frame, they are introduced in section 2. Besides, in section 3 we propose special data structures used in the implementation of P-min and P-max in order to calculate $\omega(\delta T)$ efficiently. They could also be useful in the swapping phase, possibly expediting local search and metaheuristics like the Variable Neighbourhood Search (VNS). Observations on the effect of an edge swap on fundamental cuts/cycles can be found in Amaldi et al. [ALMM04] where VNS is also applied to the Minimum Fundamental Cycle Basis Problem. Schwahn investigated the use of several heuristics, local search, and VNS for the Cut Problem [Sch05]. In section 4, her results are compared to the results of P-min and P-max where the ladder ones turn out to be advantageous for a large number of graphs.

Even though the MinFDB has not yet found its real world application, numerous fields of application for the related Minimum Fundamental Cycle Basis Problem suggest a practical use of our problem as well.

2 Strategies

Let a spanning tree T be fixed, it induces a fundamental cut basis with weight

$$\omega(\delta T) = \sum_{e \in ET} \sum_{f \in \delta_T^e} \omega(f). \tag{1}$$

This equation can be rewritten as

$$\omega(\delta T) = \sum_{e \in E} \omega(e) + \sum_{e \in E \setminus ET} (d_T^e - 1)\, \omega(e) \tag{2}$$

where d_T^e is the number of edges on the path in T connecting the endpoints of e. This reformulation can also be deduced from results concerning the strong relationship between fundamental cuts and fundamental cycles [SR61].

The first summand of Equation (2) is $\omega(E)$, thus it follows that $\omega(\delta T) \geq \omega(E)$ and since $\omega(E)$ is constant, we only need to minimize the function

$$\sum_{e \in E \setminus ET} (d_T^e - 1)\, \omega(e). \tag{3}$$

Since each summand of the above formula is a product, our heuristics concentrate on its factors, namely, distance and weight.

2.1 P-min heuristic

Assume a tree T_k, which is not yet spanning, has been constructed. We have to choose the next edge to be added to T_k.

Solve the following optimization problem:

$$\min_{\substack{e \in E \\ |e \cap T_k| = 1}} \quad \varphi_{T_k}(e) := \sum_{\substack{f \in E \setminus \{ET_k \cup e\} \\ |f \cap e| = 1}} d^f_{T_k \cup e}. \tag{4}$$

Here $|e \cap T_k| = 1$ and $|f \cap e| = 1$ mean that edge e and the connected subgraph T_k have exactly one vertex in common, and e and f also possess one common vertex. Note that $d^f_{T_k \cup e}$ is only defined for edges f having both endvertices in the union of the current tree and e, if this is not the case, $d^f_{T_{k+1}}$ is set to 0. The value $\varphi_{T_k(e)}$ is the sum of the length of fundamental cycles edge e induces if added to T_k minus 1 each. An edge solving (4) is added to T_k.

This greedy strategy affects the first multiplier of the sum in equation (3) in order to make it smallest possible, thus, we name it *P-min heuristic*. The tree obtained this way is locally branched and can in certain cases approach a star tree.

2.2 P-max heuristic

In this different approach we consider the second multiplier of (3). It seems reasonable to have edges with heavy weight within the tree or at least near the tree, where "near" refers to the distance d_T^e. Therefore we solve the following optimization problem:

$$\max_{\substack{u \notin VT_k \\ \exists v' \in VT_k \,:\, (u,v') \in E}} \vartheta(u) := \sum_{f=(u,v)\in E} \omega(f). \tag{5}$$

We name this procedure *P-max heuristic.* For each vertex we sum up the weights of all incident edges to obtain the vertex weight. Starting from the median we always connect the "heaviest" of vertices adjacent to the current tree. That means if we have heavy edges, their weight add to the vertex weight of both endpoints. Thus, they are connected to the tree and therewith to the median as soon as possible. Consequently, the distance between the endpoints is kept short or the heavy edge is even made part of the tree.

2.3 Short Tree and Heavy Tree

The pre-existing heuristics of Schwahn [Sch05] base on the same idea - the multipliers of equation (3). We shortly summarize the methods and differentiate them from the ones introduced above.

The *Short Tree* heuristic calculates the median of a graph and constructs a shortest path spanning tree out of it. Therefore the algorithm of Floyd and Warshall [Flo62] is used giving this heuristic a complexity of $O(n^3)$. The P-min heuristic, however, foregoes these global calculations. It does not reduce the worst case complexity (for $m = O(n^2)$) but performs better in practice.

The *Heavy Tree* heuristic uses the algorithm of Kruskal [Kru56] to put the heavy edges into a tree, this can be done in $O(m \log n)$. The P-max heuristic goes beyond and considers also the position of the heavy edges which are not part of the tree, affecting more multipliers in equation (2).

3 Realization

In this section we elaborate on the implementation of P-min and P-max and introduce a data structure easing the calculation of the distances.

3.1 P-max heuristic

The maximization problem (5) is easy to handle. While the set of all vertices which are adjacent to one vertex of the current tree T_k changes in each iteration, the value ϑ_{T_k} does not depend on the current tree and thus does not have to be updated in the algorithm. We use ϑ as a weight function on the vertices and run a modification of a minimum spanning tree algorithm. Realizing the priority queue by *d-heaps* [RLC90] or *Fibonacci-heaps* [FT87], the complexity of P-max is $O(m \log n)$.

3.2 P-min heuristic

The P-min heuristic poses further problems. Not only the set of edges to be minimized upon changes while T_k grows, but the objective function φ_{T_k} also has to be updated when an edge has been added to T_k. Furthermore the distances between vertices in the growing tree, the atomic summands, have to be determined efficiently.

It is reasonable to update φ only for those edges which are actually influenced by the choice of the new edge. This is done in the following algorithm:

Algorithm 3.1. *Given a tree T_k, $S(T_k)$ denotes the set of edges with exactly one vertex in the current tree, e.g. $S(T_k) = \{e = (a, b) \in E :\ a \in VT_k,\ b \notin VT_k\}$, the values $\varphi_{T_k}(e)$ are known for all $e \in S(T_k)$. Then*

- *choose edge $e^* = (a^*, b^*)$ solving* $\min_{e \in S(T_k)} \varphi_{T_k}(e)$
- *generate the list $Q = N(b^*) \setminus VT_k$, where $N(b^*)$ is the set of vertices adjacent to b^**
- *φ is updated for every edge (v, u) such that $v \in Q$ and $u \in VT_k$:*
  $\varphi_{T_{k+1}}(v, u) := \varphi_{T_k}(v, u) + d_{T_k}^{ua^*} + 2$
- *calculate $\varphi_{T_{k+1}}(v, b^*)$ for every $v \in Q$ which is adjacent to a vertex in VT_k*
- *create $S(T_{k+1})$ from $S(T_k)$ by deleting edge e^* and all edges(b^*, u) where $u \in V_{T_k}$ and adding all edges (b^*, v) where $v \in Q$*

The following straight forward data structure is used to ease the computation of φ for all edges of a graph. Given a tree T and a fixed vertex v_0 in T, the distance function $d_T(\cdot, v_0)$ induces a partial ordering on T, namely, vertex a *precedes* vertex b if $d_T(a, v_0) \leq d_T(b, v_0)$ and a lies on the path from b to v_0 in T. We create a directed tree $T' = (VT, ET')$ rooted at v_0. For any vertex $v_i \in VT$ let the label $m_{T'}(v_i)$ be equal to the distance $d_T(v_0, v_i)$. A directed edge (u, v) is in ET' if and only if it is in ET and $m_{T'}(u) \geq m_{T'}(v)$, i.e. v precedes u. Thus, all edges are directed towards the root v_0.

It is obvious that

$$d_T^{uv} = m_{T'}(u) + m_{T'}(v) - 2\, m_{T'}(w), \tag{6}$$

where w is the nearest common ancestor of vertices u and v in the directed tree T'.

Thus, in order to efficiently compute the distance between two vertices in T, we only need to find their nearest common ancestor in T'. Therefore, we use the following results.

Let T_k be a tree with $x \in T_k$, $y \notin T_k$ subjected to the following operations:

- *add_leaf* (x, y): add the edge directed from x to y to T_k
- *add_root* (y): add the edge directed from y to the current root of T_k to T_k, thus, y is the new root
- *nca* (x, y): return the nearest common ancestor of x and y

Theorem 3.2. *[Gab90] A sequence of m* nca *operations and n* add_leaf *and* add_root *operations can be processed in time $O(m + n)$ and space $O(n)$.*

Using the above techniques and a special data structure [PTS05] it can be shown that the P-min heuristic for the MinFDB runs in $O(mn)$ time.

Modifications of the tree, i.e., in order to perform edge swaps, can be done efficiently using a special data structure of Sleator and Tarjan [ST83].

4 Computational results

In order to compare the heuristics we used different ways to obtain random graphs having 20 to 100 vertices, a procedure proposed by Viger and Latapy [VL05], [Vig] for a prescribed degree sequence, a standard method generating random graphs and the graphs from the SIVALAB [Siv03] collection. All the graphs have been complemented with uniformly distributed edge weights.

For each single class of connected graphs, characterized by its numbers of vertices and edges and by its weight distribution, we created 100 to 1000 graph-representatives. The overall number of classes is almost 400.

We compared the four algorithms with each other. For each class of graphs and parameter combinations we counted how often an algorithm outputted the best result (the lowest total cut weight) of the investigated heuristics. In many cases more than one algorithm came up with the best result, then we assigned the "hit" to all of them, therefore the percentages may add up to more than one. We also calculated the duality gap for the heuristics, i.e., we divided their results by a lower bound. Therefore, we used the algorithm of Gomory and Hu [GH61] to obtain a minimum cut basis.

A selection of the results of our numerical comparison is shown in the following table. We varied the number of edges in tables 1 and 2 and the weight in table 3. The tables provided below only display the results of the randomly generated graphs, this method ensured a fixed numbers of vertices and edges for a large number of graphs.

We observed that a variation of the weight span does not affect the quality of the heuristics significantly whereas the density of the graph has a larger influence on the duality gap. The denser a graph the better perform all algorithms but the Heavy Tree heuristic.

Beyond the proposed heuristics we analyzed some modifications as well, e.g., the weighted version of equation (4), but came across that they deliver worse results. The presented heuristics are simple as well as appropriately connected to the objective function of MinFDB.

P-min and P-max use the same starting point, namely the median which is used in the Short Tree heuristic by default. Further research can explore the advantageousness of different starting points for the new heuristics, e.g., center instead of median can be used.

	Number of best hits				Duality Gap			
m	P-min	P-max	Short	Heavy	P-min	P-max	Short	Heavy
100	0.35	0.47	0.20	0.00	1.186	1.182	1.191	1.962
110	0.37	0.46	0.22	0.00	1.160	1.159	1.163	1.981
120	0.35	0.44	0.25	0.00	1.135	1.131	1.135	2.000
130	0.34	0.51	0.26	0.00	1.107	1.102	1.106	2.026
140	0.35	0.53	0.28	0.00	1.083	1.078	1.081	2.038
150	0.41	0.57	0.37	0.00	1.059	1.054	1.056	2.056
160	0.64	0.73	0.68	0.00	1.034	1.032	1.032	2.081
170	0.97	0.98	0.97	0.00	1.011	1.011	1.011	2.085
180	1.00	1.00	1.00	0.00	1.011	1.011	1.011	2.117

Table 1: Influence of the number of edges ($n = 20$, w in [1,100])

	Number of best hits				Duality Gap			
m	P-min	P-max	Short	Heavy	P-min	P-max	Short	Heavy
100	0.42	0.39	0.24	0.00	1.173	1.176	1.175	1.681
110	0.37	0.41	0.29	0.00	1.151	1.150	1.151	1.700
120	0.40	0.42	0.29	0.00	1.127	1.126	1.127	1.721
130	0.41	0.46	0.30	0.00	1.100	1.097	1.098	1.730
140	0.39	0.48	0.34	0.00	1.076	1.073	1.073	1.762
150	0.47	0.52	0.49	0.00	1.048	1.046	1.046	1.767
160	0.69	0.74	0.70	0.00	1.024	1.023	1.023	1.769
170	0.96	0.96	0.97	0.00	1.003	1.003	1.003	1.803
180	1.00	1.00	1.00	0.00	1.000	1.000	1.000	1.831

Table 2: Influence of the number of edges ($n = 20$, w in [990,1000])

5 Acknowledgements

The research has been partially supported by research grants from INTAS (INTAS-03-50-5975) and by Rheinland-Pfalz cluster of excellence “Dependable adaptive systems and mathematical modeling” (DASMOD).

References

[ALMM04] E. Amaldi, L. Liberti, N. Maculan, and F. Maffioli. Efficient edge-swapping heuristics for finding minimum fundamental cycle bases. In *WEA '04: 3rd international workshop on experimental and efficient algorithms*, Angra dos Reis, Brazil, 2004.

	Number of best hits				Duality Gap			
w	P-min	P-max	Short	Heavy	P-min	P-max	Short	Heavy
$[10, 100]$	0.34	0.44	0.24	0.00	1.183	1.181	1.185	1.932
$[30, 100]$	0.37	0.40	0.25	0.00	1.178	1.177	1.178	1.875
$[50, 100]$	0.38	0.42	0.23	0.00	1.177	1.176	1.179	1.806
$[70, 100]$	0.39	0.42	0.22	0.00	1.180	1.180	1.181	1.756
$[90, 100]$	0.41	0.37	0.26	0.00	1.178	1.178	1.179	1.688
$[99, 100]$	0.49	0.59	0.37	0.59	1.178	1.175	1.179	1.175
$[1, 10]$	0.39	0.43	0.25	0.00	1.180	1.178	1.184	1.819
$[1, 100]$	0.32	0.46	0.26	0.00	1.187	1.181	1.186	1.971
$[1, 1000]$	0.34	0.44	0.24	0.00	1.185	1.178	1.184	1.972

Table 3: Influence of the weight ($n = 20$, $m = 100$)

[BHMM07] F. Bunke, H. W. Hamacher, F. Maffioli, and Schwahn A. M. Minimum cut bases in undirected networks. Report in Wirtschaftsmathematik 56, Fachbereich Mathematik, TU Kaiserslautern, 2007. 108.

[Bun06] F. Bunke. *Circuit bases problems in binary matroids.* PhD thesis, Department of Mathematics, University of Kaiserslautern, 2006.

[Flo62] R. W. Floyd. Algorithm 97, shortest path. *Communications ACM*, 5:345, 1962. (INF Z 1415).

[FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[Gab90] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[GH61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 9(4):551–570, 1961.

[Kru56] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the AMS*, 7:48–50, 1956. (MAT Z 1046).

[PTS05] A. J. Perez Tchernov and S. V. Suzdal. Special data structure for graph problems connected with the notion of clique or modular decomposition. Abstract of the ECCO XVIII, United Institute of Informatics Problems of the National Academy of Sciences, 2005.

[RLC90] R. L. Rivest, C. E. Leiserson, and T. H. Cormen, editors. *Introduction to Algorithms.* McGraw-Hill Companies, 1990.

[Sch05] A. M. Schwahn. Minimum fundamental cut basis problem. Diploma Thesis, Technische Universität Kaiserslautern, 2005.

[Siv03] *The graph database CD.* Universita di Napoli, Federico II, Italy, 2003. http://amalfi.dis.unina.it/graph/db/doc/graphdbat.html.

[SR61] S. Seshu and M. B. Reed. *Linear Graphs and Electrical Networks.* Addison-Wesley, Massachusetts, 1961. (EIT 608/008).

[ST83] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[Vig] F. Viger. Graph generator. http://www.liafa.jussieu.fr/ fabien/generation/.

[VL05] F. Viger and M. Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In *COCOON: Computing and Combinatorics, 11th Annual International Conference*, volume 3595 of *LNCS*, pages 440–449. Springer, 2005.