Open Access Library Journal
2025, Volume 12, e13079
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Another Look at Node Renumbering

Ibrahim F. Khatib

(Retired) Khatib & Alami CEC, Beirut, Lebanon
Email: ibrahimfkhatib84@gmail.com

How to cite this paper: Khatib, I.F. (2025) Another Look at Node Renumbering. *Open Access Library Journal*, **12**: e13079.
https://doi.org/10.4236/oalib.1113079

Received: February 13, 2025
Accepted: March 22, 2025
Published: March 25, 2025

Abstract

Node renumbering is an important step in the solution of sparse systems of equations. It aims to reduce the bandwidth and profile of the matrix. This allows for the speeding up of the solution of the system or allows for the addressing of even larger systems than otherwise would be possible. Research on this topic dates to the late sixties. In most of these algorithms nodal degree is an important consideration. In the new algorithm, we consider the maximum difference in adjacent node numbers as the driving criterion. This new algorithm (IFK), when compared against well-known algorithms such as Reverse-Cuthill-McKee (RCM), Gibbs (GBS), Gibbs-Poole-Stockmeyer (GPS) and Sloan's (SLN) over 101 test cases from the Boeing-Harwell sparse matrix cases, achieves significant performance improvement in profile and bandwidth reduction and execution time.

Subject Areas

Computer Engineering, Software Engineering

Keywords

Node Renumbering, Matrix Profile, Matrix Bandwidth, Symmetric Matrix

1. Introduction

Node renumbering is an important step in the solution of sparse systems of equations. It aims to reduce the bandwidth and profile of the matrix. This allows for the speeding up of the solution of the system or allows for the addressing of even larger systems than otherwise would be possible. Research on this topic dates to the late sixties. In most of these algorithms nodal degree is an important consideration (c.f. TERMINOLOGY section below).

However, the earliest algorithms looked at the maximum difference in adjacent node numbers and attempted an iterative approach combined with some logic to propose a new ordering [1] [2]. These algorithms were limited in the size of ma-

trices they could handle. They were superseded by so-called direct algorithms. Most of these direct algorithms relied on building a rooted level structure (RLS) based on some start node and proceeded with the numbering nodes successively down the levels of the RLS. As such, these algorithms comprise two main phases. The first phase consists of searching for the deepest (and narrowest) RLS, the motivation being theoretical results [3] that relate the actual bandwidth to the largest level width in the RLS. The second phase consists of actual renumbering of nodes, generally proceeding level by level down the RLS.

The algorithms differ from one another in how they deal with each of the two phases and how much effort is spent on obtaining a deep and narrow starting RLS and what tricks are used to limit the bandwidth to a value below the theoretical upper bound.

The new algorithm is described in section 2, the test methodology is outlined in section 3. The test results are reviewed in section 4, profile and bandwidth reduction in section 4.1, execution time in section 4.2 and the resulting matrix profiles in section 4.3. Conclusion in section 5 summarizes the findings. Appendix 1 contains the detailed numerical result tables. Appendix 2 shows the profile of some matrices on which all algorithms failed to improve the profile or bandwidth. Appendix 3 contains a result comparison of classical algorithms as programmed in this work against published results.

2. Algorithm

The proposed algorithm focuses on the maximum difference in adjacent node numbers as the driving criterion. In this focus it is like the earlier iterative algorithms, but it differs in that it adopts an RLS approach to proceed with the renumbering. The use of an RLS allows numbering of the nodes in a logical sequence based on their connectivity. The algorithm starts with the adjacency list for each node in the matrix and proceeds as follows:

1-For each node I collect the maximum difference in node numbers in its adjacency list ND_I = max(abs($N_I - N_K$) for N_K label of node K in adj(I)).

2-For each node N_I calculate the average maximum difference AD_I = sum(ND_k for N_k in adj(I))/DEG_I where DEG_I is the degree of node I (or length of adjacency list of I adj(I)).

3-Pick node S with highest AD_S not already stored in the set of starting nodes already visited and build a rooted level structure RLS rooted at S. Record S in a set of starting nodes. Label S as the first node.

4-Looping over the levels of the RLS, order the nodes of each level in order of decreasing ADK and then label them sequentially.

5-Calculate ND_I based on the new labeling. Calculate the new bandwidth (max of ND_I) and the new profile (sum of ND_I).

6-If the new bandwidth is less than the current "best bandwidth", set the "best bandwidth" equal to the new bandwidth and the "best ordering" equal to the new ordering. (The initial "best ordering" is the initial ordering, and the "best band-

width" is based on the initial ordering). If the difference between the new "best bandwidth" and the previous "best bandwidth" is less than a certain percentage of the "best bandwidth" set the "done" flag to true.

7-If the number of iterations exceeds a preset value, set the "done" flag to true. If the "done" flag is false go back to step 2, otherwise terminate and return the "best ordering" calculated so far.

8-Reverse "best ordering". This improves on the profile for the same best bandwidth.

The maximum difference in node number ND is the goal that needs to be reduced, and therefore it makes sense to look at it directly. The summation of NDs over a node's adjacency list allows for the prioritization of the nodes that have large NDs that need to be reduced first. Here, one can have two choices: 1) either look at the sum of NDs over a node's adjacency list or 2) divide this sum by the nodal degree (as is done in step 2 of the algorithm above). The first choice can find solutions in cases where the second choice doesn't, particularly for large matrix dimensions. However, testing indicates that the second choice is generally more effective than the first in finding optimal numbering, it is the one finally retained.

One variant considered is to adopt in step 4 Gibb's approach of labeling nodes in groups according to their adjacency to nodes in the previous level. This should ensure that the bandwidth obtained is significantly smaller than the width of the level. After some tests this option achieved only minor improvements in profile but a slight increase in bandwidth at the cost of increased computation time (x15 on average). It is finally discarded.

3. Test Methodology

To test the new algorithm, it was programmed in C++ under Microsoft Visual Studio along with the RCM, GBS, GPS and SLN algorithms. As there are many variants and interpretations of these algorithms, below is a brief description of the implementation used for the comparison:

3.1. Reverse Cuthill-McKee (RCM)

The original algorithm [4] looped over the nodes of minimum degree to generate an RLS, label the nodes and retain the best label (minimum bandwidth and minimum profile after inverting the number as per George and Liu recommendations [5] [6]). As this approach is too time consuming for large matrices, Gibbs' approach [7] is adopted to find a starting RLS of maximum length and minimum width.

We loop over the set of nodes of minimum degree to obtain an RLS of maximum length and minimum width. Then, from the nodes in the last level of this RLS, we repeat the process to obtain another RLS with maximum length and minimum width. We retain the tallest (and narrowest in case of equal length) RLS as the starting RLS.

Looping over the levels of the RLS, we order the nodes of each level by increasing degree and label them sequentially. Finally, we reverse the labelling according to George and Liu's recommendations.

3.2. Gibbs (GBS)

For the first phase, we loop over the set of nodes of minimum degree to obtain and RLS of maximum length and minimum width. Then, from the nodes in the last level of this RLS, we repeat the process to obtain another RLS with maximum length and minimum width. We retain the tallest RLS as the starting RLS.

For the second phase, as we loop over the levels, we obtain the intersection of its adjacency list for each node at a level with the next level and label these nodes sequentially. The details are explained in Gibbs' [8].

3.3. Gibbs-Poole-Stockmeyer (GPS)

For the first phase, we loop over the set of nodes of minimum degree to obtain and RLS of maximum length and minimum width. Then, from the nodes in the last level of this RLS, we repeat the process to obtain another RLS with maximum length and minimum width. Then, we generate a pair of indices for each node based on their distance from each of the roots of the two RLS. Thus, nodes with identical numbers in the pair belong to a new RLS (the "intersection of the two RLSs"). Then, an elaborate procedure is applied to assign the remaining nodes to the levels of this last RLS.

For the second phase, as we loop over the levels, we obtain for each node in a level the intersection of its adjacency list with the next level and label these nodes sequentially. The details are explained in Gibbs-Poole-Stockmeyer [7].

3.4. Sloan (SLN)

For the first phase, we loop over the set of nodes of minimum degree to obtain and RLS of maximum length and minimum width. Then, from the nodes in the last level of this RLS, we repeat the process to obtain another RLS with maximum length and minimum width. The RLS with the narrowest width is considered the starting RLS, and its root node is the start node. The other RLS is considered as the ending RLS, and its root node is the end node.

For the second phase, nodes are assigned to categories such as inactive, pre-active, active, and inactive. A priority is calculated for each node based on its distance from the end node and its "effective degree". Nodes are labeled according to their category and priority. A protocol is applied to transfer a node from one category to another. Details are in Sloan's paper [9]. The weights used for distance from end node and effective degree are 1 and 2, respectively.

The test cases are obtained from Boeing Harwell's suite of sparse matrix files (http://math.nist.gov/MatrixMarket/data/Harwell-Boeing). Initially, 120 test cases were considered (BCSPWR01 to BCSPWR10, BCSSTK01 to BCSSTK33, CAN series, CEGB series, DWT series, JAGMESH1 to JAGMESH9, LOCK series, LSHP series, NOS1 to NOS7, BLCKHOLE and SSTMODEL). It was found that for some test cases (19), all five algorithms (RCM, IFK, GBS, GPS, SLN) failed to obtain a better ordering (smaller bandwidth and smaller profile). Upon closer look at the matrix profile, it is found that these are already ordered in an optimal manner as

can be seen from the figures below (APPENDIX 2).

The results of the RCM, GBS, GPS and SLN algorithms as programmed in C++ were compared to published results [10]-[12]. In some cases, the bandwidths or profiles obtained are higher than the published results and, in some cases, they are slightly lower (APPENDIX 3). One can conclude that the algorithms as programmed are valid for comparison purposes.

4. Test Results

The properties of the retained 101 cases are detailed in Appendix 1 Table A1. The matrix dimensions range from 24 to 44,609 and the number of non-zeros range from 167 to 2,072,376. The sparsity (ratio of non-zeros to square of dimension) ranges from 0.1% to 30.4%. The initial normalized profile (2 x profile/square of dimension) ranges from 0.7% to 97.1%. The initial normalized bandwidth (bandwidth/(dimension − 1)) ranges from 0.4% to 100%.

In assessing the performance of the new algorithm, we measure the percent reduction in profile and the percent reduction in bandwidth against the initial profile and bandwidth as well as against the other algorithms. The duration of reordering is monitored as well.

4.1. Bandwidth and Profile Reduction

Over the 101 retained cases there are cases where one or more algorithms fail to improve on the original ordering while others do obtain an improvement. There are 4 cases (DWT1007, DWT361, DWT869, LOCK1074) where IFK is the only one that fails at improving on the original ordering.

On the other hand, there are 8 cases (BCSSTK11, CAN445, DWT87, DWT193, DWT198, JAGMESH5, JAGMESH8, LOCK2232) where the IFK algorithm achieves the maximum improvement in bandwidth and/or profile reduction. In addition, there are 59 cases where the new algorithm achieves an improvement in bandwidth and profile over all the other algorithms.

- The average profile reduction of the new algorithm is 31%, compared to 29% for RCM, 31% for GBS, 26% for GPS and 36% for SLN.
- The maximum profile reduction of the new algorithm is 90%, compared to 89% for RCM, 90% for GBS, 86% for GPS and 94% for SLN.
- The average bandwidth reduction of the new algorithm is 47%, compared to 46% for RCM, 47% for GBS, 34% for GPS and 42% for SLN.
- The maximum bandwidth reduction of the new algorithm is 97%, compared to 97% for RCM, 98% for GBS, 97% for GPS and 97% for SLN.
- The new algorithm achieves a reduction in profile in 84 of the 101 cases, compared to 78 for RCM, 81 for GBS, 74 for GPS and 83 for SLN.
- The new algorithm achieves a bandwidth reduction in 78 of the 101 cases, compared to 75 for RCM, 79 for GBS, 74 for GPS and 78 for SLN.

The detailed numbers are collected in APPENDIX 1 Table A2 and Table A3. The new algorithm's relative improvement in profile versus the other algorithms

can best be described by Figure 1 below, which plots the probability density of the extra improvement in profile of IFK versus each of the other algorithms. The new algorithm tends to achieve better profile reduction than GPS.

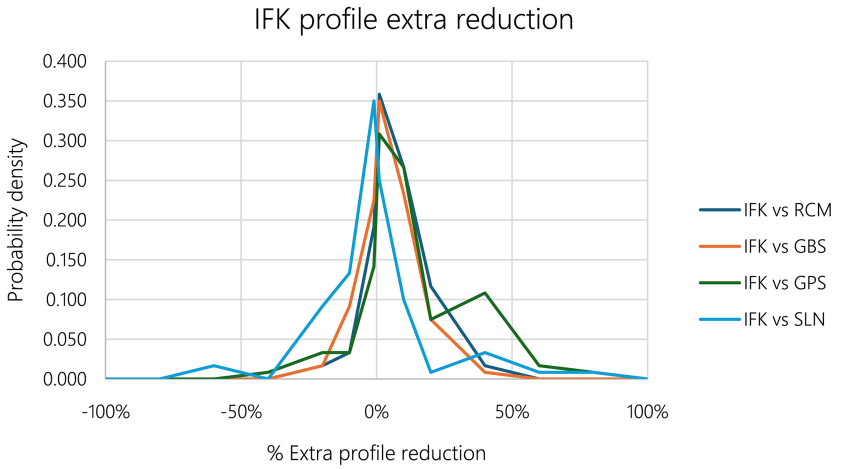


Figure 1. IFK profile extra reduction distribution.

The new algorithm is more effective in improving bandwidth than profile as is shown in Figure 2 below, which plots the probability density of the extra improvement in bandwidth of IFK versus each of the other algorithms. The new algorithm's bandwidth is likely to be better than that computed by GPS, RCM or SLN.

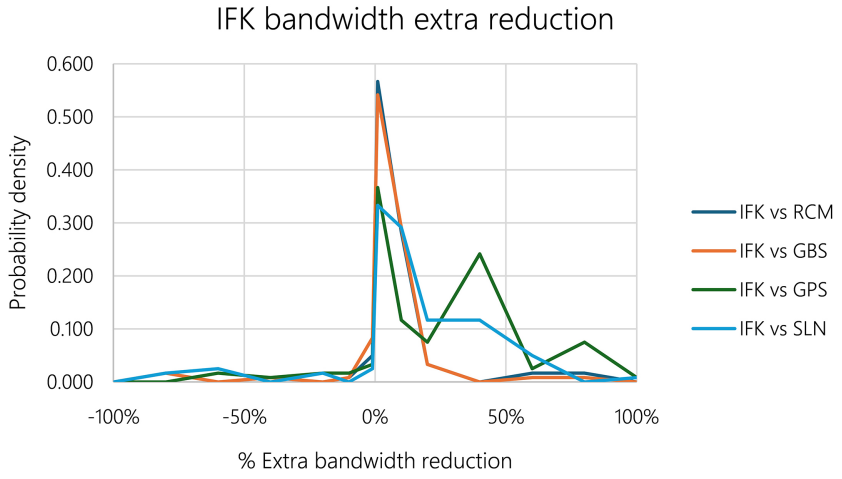


Figure 2. IFK bandwidth extra reduction distribution.

Figure 3 below captures the scatter of the new algorithm's improvement in profile and bandwidth over the other algorithms. Again, the improvements in bandwidth are more likely to be positive than those in profile which are equally scattered in the positive and negative range.

For bandwidth:

- Compared to RCM, it achieves a maximum improvement of 74%.

- Compared to GBS, it achieves a maximum improvement of 71%.
- Compared to GPS, it achieves a maximum improvement of 93%.
- Compared to SLN, it achieves a maximum improvement of 83%.

For profile:

- Compared to RCM, it achieves a maximum improvement of 31%.
- Compared to GBS, it achieves a maximum improvement of 31%.
- Compared to GPS, it achieves a maximum improvement of 65%.
- Compared to SLN, it achieves a maximum improvement of 68%.

Table A3 in Appendix 1 below shows the relative performance of IFK versus the other reference algorithms.

We attempted plotting the percent reduction in profile of all algorithms against either matrix dimension, number of zeros, sparsity or initial normalized profile. No clear trend emerged with any of these quantities. Hence, we retained the plot below as it most clearly shows the scatter in profile reduction for all algorithms. All algorithms show considerable performance scatter across the test cases used. The SLN algorithm tends to lead in profile reduction.

Similarly, the figure below shows the scatter of percent reduction in bandwidth plotted against the normalized initial bandwidth.

Again, all algorithms show considerable scatter across the test cases used. Maximum reduction in bandwidth tends to occur in cases where the initial bandwidth was high. The IFK algorithm tends to have higher bandwidth reduction than the other algorithms. The figures below illustrate the relative performance of the IFK algorithm versus the other reference algorithms.

IFK tends to consistently achieve better profile reduction than GPS but consistently lags behind SLN, which is the leader in profile reduction.

IFK tends to achieve better performance than all the other algorithms in bandwidth reduction, particularly against GPS and then SLN. The closest contender in bandwidth reduction is GBS. Figures 4-7 illustrate various aspects of the algorithm's performance as indicated in the figures' titles.

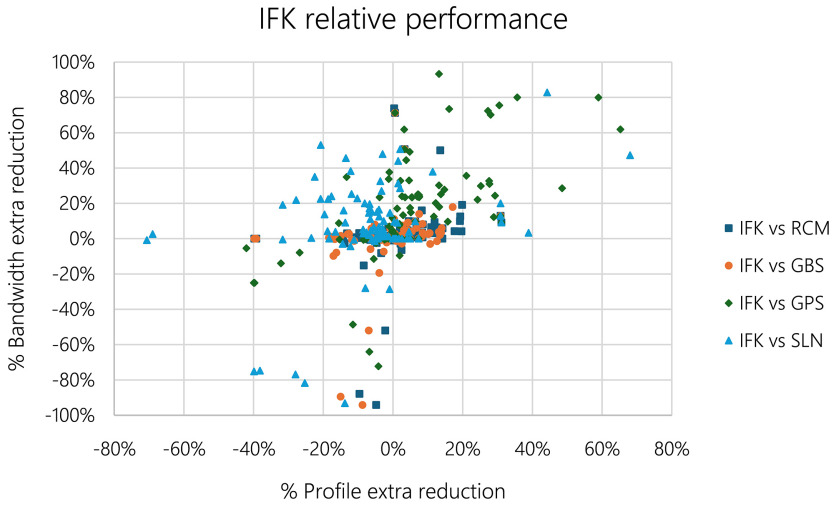


Figure 3. IFK bandwidth and profile improvement scatter.

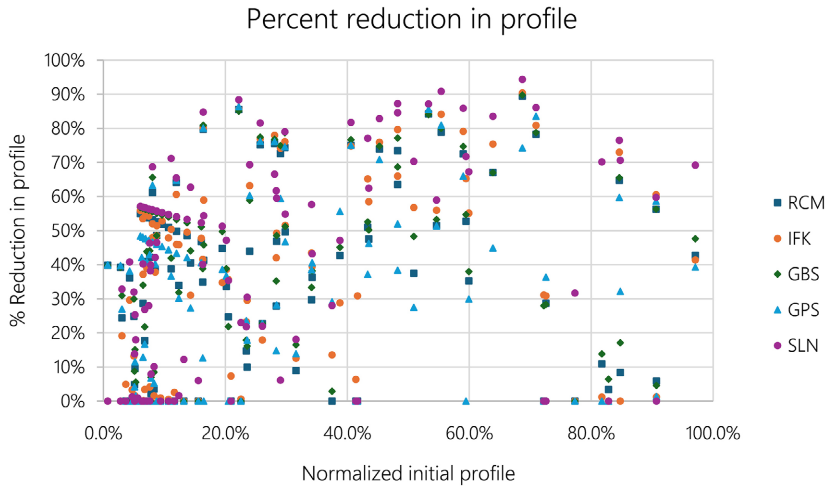


Figure 4. Percent reduction in profile against normalized initial profile.

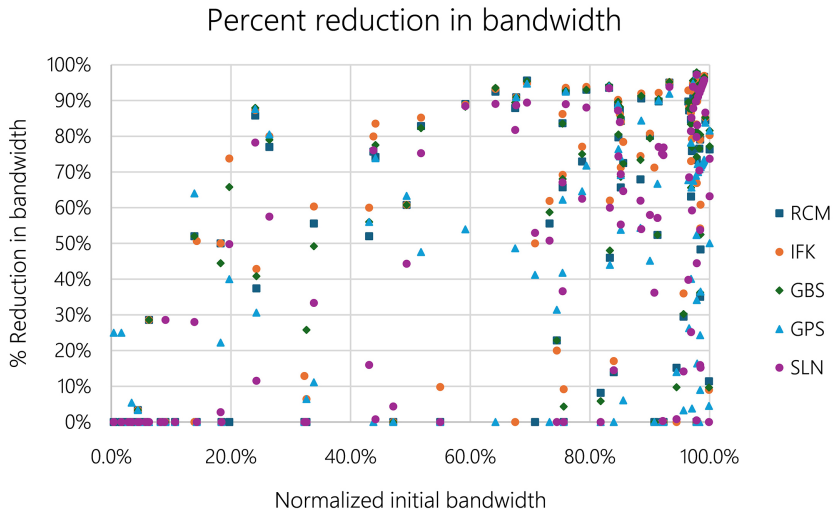


Figure 5. Percent reduction in bandwidth against normalized initial bandwidth.

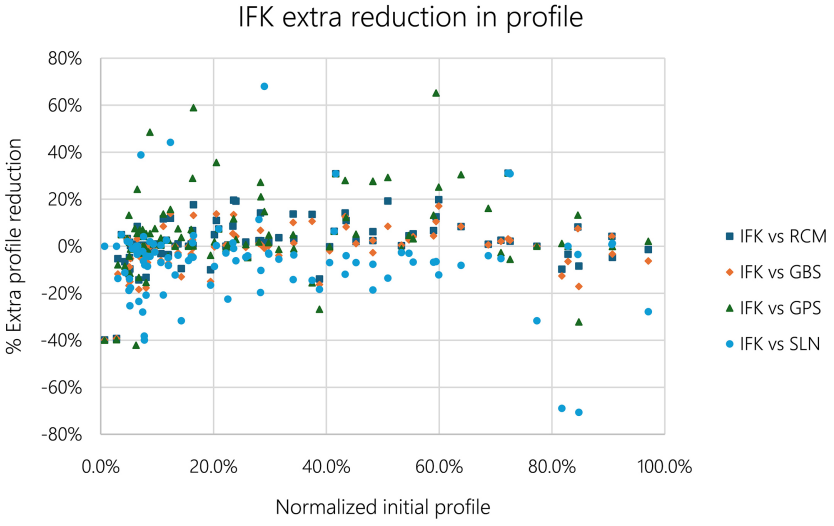


Figure 6. Extra profile reduction of IFK algorithm against the other reference algorithms.

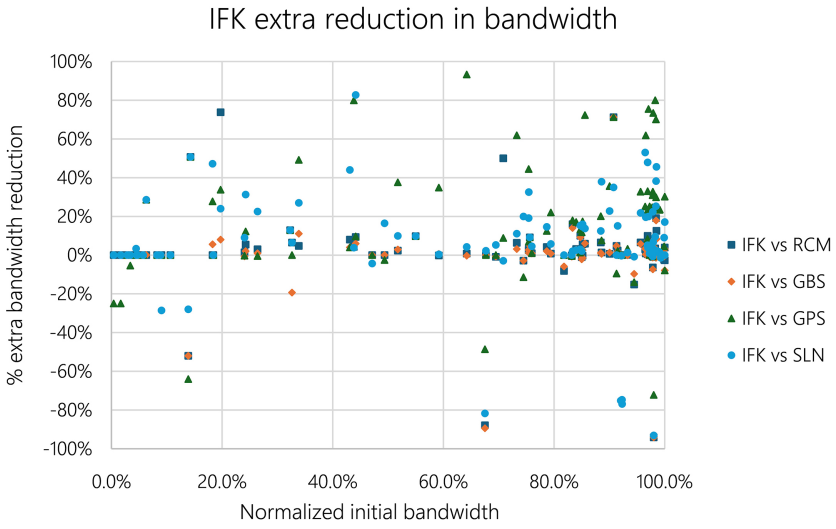


Figure 7. Extra bandwidth reduction of IFK algorithm against the other reference algorithms.

4.2. Reordering Time

Besides bandwidth and profile reduction, execution time is an important consideration. Figures 8-10 below illustrate the relative speeds of the various algorithms compared. The runs were executed on an HP ZBook 15 G3 with an Intel Xeon CPU-1505 M v5 running at 2.8 Ghz with 64 GB of system memory under Windows 10 Pro 64 bit version as a console app.

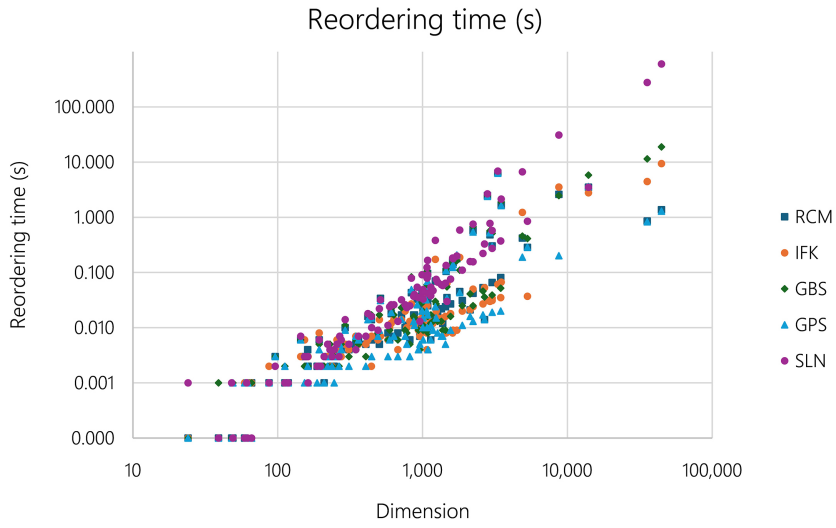


Figure 8. Variation of reordering time versus matrix dimension.

One important aspect of tuning the algorithm is balancing the convergence tolerance on bandwidth and the maximum number of iterations allowed. Tightening the convergence tolerance requires increasing the maximum number of iterations allowed. This tends to increase the reordering time, particularly for the cases where the algorithm fails to improve on the initial ordering.

Through testing, a convergence tolerance of 1% was adopted and a maximum number of iterations equal to 2 * (N_nzero/N_dim) were found optimal, where N_nzero is the number of non-zero terms in the matrix and N_dim is the dimension of the square symmetric matrix. When the algorithm finds a successful ordering, it uses less iterations than the maximum number of iterations set. When it fails to find a successful ordering, the maximum number of iterations consumed is still limited.

As both axes are logarithmic, the relationship between reordering time and matrix dimension is a power relation. The 0.000 on the time axis indicates that the execution time is shorter than 1ms. The SLN algorithm tends to have the longest run time. The IFK algorithm is faster than many other algorithms, particularly for large matrices.

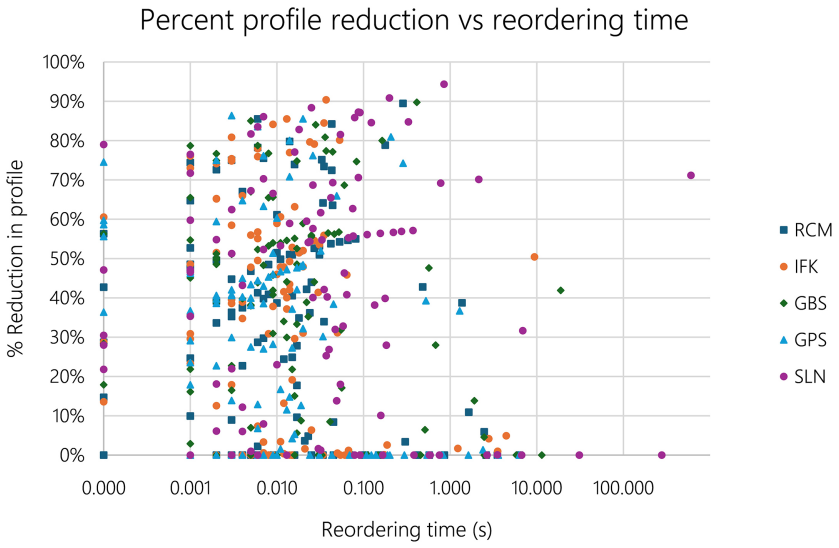


Figure 9. Percent profile reduction versus reordering time.

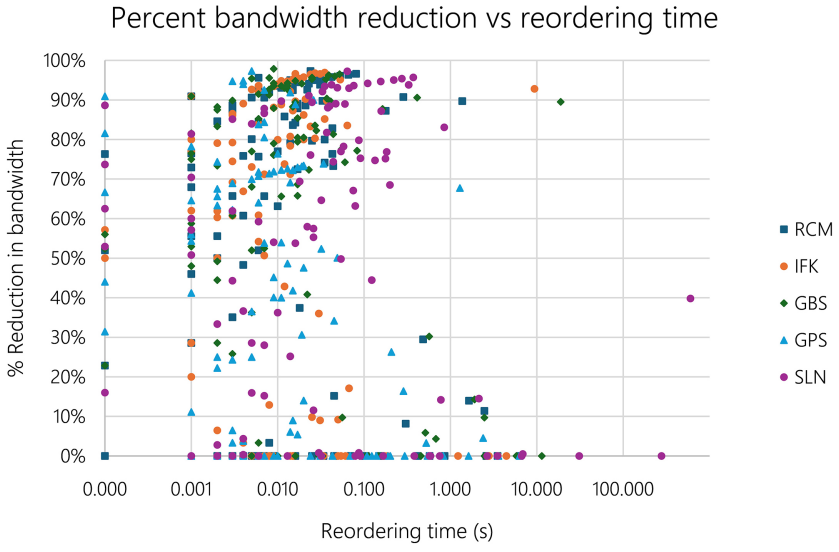


Figure 10. Percent bandwidth reduction versus reordering time.

At any profile reduction percentage, the IFK algorithm tends to have the shortest reordering time.

Again, at any bandwidth reduction percentage, the IFK algorithm times tend to be among the shortest. Alternatively, for any reordering time, the IFK algorithm bandwidth reduction tends to be among the highest.

4.3. Matrix Profile

Finally, Figures 11-14 below show the kind of reordering obtained by the IFK algorithm compared to the other algorithms. To demonstrate the versatility of solutions found, these cases are picked among those where all algorithms find an optimal renumbering. In the figures below, the original matrix non-zeros pattern is in the top row to the left. The RCM pattern is in the top row in the middle. The IFK pattern is in the top row to the right. The GBS pattern is in the bottom row to the left. The GPS pattern is in the bottom row in the middle. The SLN pattern is in the bottom row to the right.

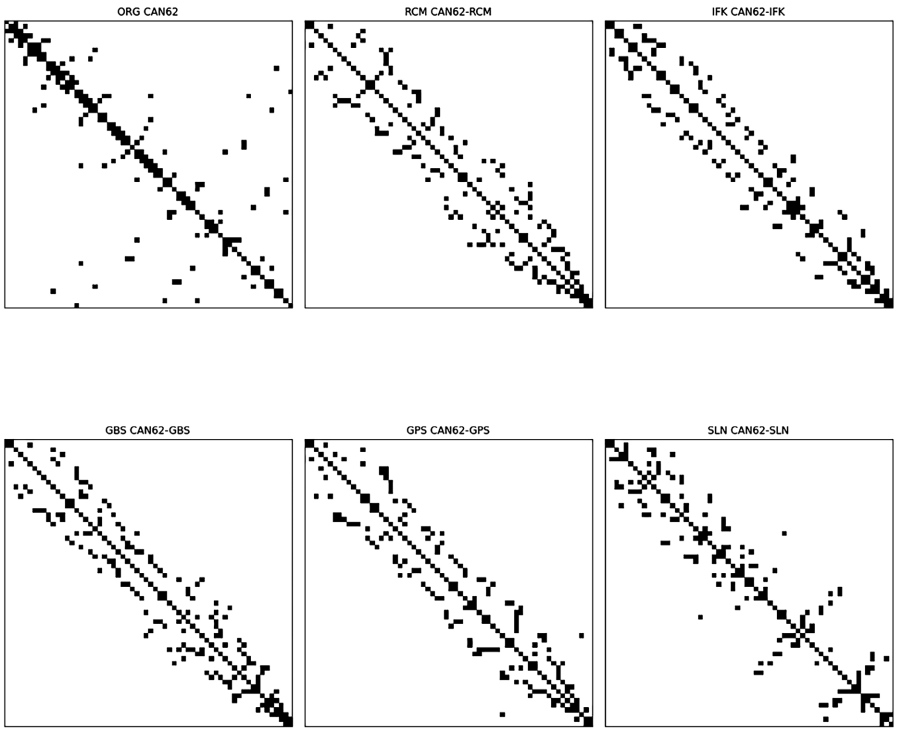


Figure 11. CAN62.

IFK obtained a 30% reduction in profile: 20% better than RCM, 13% better than GBS, 12% better than GPS and 1% less than SLN. IFK obtained a 77% reduction in bandwidth: 4% better than RCM, 2% better than GBS, 13% better than GPS and 15% better than SLN.

IFK obtained a 55% reduction in profile: 20% better than RCM, 17% better than GBS, 25% better than GPS and 12% less than SLN. IFK obtained a 54% reduction in bandwidth: 19% better than RCM, 18% better than GBS, 30% better than GPS

and 38% better than SLN.

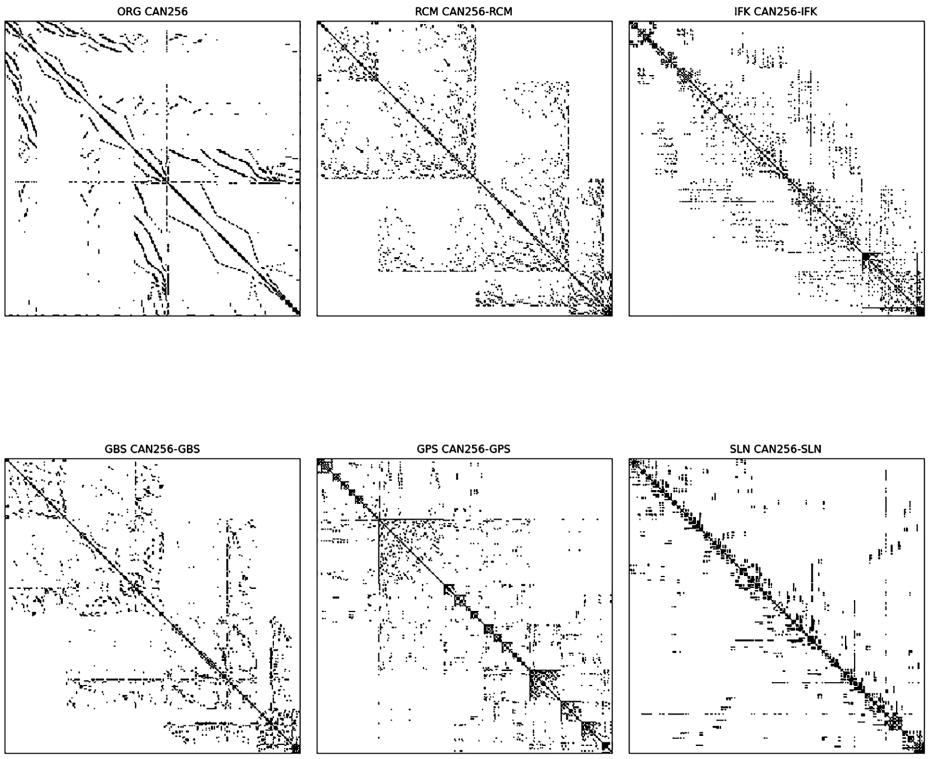


Figure 12. CAN256.

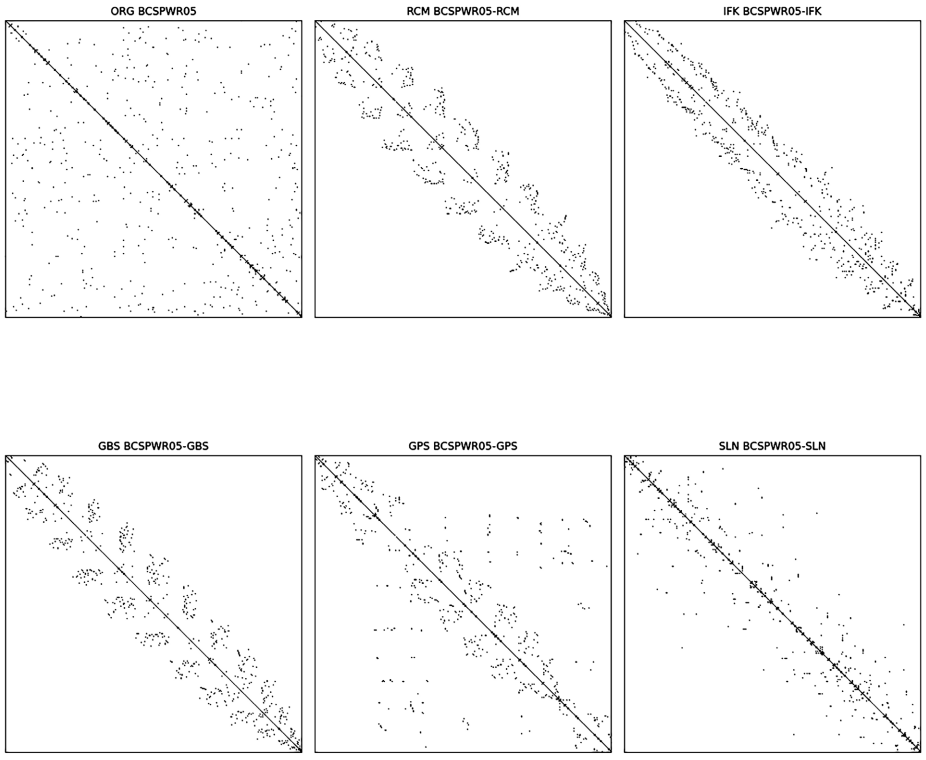


Figure 13. BCSPWR05.

IFK obtained a 65% reduction in profile: 14% better than RCM, 13% better than GBS, 28% better than GPS and 12% less than SLN. IFK obtained a 79% reduction in bandwidth: equal to RCM, 1% less than GBS, 70% better than GPS and 25% better than SLN.

IFK obtained a 43% reduction in profile: 14% better than RCM, 10% better than GBS, 5% better than GPS and 14% less than SLN. IFK obtained a 71% reduction in bandwidth: 6% better than RCM, 3% better than GBS, 17% better than GPS and 16% better than SLN.

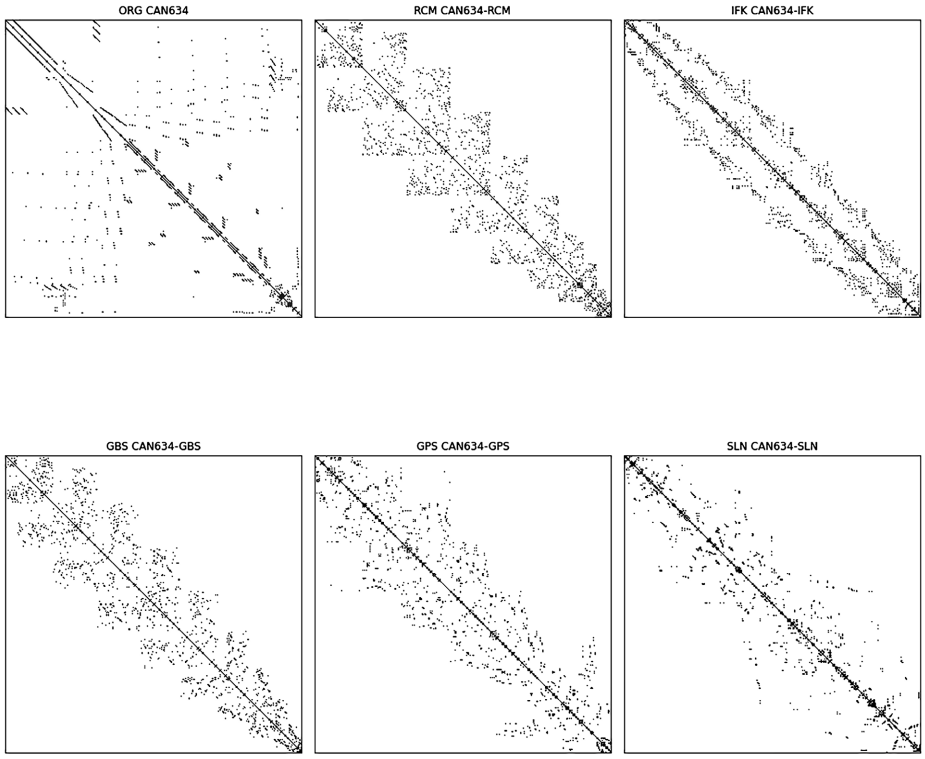


Figure 14. CAN634.

5. Conclusion

A new node renumbering algorithm is presented that adopts a different principle than the other reference algorithms currently in use. As all algorithms are based on heuristic principles, no one can claim to be superior to the others except by their performance on actual cases. In the comparisons done, it is shown that the new algorithm has performance that can significantly exceed that of the other reference algorithms, particularly in bandwidth reduction. Its execution time is often faster than that of the other algorithms. In addition, it has the potential to be run on parallel processors to further speed up the execution.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Rosen, R. (1968) Matrix Bandwidth Minimization. *Proceedings of the* 1968 23*rd ACM National Conference*, Rosenberg, 27-29 August 1968, 585-595. https://doi.org/10.1145/800186.810622

[2] Alway, G.G. (1965) An Algorithm for Reducing the Bandwidth of a Matrix of Symmetrical Configuration. *The Computer Journal*, **8**, 264-272. https://doi.org/10.1093/comjnl/8.3.264

[3] Smyth, W.F. (1985) Algorithms for the Reduction of Matrix Bandwidth and Profile. *Journal of Computational and Applied Mathematics*, **12**, 551-561. https://doi.org/10.1016/0377-0427(85)90048-2

[4] Cuthill, E. and McKee, J. (1969) Reducing the Bandwidth of Sparse Symmetric Matrices. *Proceedings of the* 1969 24*th National Conference*, New York, 26-28 August 1969, 157-172. https://doi.org/10.1145/800195.805928

[5] George, A. (1971) Computer Implementation of the Finite Element Method. Ph. D Dissertation Tech. Rep. STAN-CS-71-208, Computer Science Department Stanford University.

[6] Liu, W. and Sherman, A.H. (1976) Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices. *SIAM Journal on Numerical Analysis*, **13**, 198-213. https://doi.org/10.1137/0713020

[7] Gibbs, N.E., Poole, Jr., W.G. and Stockmeyer, P.K. (1976) An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. *SIAM Journal on Numerical Analysis*, **13**, 236-250. https://doi.org/10.1137/0713023

[8] Gibbs, N.E. (1976) Algorithm 509: A Hybrid Profile Reduction Algorithm [F1]. *ACM Transactions on Mathematical Software*, **2**, 378-387. https://doi.org/10.1145/355705.355713

[9] Sloan, S.W. (1986) An Algorithm for Profile and Wavefront Reduction of Sparse Matrices. *International Journal for Numerical Methods in Engineering*, **23**, 239-251. https://doi.org/10.1002/nme.1620230208

[10] Koohestani, B. (2013) Genetic Hyper-Heuristics for Graph Layout Problems. Ph.D. Thesis, University of Essex.

[11] Barnard, S.T., Pothen, A. and Simon, H.D. (1993) A Spectral Algorithm for Envelope Reduction of Sparse Matrices. *Proceedings of the* 1993 *ACM/IEEE Conference on Supercomputing*, Portland, 15-19 November 1993, 493-502. https://doi.org/10.1145/169627.169790

[12] Mafteiu-Scai, L.O., Negru, V., Zaharie, D. and Aritoni, O. (2011) Average Bandwidth Reduction in Sparse Matrices Using Hybrid Heuristics. Studia Universitatis Babes-Bolyai, Informatica Vol. LVI No. 3.

Terminology

Adjacency list: The adjacency list of a node is the list of nodes connected to this node by an edge in the graph. In a symmetric matrix, for node I, the adjacency list consists of the set of values J for which $A_{IJ} \neq 0$ and $J \neq I$.

Nodal degree: The nodal degree is the length of the adjacency list.

Root level structure (abbreviated as RLS): The root level structure is a series of sets starting from a root node such that each consecutive set contains the nodes connected to the nodes of the previous set. The first set after the root node consists of the adjacency list of this node. The next set consists of the union of the adjacency lists of the nodes in the previous set, excluding nodes already figuring in the previous set.

Length or height of an RLS: the number of levels or sets in the RLS.

Width of an RLS level: Is the number of nodes in that level.

Width of an RLS: Is the maximum width of all the RLS level widths.

Appendix 1: Test Matrices

Table A1. Properties of test cases used.

BOEING HARWELL MATRICES				ORIGINAL			
Matrix	Dimension	Non zeros	Sparsity	Profile	Bandwidth	2.Profile/Dim^2	Bandwidth /Dim
BCSPWR01	39	167	11.0%	295	38	38.8%	100.0%
BCSPWR02	49	214	8.9%	450	34	37.5%	70.8%
BCSPWR03	118	592	4.3%	1,427	115	20.5%	98.3%
BCSPWR04	274	1,876	2.5%	23,979	265	63.9%	97.1%
BCSPWR05	443	2,063	1.1%	42,534	435	43.3%	98.4%
BCSPWR06	1,454	6,751	0.3%	81,554	1,341	7.7%	92.3%
BCSPWR07	1,612	7,433	0.3%	96,670	1,487	7.4%	92.3%
BCSPWR08	1,624	7,671	0.3%	102,276	1,494	7.8%	92.1%
BCSPWR09	1,723	8,232	0.3%	822,213	1,663	55.4%	96.6%
BCSPWR10	5,300	27,138	0.1%	9,649,155	5,189	68.7%	97.9%
BCSSTK01	48	355	15.4%	836	35	72.6%	74.5%
BCSSTK03	112	748	6.0%	548	7	8.7%	6.3%
BCSSTK05	153	2,477	10.6%	2,449	28	20.9%	18.4%
BCSSTK06	420	7,503	4.3%	14,354	47	16.3%	11.2%
BCSSTK08	1,074	13,596	1.2%	238,694	590	41.4%	55.0%
BCSSTK09	1,083	19,145	1.6%	62,579	62	10.7%	5.7%
BCSSTK10	1,086	21,907	1.9%	36,981	37	6.3%	3.4%
BCSSTK11	1,473	33,773	1.6%	134,185	650	12.4%	44.2%
BCSSTK14	1,806	58,396	1.8%	189,800	161	11.6%	8.9%
BCSSTK16	4,884	263,859	1.1%	610,131	140	5.1%	2.9%
BCSSTK19	817	7,648	1.1%	74,053	567	22.2%	69.5%
BCSSTK27	1,224	57,335	3.8%	49,292	56	6.6%	4.6%
BCSSTK29	13,992	633,470	0.3%	7,441,194	1,157	7.6%	8.3%
BCSSTK31	35,588	1,216,983	0.1%	23,168,692	1,668	3.7%	4.7%
BCSSTK32	44,609	2,059,280	0.1%	110,527,814	43,030	11.1%	96.5%
BCSSTK33	8,738	600,622	0.8%	3,571,397	932	9.4%	10.7%
BLCKHOLE	2,132	16,998	0.4%	188,491	1,805	8.3%	84.7%
CAN1054	1,054	13,234	1.2%	268,001	1,030	48.2%	97.8%
CAN1072	1,072	13,500	1.2%	277,254	1,048	48.3%	97.9%
CAN144	144	1,431	6.9%	7,356	142	70.9%	99.3%
CAN161	161	1,529	5.9%	3,379	79	26.1%	49.4%
CAN187	187	1,672	4.8%	5,213	63	29.8%	33.9%
CAN229	229	1,997	3.8%	8,975	172	34.2%	75.4%

Continued

CAN24	24	175	30.4%	261	21	90.6%	91.3%
CAN256	256	3,163	4.8%	19,635	251	59.9%	98.4%
CAN268	268	3,337	4.6%	18,279	263	50.9%	98.5%
CAN292	292	2,827	3.3%	23,293	282	54.6%	96.9%
CAN445	445	4,245	2.1%	22,323	403	22.5%	90.8%
CAN61	61	612	16.4%	1,574	50	84.6%	83.3%
CAN62	62	278	7.2%	453	48	23.6%	78.7%
CAN634	634	7,853	2.0%	68,595	539	34.1%	85.2%
CAN715	715	7,365	1.4%	72,425	611	28.3%	85.6%
CAN73	73	444	8.3%	799	39	30.0%	54.2%
CAN838	838	10,830	1.5%	207,201	837	59.0%	100.0%
CAN96	96	858	9.3%	1,455	31	31.6%	32.6%
CEGB2802	2,802	17,966	0.2%	3,560,497	2,798	90.7%	99.9%
CEGB2919	2,919	18,085	0.2%	4,135,834	2,791	97.1%	95.6%
CEGB3024	3,024	19,354	0.2%	3,787,382	2,473	82.8%	81.8%
CEGB3306	3,306	28,254	0.3%	4,225,634	3,233	77.3%	97.8%
DWT1005	1,005	9,617	1.0%	121,087	851	24.0%	84.8%
DWT1007	1,007	9,578	0.9%	25,817	986	5.1%	98.0%
DWT1242	1,242	11,662	0.8%	110,196	936	14.3%	75.4%
DWT162	162	1,340	5.1%	2,646	156	20.2%	96.9%
DWT193	193	3,678	9.9%	7,761	62	41.7%	32.3%
DWT198	198	1,586	4.0%	5,689	36	29.0%	18.3%
DWT209	209	1,947	4.5%	9,504	184	43.5%	88.5%
DWT221	221	1,843	3.8%	9,911	187	40.6%	85.0%
DWT245	245	1,702	2.8%	3,960	115	13.2%	47.1%
DWT2600	2,680	27,695	0.4%	587,958	2,499	16.4%	93.3%
DWT310	310	2,754	2.9%	2,697	28	5.6%	9.1%
DWT346	346	3,565	3.0%	9,316	318	15.6%	92.2%
DWT361	361	3,310	2.5%	5,086	50	7.8%	13.9%
DWT419	419	3,975	2.3%	39,733	356	45.3%	85.2%
DWT492	492	3,642	1.5%	33,965	435	28.1%	88.6%
DWT503	503	6,524	2.6%	35,916	452	28.4%	90.0%
DWT512	512	4,013	1.5%	6,162	73	4.7%	14.3%
DWT59	59	322	9.3%	408	25	23.4%	43.1%
DWT592	592	5,689	1.6%	28,806	259	16.4%	43.8%
DWT607	607	5,734	1.6%	30,019	147	16.3%	24.3%
DWT66	66	384	8.8%	648	44	29.8%	67.7%

Continued

DWT758	758	6,748	1.2%	23,116	200	8.0%	26.4%
DWT869	869	8,150	1.1%	19,529	586	5.2%	67.5%
DWT87	87	621	8.2%	2,250	63	59.5%	73.3%
DWT878	878	8,322	1.1%	26,059	519	6.8%	59.2%
DWT918	918	8,294	1.0%	108,358	839	25.7%	91.5%
DWT992	992	17,728	1.8%	262,307	513	53.3%	51.8%
JAGMESH1	936	7,196	0.8%	37,241	778	8.5%	83.2%
JAGMESH2	1,009	7,867	0.8%	54,299	991	10.7%	98.3%
JAGMESH3	1,089	8,447	0.7%	70,875	1,058	12.0%	97.2%
JAGMESH4	1,440	10,940	0.5%	44,472	1,408	4.3%	97.8%
JAGMESH5	1,180	8,926	0.6%	34,854	757	5.0%	64.2%
JAGMESH6	1,377	10,367	0.5%	28,754	331	3.0%	24.1%
JAGMESH7	1,138	8,583	0.7%	42,016	903	6.5%	79.4%
JAGMESH8	1,141	8,602	0.7%	34,431	225	5.3%	19.7%
JAGMESH9	1,349	10,445	0.6%	69,013	1,024	7.6%	76.0%
LOCK1074	1,068	13,139	1.2%	483,440	1,008	84.8%	94.5%
LOCK2232	2,232	24,724	0.5%	1,798,963	1,687	72.2%	75.6%
LOCK3491	3,491	32,917	0.3%	4,981,162	2,931	81.7%	84.0%
LSHP1009	1,009	7,867	0.8%	54,299	991	10.7%	98.3%
LSHP1270	1,270	9,931	0.6%	77,526	1,250	9.6%	98.5%
LSHP1561	1,561	12,235	0.5%	106,636	1,539	8.8%	98.7%
LSHP1882	1,882	14,779	0.4%	142,298	1,858	8.0%	98.8%
LSHP2233	2,233	17,563	0.4%	185,181	2,207	7.4%	98.9%
LSHP2614	2,614	20,587	0.3%	235,954	2,586	6.9%	99.0%
LSHP265	265	2,011	2.9%	6,841	256	19.5%	97.0%
LSHP3025	3,025	23,851	0.3%	295,286	2,995	6.5%	99.0%
LSHP3446	3,466	27,355	0.2%	363,846	3,434	6.1%	99.1%
LSHP406	406	3,115	1.9%	13,226	394	16.0%	97.3%
LSHP577	577	4,459	1.3%	22,818	563	13.7%	97.7%
LSHP778	778	6,043	1.0%	36,286	762	12.0%	98.1%
NOS1	237	1,252	2.2%	788	4	2.8%	1.7%
NOS2	957	5,092	0.6%	3,188	4	0.7%	0.4%
NOS6	675	3,927	0.9%	16,230	30	7.1%	4.5%

Table A2. Numerical results.

Matrix	RCM			IFK			GBS			GPS			SLN		
	Profile	Band-width	Time	Profile	Band-width	Time	Profile	Band-width	Time	Profile	Band-width	Time	Profile	Band-width	Time
BCSPWR01	169	9	0.000	210	10	0.000	162	7	0.001	131	7	0.000	156	10	0.000
BCSPWR02	450	34	0.000	389	17	0.000	437	16	0.001	319	20	0.001	324	16	0.000

Continued

BCSPWR03	1,075	27	0.001	918	23	0.001	1,115	27	0.001	1,427	115	0.001	923	34	0.001
BCSPWR04	7,906	64	0.004	5,897	55	0.003	7,915	61	0.005	13,206	255	0.004	3,953	108	0.006
BCSPWR05	20,841	91	0.014	14,814	91	0.002	20,186	85	0.017	26,732	396	0.015	9,746	201	0.016
BCSPWR06	81,554	1,341	0.105	81,554	1,341	0.008	81,554	1,341	0.112	81,554	1,341	0.108	50,418	339	0.134
BCSPWR07	96,670	1,487	0.157	96,670	1,487	0.008	96,670	1,487	0.148	96,670	1,487	0.123	69,635	344	0.183
BCSPWR08	102,276	1,494	0.149	102,276	1,494	0.009	102,276	1,494	0.160	102,276	1,494	0.139	61,499	371	0.177
BCSPWR09	174,037	212	0.178	130,501	198	0.009	164,317	204	0.164	156,819	1,226	0.207	75,288	524	0.199
BCSPWR10	1,016,420	481	0.285	928,846	530	0.037	988,429	489	0.412	2,484,907	4,339	0.286	543,754	878	0.848
BCSSTK01	596	27	0.000	578	28	0.001	599	27	0.000	532	24	0.000	836	35	0.001
BCSSTK03	282	5	0.001	282	5	0.001	282	5	0.002	548	7	0.002	293	7	0.001
BCSSTK05	2,449	28	0.003	2,269	28	0.006	2,449	28	0.002	2,449	28	0.001	2,449	28	0.003
BCSSTK08	238,694	590	0.009	223,504	532	0.025	238,694	590	0.105	238,694	590	0.010	238,694	590	0.079
BCSSTK09	62,579	62	0.096	62,263	62	0.050	62,579	62	0.090	62,579	62	0.069	62,579	62	0.166
BCSSTK10	36,981	37	0.025	36,981	37	0.054	36,981	37	0.016	21,419	35	0.017	36,981	37	0.091
BCSSTK11	88,663	168	0.035	72,657	107	0.064	91,444	146	0.055	93,727	170	0.034	131,999	645	0.030
BCSSTK14	189,800	161	0.045	184,921	161	0.187	189,800	161	0.110	189,800	161	0.043	189,800	161	0.586
BCSSTK16	610,131	140	0.420	599,816	140	1.226	610,131	140	0.455	610,131	140	0.189	610,131	140	6.667
BCSSTK19	10,735	25	0.006	10,714	30	0.013	11,075	26	0.005	10,099	30	0.003	8,614	60	0.025
BCSSTK27	49,292	56	0.070	49,240	56	0.173	49,292	56	0.030	49,292	56	0.024	49,292	56	0.381
BCSSTK29	7,441,194	1,157	3.510	7,132,151	1,157	2.783	7,441,194	1,157	5.805	7,441,194	1,157	3.593	7,437,560	1,157	3.497
BCSSTK31	23,168,692	1,668	0.857	22,027,503	1,668	4.436	23,168,692	1,668	11.422	23,168,692	1,668	0.829	23,168,692	1,668	276.941
BCSSTK32	67,702,082	4,430	1.371	54,805,328	3,089	9.396	64,209,383	4,512	18.886	69,978,783	13,893	1.288	31,875,564	25,899	600.905
BCSSTK33	3,571,397	932	2.606	3,537,209	932	3.532	3,571,397	932	2.476	3,571,397	932	0.201	3,571,397	932	30.910
BLCKHOLE	181,637	206	0.021	185,487	177	0.021	172,494	187	0.041	178,531	201	0.016	169,451	231	0.158
CAN1054	71,251	206	0.035	54,529	172	0.024	61,227	192	0.044	128,744	491	0.032	34,129	208	0.087
CAN1072	101,084	280	0.044	94,310	347	0.004	86,880	270	0.060	170,784	690	0.045	42,774	582	0.123
CAN144	1,600	22	0.006	1,408	21	0.003	1,564	21	0.006	1,210	23	0.006	1,023	19	0.007
CAN161	2,611	31	0.004	2,774	31	0.003	2,611	31	0.003	2,611	29	0.002	2,637	44	0.003
CAN187	2,628	28	0.002	2,527	25	0.002	2,544	32	0.002	2,777	56	0.001	2,355	42	0.002
CAN229	5,716	59	0.003	5,427	53	0.003	5,549	55	0.005	5,332	65	0.003	5,099	109	0.004
CAN24	114	10	0.000	103	9	0.000	114	10	0.000	108	7	0.000	105	9	0.001
CAN256	12,712	163	0.003	8,816	115	0.006	12,177	160	0.005	13,762	190	0.003	6,428	211	0.005
CAN268	11,428	136	0.004	7,904	103	0.006	9,453	125	0.007	13,254	167	0.005	5,432	223	0.007
CAN292	11,302	104	0.010	10,255	76	0.005	10,893	97	0.011	11,330	169	0.009	9,561	211	0.014
CAN445	22,323	403	0.006	22,198	116	0.007	22,323	403	0.010	22,323	403	0.003	17,186	257	0.010
CAN61	555	27	0.001	425	19	0.001	543	26	0.001	634	28	0.000	370	20	0.001
CAN62	408	13	0.001	319	11	0.001	380	12	0.001	372	17	0.001	315	18	0.000

Continued

CAN634	48,186	185	0.007	38,779	155	0.014	45,742	169	0.017	42,151	249	0.007	29,056	241	0.026
CAN715	52,272	168	0.017	41,979	132	0.014	46,909	169	0.023	61,714	574	0.014	27,746	216	0.032
CAN838	57,062	198	0.043	43,230	165	0.027	52,466	191	0.083	70,563	418	0.049	29,272	308	0.079
CAN96	1,325	31	0.003	1,272	29	0.002	1,215	23	0.003	1,253	29	0.003	1,192	31	0.002
CEGB2802	3,349,701	2,479	2.471	3,516,466	2,546	0.031	3,397,712	2,527	2.467	3,512,846	2,672	2.394	3,560,497	2,798	2.656
CEGB2919	2,367,123	1,968	0.482	2,424,039	1,786	0.030	2,166,754	1,948	0.568	2,511,941	2,700	0.525	1,274,451	2,395	0.776
CEGB3024	3,659,587	2,271	0.303	3,787,382	2,473	0.034	3,543,442	2,328	0.512	3,787,382	2,473	0.296	3,787,382	2,473	0.574
CEGB3306	4,225,634	3,233	6.324	4,225,634	3,233	0.061	4,225,634	3,233	6.616	4,225,634	3,233	6.284	2,887,457	3,217	6.876
DWT1005	67,845	173	0.025	44,575	88	0.016	49,751	166	0.020	48,082	201	0.010	37,132	218	0.044
DWT1007	24,572	58	0.023	25,817	986	0.014	23,555	58	0.019	24,734	274	0.015	22,249	68	0.049
DWT1242	65,525	153	0.015	75,974	129	0.020	61,654	154	0.027	80,148	545	0.015	41,084	308	0.075
DWT162	1,756	24	0.002	1,625	21	0.003	1,619	26	0.002	1,675	34	0.001	1,398	29	0.001
DWT193	7,761	62	0.006	5,365	54	0.008	7,761	62	0.005	7,761	62	0.004	7,761	62	0.006
DWT198	1,558	18	0.002	1,471	18	0.002	1,426	20	0.002	2,309	28	0.002	5,341	35	0.002
DWT209	4,986	59	0.001	3,946	47	0.003	4,733	49	0.002	5,109	84	0.001	3,569	70	0.003
DWT221	2,477	24	0.003	2,496	26	0.003	2,312	22	0.002	2,479	48	0.002	1,812	30	0.005
DWT245	3,960	115	0.002	3,960	115	0.003	3,960	115	0.003	3,960	115	0.001	3,477	110	0.004
DWT2600	118,939	125	0.014	116,840	122	0.053	112,515	120	0.036	117,429	203	0.014	89,591	155	0.329
DWT310	2,697	28	0.004	2,697	28	0.004	2,697	28	0.003	2,697	28	0.002	2,671	20	0.005
DWT346	9,316	318	0.005	9,316	318	0.007	9,316	318	0.006	9,316	318	0.007	8,754	317	0.004
DWT361	4,973	24	0.006	5,086	50	0.005	4,732	24	0.005	4,741	18	0.006	4,683	36	0.007
DWT419	10,365	56	0.016	9,564	57	0.006	10,014	52	0.017	11,602	110	0.014	6,818	109	0.018
DWT492	8,315	41	0.007	7,490	35	0.006	7,893	38	0.008	8,103	68	0.007	11,361	200	0.009
DWT503	19,089	90	0.005	18,225	87	0.014	18,478	93	0.017	25,789	248	0.009	14,543	190	0.022
DWT512	6,162	73	0.034	5,958	36	0.007	6,162	73	0.033	6,162	73	0.031	6,085	73	0.032
DWT59	348	12	0.000	313	10	0.001	335	11	0.000	312	11	0.001	319	21	0.000
DWT592	16,913	63	0.006	11,824	52	0.010	15,614	62	0.009	28,806	259	0.006	13,147	62	0.024
DWT607	19,551	92	0.018	17,542	84	0.012	18,350	87	0.022	26,226	102	0.019	17,988	130	0.026
DWT66	166	4	0.001	155	4	0.001	138	4	0.001	165	4	0.000	136	5	0.000
DWT758	8,984	46	0.010	12,038	40	0.020	7,947	42	0.009	8,487	39	0.007	7,238	85	0.026
DWT869	17,651	71	0.017	19,529	586	0.015	16,590	62	0.016	17,280	301	0.013	14,587	107	0.037
DWT87	1,064	28	0.001	782	24	0.002	1,019	26	0.001	2,250	63	0.001	636	31	0.001
DWT878	21,437	57	0.017	25,168	58	0.011	20,378	61	0.015	21,702	239	0.011	19,059	60	0.040
DWT918	26,934	86	0.033	24,980	66	0.014	24,502	81	0.037	25,798	84	0.026	19,999	193	0.054
DWT992	41,352	88	0.043	40,668	76	0.035	41,872	91	0.028	37,952	269	0.020	33,679	127	0.091
JAGMESH1	22,833	50	0.010	23,156	51	0.009	21,633	45	0.009	22,258	47	0.004	21,555	51	0.035
JAGMESH2	26,630	61	0.015	28,311	58	0.011	24,979	64	0.009	30,196	290	0.006	24,577	78	0.033

Continued

JAGMESH3	25,413	63	0.034	27,912	55	0.011	24,465	47	0.008	24,983	55	0.004	24,478	65	0.042
JAGMESH4	28,378	38	0.024	31,297	48	0.016	26,338	30	0.009	27,528	38	0.005	26,326	39	0.064
JAGMESH5	26,178	57	0.015	30,251	51	0.012	24,419	49	0.013	34,854	757	0.009	23,709	83	0.047
JAGMESH6	21,735	47	0.012	23,253	42	0.015	19,862	40	0.009	20,988	41	0.007	19,317	72	0.058
JAGMESH7	29,958	63	0.006	26,404	56	0.013	27,725	63	0.012	36,603	255	0.006	25,114	108	0.038
JAGMESH8	34,431	225	0.016	34,315	59	0.012	32,512	77	0.017	33,896	135	0.011	28,236	113	0.054
JAGMESH9	39,899	76	0.022	41,384	66	0.013	38,585	75	0.013	39,349	77	0.007	37,040	113	0.060
LOCK1074	442,866	855	0.045	483,440	1,008	0.008	400,754	910	0.056	327,953	867	0.020	142,082	1,000	0.087
LOCK2232	1,798,963	1,687	0.574	1,239,075	1,532	0.050	1,295,606	1,614	0.679	1,798,963	1,687	0.541	1,798,963	1,687	0.752
LOCK3491	4,437,755	2,522	1.639	4,921,557	2,430	0.067	4,292,267	2,512	1.902	4,981,162	2,931	1.624	1,486,816	2,506	2.134
LSHP1009	26,630	61	0.031	28,311	58	0.012	24,979	64	0.011	30,196	290	0.006	24,577	78	0.033
LSHP1270	37,329	69	0.018	36,563	60	0.015	35,194	72	0.013	42,400	358	0.008	34,672	88	0.067
LSHP1561	50,535	77	0.027	51,781	71	0.018	47,855	80	0.016	57,568	433	0.009	47,228	98	0.076
LSHP1882	66,549	85	0.031	68,276	79	0.020	63,259	88	0.025	75,896	515	0.011	62,486	108	0.110
LSHP2233	85,597	93	0.042	84,697	78	0.024	81,639	96	0.025	97,825	604	0.013	80,739	118	0.156
LSHP2614	107,990	101	0.053	107,130	84	0.027	103,296	104	0.046	123,518	700	0.017	102,224	128	0.221
LSHP265	3,779	29	0.003	4,463	28	0.004	3,439	32	0.002	4,200	88	0.002	3,334	38	0.003
LSHP3025	133,944	109	0.066	137,349	103	0.031	128,459	112	0.039	153,450	803	0.019	127,238	138	0.274
LSHP3446	163,780	117	0.081	160,488	106	0.035	157,433	120	0.052	187,746	913	0.020	156,014	148	0.372
LSHP406	7,040	37	0.005	6,907	29	0.005	6,468	40	0.003	7,855	128	0.002	6,308	48	0.007
LSHP577	11,754	45	0.008	11,521	36	0.006	10,883	48	0.006	13,225	175	0.003	10,658	58	0.011
LSHP778	18,202	53	0.011	19,604	46	0.010	16,973	56	0.008	20,548	229	0.005	16,654	68	0.023
NOS1	479	4	0.002	788	4	0.002	479	4	0.002	475	3	0.002	788	4	0.003
NOS2	1,919	4	0.007	3,188	4	0.007	1,919	4	0.004	1,915	3	0.005	3,188	4	0.013
NOS6	9,592	29	0.008	9,915	29	0.004	9,110	29	0.006	9,607	29	0.003	16,230	30	0.013

Table A3. Relative performance of algorithm IFK versus the other reference algorithms.

Matrix	IFK vs RCM		IFK vs GBS		IFK vs GPS		IFK vs SLN	
	Profile red.	Bandwidth red.	Profile red.	Bandwidth red.	Profile red.	Bandwidth red.	Profile red.	Bandwidth red.
BCSPWR01	−14%	−3%	−16%	−8%	−27%	−8%	−18%	0%
BCSPWR02	14%	50%	11%	−3%	−16%	9%	−14%	−3%
BCSPWR03	11%	3%	14%	3%	36%	80%	0%	10%
BCSPWR04	8%	3%	8%	2%	30%	75%	−8%	20%
BCSPWR05	14%	0%	13%	−1%	28%	70%	−12%	25%
BCSPWR06	0%	0%	0%	0%	0%	0%	−38%	−75%
BCSPWR07	0%	0%	0%	0%	0%	0%	−28%	−77%
BCSPWR08	0%	0%	0%	0%	0%	0%	−40%	−75%

Continued

BCSPWR09	5%	1%	4%	0%	3%	62%	−7%	20%
BCSPWR10	1%	−1%	1%	−1%	16%	73%	−4%	7%
BCSSTK01	2%	−3%	3%	−3%	−6%	−11%	31%	20%
BCSSTK03	0%	0%	0%	0%	49%	29%	2%	29%
BCSSTK05	7%	0%	7%	0%	7%	0%	7%	0%
BCSSTK08	6%	10%	6%	10%	6%	10%	6%	10%
BCSSTK09	1%	0%	1%	0%	1%	0%	1%	0%
BCSSTK10	0%	0%	0%	0%	−42%	−5%	0%	0%
BCSSTK11	12%	9%	14%	6%	16%	10%	44%	83%
BCSSTK14	3%	0%	3%	0%	3%	0%	3%	0%
BCSSTK16	2%	0%	2%	0%	2%	0%	2%	0%
BCSSTK19	0%	−1%	0%	−1%	−1%	0%	−3%	5%
BCSSTK27	0%	0%	0%	0%	0%	0%	0%	0%
BCSSTK29	4%	0%	4%	0%	4%	0%	4%	0%
BCSSTK31	5%	0%	5%	0%	5%	0%	5%	0%
BCSSTK32	12%	3%	9%	3%	14%	25%	−21%	53%
BCSSTK33	1%	0%	1%	0%	1%	0%	1%	0%
BLCKHOLE	−2%	2%	−7%	1%	−4%	1%	−9%	3%
CAN1054	6%	3%	2%	2%	28%	31%	−8%	3%
CAN1072	2%	−6%	−3%	−7%	28%	33%	−19%	22%
CAN144	3%	1%	2%	0%	−3%	1%	−5%	−1%
CAN161	−5%	0%	−5%	0%	−5%	−3%	−4%	16%
CAN187	2%	5%	0%	11%	5%	49%	−3%	27%
CAN229	3%	3%	1%	1%	−1%	7%	−4%	33%
CAN24	4%	5%	4%	5%	2%	−10%	1%	0%
CAN256	20%	19%	17%	18%	25%	30%	−12%	38%
CAN268	19%	13%	8%	8%	29%	24%	−14%	46%
CAN292	4%	10%	3%	7%	5%	33%	−3%	48%
CAN445	1%	71%	1%	71%	1%	71%	−22%	35%
CAN61	8%	16%	7%	14%	13%	18%	−3%	2%
CAN62	20%	4%	13%	2%	12%	13%	−1%	15%
CAN634	14%	6%	10%	3%	5%	17%	−14%	16%
CAN715	14%	6%	7%	6%	27%	72%	−20%	14%
CAN838	7%	4%	4%	3%	13%	30%	−7%	17%
CAN96	4%	6%	−4%	−19%	−1%	0%	−5%	6%
CEGB2802	−5%	−2%	−3%	−1%	0%	5%	1%	9%
CEGB2919	−1%	7%	−6%	6%	2%	33%	−28%	22%

Continued

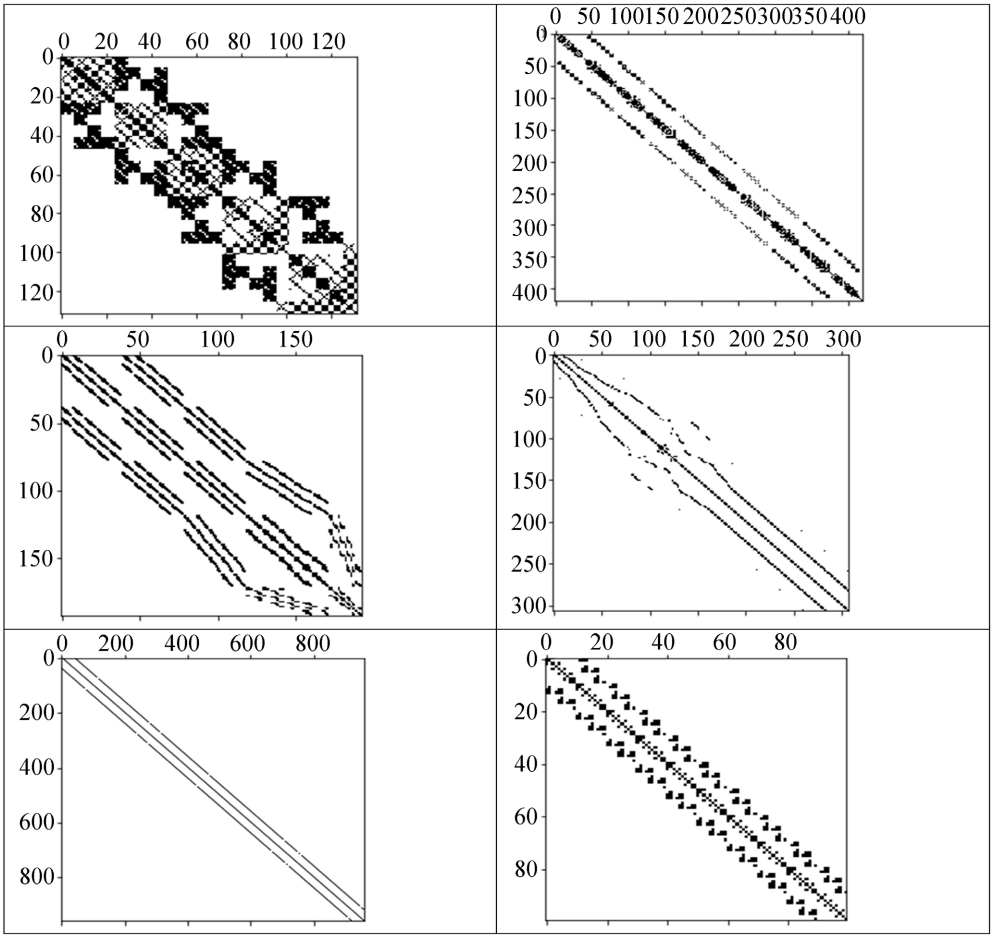
CEGB3024	−3%	−8%	−6%	−6%	0%	0%	0%	0%
CEGB3306	0%	0%	0%	0%	0%	0%	−32%	0%
DWT1005	19%	10%	4%	9%	3%	13%	−6%	15%
DWT1007	−5%	−94%	−9%	−94%	−4%	−72%	−14%	−93%
DWT1242	−9%	3%	−13%	3%	4%	44%	−32%	19%
DWT162	5%	2%	0%	3%	2%	8%	−9%	5%
DWT193	31%	13%	31%	13%	31%	13%	31%	13%
DWT198	2%	0%	−1%	6%	15%	28%	68%	47%
DWT209	11%	7%	8%	1%	12%	20%	−4%	13%
DWT221	0%	−1%	−2%	−2%	0%	12%	−7%	2%
DWT245	0%	0%	0%	0%	0%	0%	−12%	−4%
DWT2600	0%	0%	−1%	0%	0%	3%	−5%	1%
DWT310	0%	0%	0%	0%	0%	0%	−1%	−29%
DWT346	0%	0%	0%	0%	0%	0%	−6%	0%
DWT361	−2%	−52%	−7%	−52%	−7%	−64%	−8%	−28%
DWT419	2%	0%	1%	−1%	5%	15%	−7%	15%
DWT492	2%	1%	1%	1%	2%	8%	11%	38%
DWT503	2%	1%	1%	1%	21%	36%	−10%	23%
DWT512	3%	51%	3%	51%	3%	51%	2%	51%
DWT59	9%	8%	5%	4%	0%	4%	1%	44%
DWT592	18%	4%	13%	4%	59%	80%	5%	4%
DWT607	7%	5%	3%	2%	29%	12%	1%	31%
DWT66	2%	0%	−3%	0%	2%	0%	−3%	2%
DWT758	−13%	3%	−18%	1%	−15%	0%	−21%	23%
DWT869	−10%	−88%	−15%	−89%	−12%	−49%	−25%	−82%
DWT87	13%	6%	11%	3%	65%	62%	−6%	11%
DWT878	−14%	0%	−18%	1%	−13%	35%	−23%	0%
DWT918	2%	2%	0%	2%	1%	2%	−5%	15%
DWT992	0%	2%	0%	3%	−1%	38%	−3%	10%
JAGMESH1	−1%	0%	−4%	−1%	−2%	−1%	−4%	0%
JAGMESH2	−3%	0%	−6%	1%	3%	23%	−7%	2%
JAGMESH3	−4%	1%	−5%	−1%	−4%	0%	−5%	1%
JAGMESH4	−7%	−1%	−11%	−1%	−8%	−1%	−11%	−1%
JAGMESH5	−12%	1%	−17%	0%	13%	93%	−19%	4%
JAGMESH6	−5%	2%	−12%	−1%	−8%	0%	−14%	9%
JAGMESH7	8%	1%	3%	1%	24%	22%	−3%	6%
JAGMESH8	0%	74%	−5%	8%	−1%	34%	−18%	24%

Continued

JAGMESH9	−2%	1%	−4%	1%	−3%	1%	−6%	5%
LOCK1074	−8%	−15%	−17%	−10%	−32%	−14%	−71%	−1%
LOCK2232	31%	9%	3%	5%	31%	9%	31%	9%
LOCK3491	−10%	3%	−13%	3%	1%	17%	−69%	3%
LSHP1009	−3%	0%	−6%	1%	3%	23%	−7%	2%
LSHP1270	1%	1%	−2%	1%	8%	24%	−2%	2%
LSHP1561	−1%	0%	−4%	1%	5%	24%	−4%	2%
LSHP1882	−1%	0%	−4%	0%	5%	23%	−4%	2%
LSHP2233	0%	1%	−2%	1%	7%	24%	−2%	2%
LSHP2614	0%	1%	−2%	1%	7%	24%	−2%	2%
LSHP265	−10%	0%	−15%	2%	−4%	23%	−17%	4%
LSHP3025	−1%	0%	−3%	0%	5%	23%	−3%	1%
LSHP3446	1%	0%	−1%	0%	7%	24%	−1%	1%
LSHP406	1%	2%	−3%	3%	7%	25%	−5%	5%
LSHP577	1%	2%	−3%	2%	7%	25%	−4%	4%
LSHP778	−4%	1%	−7%	1%	3%	24%	−8%	3%
NOS1	−39%	0%	−39%	0%	−40%	−25%	0%	0%
NOS2	−40%	0%	−40%	0%	−40%	−25%	0%	0%
NOS6	−2%	0%	−5%	0%	−2%	0%	39%	3%
max	31%	74%	31%	71%	65%	93%	68%	83%
avg	1%	2%	−1%	0%	4%	13%	−5%	6%
min	−40%	−94%	−40%	−94%	−42%	−72%	−71%	−93%

Matrix names in red indicate cases where IFK failed to find a solution. Matrix names in orange indicate cases where IFK achieved an improvement in bandwidth or profile over all the other algorithms. Matrix names in yellow indicate cases where IFK achieved the maximum improvement in bandwidth or profile over some other algorithm. Cells in pink indicates cases where IFK trails behind another algorithm in improvement.

Appendix 2: Sample Matrices Where No Improvement Could be Achieved



Appendix 3: Comparison of Computed Results with Published Results

	ORIGINAL		As reported by references						As calculated in this document						
			RCM		GK		GPS		RCM		GK		GPS		
Matrix	Profile	Bandwidth	Profile	Bandwidth	Profile	Bandwidth	Profile	Bandwidth	Profile	Bandwidth	Profile	Bandwidth	Profile	Bandwidth	Reference
BCSPWR05	42,534	435	11,227						20,841	91	20,186	85	26,732	396	1
BCSPWR06	81,554	1,341	64,636						81,554	1,341	81,554	1,341	81,554	1,341	1
BCSPWR07	96,670	1,487	75,956						96,670	1,487	96,670	1,487	96,670	1,487	1
BCSPWR08	102,276	1,494	79,811						102,276	1,494	102,276	1,494	102,276	1,494	1
BCSPWR09	822,213	1,663	80,983						174,037	212	164,317	204	156,819	1,226	1
BCSPWR10	9,649,155	5,189	672,545						1,016,420	481	988,429	489	2,484,907	4,339	1
BCSSTK13	434,491	1,250	56,299	198	58,542	223	57,501	145	434,491	1,250	434,491	1,250	434,491	1,250	2
BCSSTK29	7,441,194	1,157	7,374,140	914	6,948,091	1,505	7,040,998	869	7,441,194	1,157	7,441,194	1,157	7,441,194	1,157	2
BCSSTK30	15,686,973	16,947	23,242,990	2,512	15,686,968	16,947	23,242,990	2,512	15,686,973	16,947	15,686,973	16,947	15,686,973	16,947	2
BCSSTK31	23,168,692	1,668	23,614,112	1,176	22,330,987	1,880	23,416,579	1,104	23,168,692	1,668	23,168,692	1,668	23,168,692	1,668	2
BCSSTK32	110,527,814	43,030	52,170,122	2,390	49,457,764	3,761	50,067,390	2,339	67,702,082	4,430	64,209,383	4,512	69,978,783	13,893	2
BCSSTK33	3,571,397	932	3,799,285	749	3,571,395	932	3,717,032	519	3,571,397	932	3,571,397	932	3,571,397	932	2
BLCKHOLE	188,491	1,805	171,437	105	169,219	134	173,243	106	181,637	206	172,494	187	178,531	201	2
CAN1072	277,254	1,048	56,361	175	48,538	234	74,067	159	101,084	280	86,880	270	170,784	690	2
SSTMODEL	108,747	161	105,421	88	104,562	125	110,936	83	108,747	161	108,747	161	108,747	161	2
DWT1005	121,087	851	43,068		40,141				67,845	173	49,751	166	48,082	201	3
DWT1007	25,817	986	24,703		22,465				24,572	58	23,555	58	24,734	274	3
DWT1242	110,196	936	50,052		52,952				65,525	153	61,654	154	80,148	545	3
DWT162	2,646	156	1,641	20	1,579				1,756	24	1,619	26	1,675	34	3
DWT193	7,761	62	5,505		4,609				7,761	62	7,761	62	7,761	62	3
DWT209	9,504	184	3,819		4,032				4,986	59	4,733	49	5,109	84	3
DWT221	9,911	187	2,225		2,154				2,477	24	2,312	22	2,479	48	3
DWT245	3,960	115	4,179		3,813				3,960	115	3,960	115	3,960	115	3
DWT307	7,827	63	8,132		8,132				7,827	63	7,827	63	7,827	63	3
DWT310	2,697	28	3,006		3,006				2,697	28	2,697	28	2,697	28	3
DWT361	5,086	50	5,075		5,060				4,973	24	4,732	24	4,741	18	3
DWT419	39,733	356	8,649		8,073				10,365	56	10,014	52	11,602	110	3
DWT503	35,916	452	15,319		15,042				19,089	90	18,478	93	25,789	248	3
DWT59	408	25	314	8	314				348	12	335	11	312	11	3
DWT592	28,806	259	11,440		10,925				16,913	63	15,614	62	28,806	259	3
DWT66	648	44	217	3	193				166	4	138	4	165	4	3
DWT72	177	12	244	8	244				177	12	177	12	177	12	3
DWT758	23,116	200	8,580		8,175				8,984	46	7,947	42	8,487	39	3
DWT87	2,250	63	696	18	682				1,064	28	1,019	26	2,250	63	3
DWT878	26,059	519	22,391		19,696				21,437	57	20,378	61	21,702	239	3
DWT918	108,358	839	23,105		20,498				26,934	86	24,502	81	25,798	84	3
DWT992	262,307	513	38,128		34,068				41,352	88	41,872	91	37,952	269	3

1: Koohestani [10].

2: Barnard, Pothen and Simon [11].

3: Koohestani [10] for RCM and GK profile, Octaviu [12] for RCM bandwidth.

The shaded cells are where the calculated new profile or bandwidth is larger than the initial corresponding value. In our calculations we reset these numbers to the corresponding initial value.