An Efficient Iterative Improvement Technique for VLSI Circuit Partitioning Using Hybrid Bucket Structures

C. K. Eem and J. W. Chong
Dept. of Electronic Engineering, Hanyang Univ., Seoul, Korea

Abstract

In this paper, we present a fast and efficient Iterative Improvement Partitioning (IIP) technique for VLSI circuits and hybrid bucket structures on its implementation. Due to their time efficiency, IIP algorithms are widely used in VLSI circuit partition. As the performance of these algorithms depends on choices of moving cells, various such methods have been proposed. In particular, the *Cluster-Removal* algorithm by S. Dutt[13][14] significantly improved partition quality. We indicate the weaknesses of previous algorithms using a uniform method for the choice of cells during improvement. To solve this problem, we propose a new IIP technique that selects the method for choice of cells according to improvement status and presents hybrid bucket structures for easy implementation.

The time complexity of the proposed algorithm is the same as the FM[3] method, and the experimental results on ACM/SIGDA benchmark circuits show improvement up to 33-44%, 45-50% and 10-12% in cutsize over FM[3], LA-3[4] and CLIP[14] respectively. Also with shorter CPU time, our thechnique outperforms Paraboli[10] and MELO[11] represented constructive partition methods by about 12% and 24%, respectively.

1. Introduction

Circuit partitioning is one of the critical issues in VLSI circuit design. For most applications, the goal is to minimize interconnections between partitions; this can be accomplished by various methods. The one such method is a *two-way partition* that recursively divides a circuit into two parts until the desired circuit complexity id reached. The *two-way partition* algorithms have been the most widely used in VLSI CAD and are classified into three types: *iterative improvement partition, constructive partition* [7][8][9][10][11] and *multilevel partition* [15]. Among others, move-based iterative improvement partition methods (IIP) have attracted most of the attention of circuit designers, due to their time efficiency and performance. In their review of IIP, Kernighan and Lin [1] proposed a graph partition algorithm known as the *KL algorithm*. Based on the algorithm, Schweikert and Kernighan [2] extended its use to hypergraphs; their algorithm can also be applied to circuit design. Fiduccia and Mattheyses [3] proposed the well-known FM algorithm to reduced time complexity. The FM algorithm uses *single-cell movement*, which operates by moving one cell at a time, and *bucket* structures instead of the *pair-wise exchange* of the previous algorithms: hence it can largely improve time efficiency with the acceptable results. Krishnamurthy [4] further enhanced the FM algorithm by adding higher level look-ahead gains, and improved the results for small circuits. To solve the *tie-breaking* of the FM based algorithm, Hagen et al. [12] presented the LIFO scheme, which somewhat improved the results. In addition to the above algorithms, numerous methods based on IIP have been suggested. However, these algorithms add little to the performance enhancement of FM. Recently, S. Dutt [14] proposed a new technique, CLIP(CLuster oriented Iterative improvement Partitioner) for the selection of the best cell by viewing cell gain as the sum of *initial* and *update* gain. In this algorithm, in order to remove clusters on the cutset in the early stage of improvement, the best cell is selected by using *updated gain*, which refers to the sum of dynamically updated gain values from the following cell move. Accordingly, the performance of IIP based algorithms using this technique has significantly improved. Most recently, J. Cong [16] presented a method for removing the *loose/stable* net on a circuit in early stages, but this algorithm suffers from the shortcoming that this bucket size may be unacceptably increased by his gain increasing function. To solve this problem, he propose a threshold value for gain. Nonetheless, their algorithm still fails to make up for its weakness; the performance of the algorithm largely depends on the threshold value.

Dutt's proposal is based on the observation that relying only on the updated part of the gain in choosing the next cell encourages neighboring cells to subsequently move, and thus establishes the locality. Dutt also note that this locality promotes closely connected components in the circuit straddling the cutline to be removed from the cutsets, and thus establishes cluster removal. As a result, among all the published IIP algorithms prior to their work, their algorithm achieves the best cutsize result with linear time complexity. However, their algorithm is not free from problems. First, relying on updated gain in choosing the best cell is an efficient way to remove clusters from the cutset in the early stage of improvement; yet after the clusters have been removed, relying on total (initial + updated) gain can occasionally obtain the better results. Secondly, choosing and moving the cell with the highest updated gain does not always get the cluster removal effect. To obtain this effect, we must take two sides into account: one is updated gain, and the other is the distribution of locked cells in the cluster. Depending on the distribution, the cluster cannot be removed from the cutset. Lastly, his algorithm is not free from frequent tie-breaking situations, though these situations occur slightly less often. In order to solve the above problem, we present two kinds of bucket structures. One is kept by total gain, and the other by updated gain. We also propose a new IIP technique that selects the method for choice of cells according to improvement status.

The rest of the paper is organized as follows. Section 2 discusses previous IIP based algorithms, and we indicate their weaknesses in using a uniform method for cell choice during improvement. Section 3 proposes a new iterative improvement technique and hybrid bucket structures, and discusses the efficiency of the proposed techniques. Section 4 presents our experimental results on ACM/SIGDA benchmark circuits. Section 5 concludes with a summary

2. Background and Related Works

2.1 Reviews of IIP algorithms

First, hyper-graph modeling is briefly introduced in order to refer to partitions on VLSI. Circuits are configured with cells and nets, and they may be represented as hyper-graph G = (V, E), with *V* as the set of cells and *E* the set of nets. Generally, two-way partition on a circuit means that this hyper-graph is divided into part sets, V_1 and V_2. At this time, it is limited that all cells of V must be included in only one part set of V_1 or V_2. *Cut* refers to the net covered between V_1 and V_2 and a net that is included in only one part set is called *uncut*. *Cutset* denotes that a net set whose state is cut and the size of this set is *cutsize*. The function of partition is to determine V_1 and V_2 that minimize cutsize at a state that satisfies the balance criterion of V_1 and V_2.

The traditional IIP based partition algorithm consists of two stages: One is initial partition stage to create a V_1 and V_2 that satisfy the balance criterion by a random function or by a specified method. And the other is an iterative improvement

stage to reduce cutsize by the partition improvement method with this initial partition. The iterative improvement stage is attained by repeating a process which is called *pass*. At the partition improvement stage, the process is carried out after making single pass until cutsize is no longer reduced. At this time, the configuration of one pass is as following Figure 1:

```
while(there exists free cells)
    c = pick the best cell;
    move and lock cell c;
    for(each net n incident to cell c)
            update gain of cells in net n;
endwhile
```

[Figure 1] Structure of One Pass in General Partition Improvement Algorithm

The performance of IIP based algorithms depends on the choice techniques of the best cell as in Figure 1. In the case of the FM method, the immediately reduced amount of the cutset obtained by the movement of one cell is defined as the *gain* of the cell and the choice of the best cell is decided from the order of current gain values. Most IIP algorithms based on FM use the best cell choice technique based on this gain. The gain of a cell, u is defined as follows:

$$\text{Gain } g(\mathbf{u}) = \sum_{n_i \in E(\mathbf{u})} c(n_i) - \sum_{n_j \in I(\mathbf{u})} c(n_j)$$

E(**u**) means a net included in the cutset among nets presently are connected to **u** now and I(**u**) refers to a net added in the cutset after **u** is moved. $C(n_i)$ and $C(n_j)$ are assigned the weight on nets i and j.

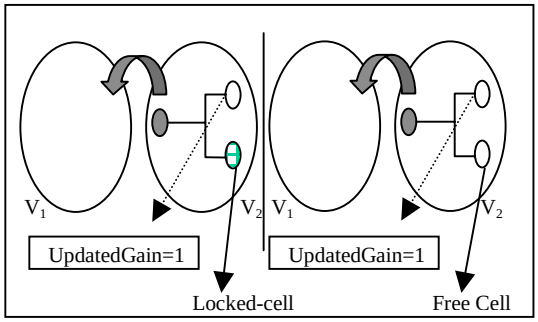
However, in the method suggested in CLIP, the gain on a cell is divied into initial gain and updated gain. Initial gain is the immediately reduced amount of cutsize depending on cell movement just after initial partition; updated gain is the amount of gain difference of the cell affected by the movement of neighbor cells. Therefore, this technique could be interpreted as follows: a cell whose updated gain is high has high potential to be dragged by a previously moved cell. The CLIP chooses and moves a cell whose dragged potential - that is, updated gain - is high, as the best cell. It can be known intuitively that in this choice technique, the movement of one cell recursively causes the movement of another cell connected to the cell. Therefore, if it is assumed that a cell **u** when improving partition moves in a cluster, the following effect may be achieved: cells in the cluster that includes **u** have priorities in movement over cells in other clusters. The technique promotes removal from the cutsets of closely connected components in the circuit that straddle the cutline, and thus establishes cluster removal. This new choice technique enhances the performance of IIP based algorithms.

2.2 Cluster Oriented Iterative Improvement Partitioner

Dutt's cell choice technique using the updated gain is quite effective in removing clusters from the cutset. Relying on updated gain, choice of the best cell is an efficient way of clusters removal from the cutset in early stage of improvement. However, after the clusters has been removed, reliance on the total (initial + updated) gain can occasionally yield the better results. Thus, continually using updated gain during the overall improving stage for partition improvement may cause problems. Accordingly, the cell choice technique to be applied should depend on the partition improvement processing status.

For example, assigning of the same updated gain could not be expected from the cluster-removal effect in the two situations as following Figure 2: it is needless to select a cell whose cluster-removal effect is lost by the cell previously moved as the best cell. As Figure 2, when previously moved cells (=Locked-cell) exists on both part sets V_1 and V_2 with a single net, the cluster removal effects are lost.

In addition, most partition improvement methods, including FM and CLIP, have a common problem called *tie-breaking* when a cell being selected and moved among many cells that satisfy the best cell condition. There is no solution for this problem now, the problem can be minimized by varying selecting conditions. In this paper, we purpose solutions to the several problems of the CLIP and to reduce the tie-breaking problem by varying these selection conditions. The proposed new hybrid bucket structures will be described in Section 3.



[Figure 2] Unreasonable Assigning Example of Updated Gain

3. Iterative Improvement Technique Using Hybrid Bucket Structures

3.1 Definition of Locked-Net

Locked-net is defined in order to refer to the partition method suggested in this paper. A Locked-net is a net having one or more *locked-cell* in each part set V_1 and V_2 divided among the nets included in a cluster. For reference, locked-cell is a cell whose movement is prohibited forcefully to other part set.

3.2 Variable Cell Choice Techniques

To utilize Dutt's updated gain is necessary for cluster-removal effect, but it can cause some problems in application as a whole partition improvement method, as mentioned above. Therefore, the proposed algorithm uses choice by updated and total gain in the overall partition improvement stage to supplement the previous choice technique.

The principle of this variable cell choice technique is as follows: when a cell that could obtain cluster-removal effect is detected during processing partition improvement, it is given a priority in movement depending on updated gain. Total gain is also applied to for cells whose cluster-removal effect is lost or that do not have the cluster-removal effect, so that cell choice techniques may be applied variably according to the processing situation during partition improvement. The more detailed description is as follows: initial and updated gain are maintained when moving each cell. Cell choice technique by updated gain is used for cells that are not connected to locked-net among cells whose updated gains are increased by the cell previously moved. The total gain technique is used for cells other than the above.

In the aspect of implementation, the buckets by updated gain(Major bucket) and total gain(Minor bucket) are maintained, and one cell is included in one bucket of both. All cells should be included in minor bucket immediately after initial partition, and the best cell is selected by total gain. When a cell moves and the updated gain of a cell affected by the movement is calculated, the bucket that will accommodate the cell is determined. Since cells that can be included in the Major bucket are limited to cells with the cluster-removal effect, they are limited to cells that are not connected to locked-net among those cells whose updated gains are increased. To achieve the cluster-removal effect, if a cell exists on major bucket, the cell is selected as the best cell first. The above method could be described as figure 3:

By using this variable cell choice technique: clearer cluster-removal effect can be achieved by first maintaining the Major bucket with priority, and then accommodating cells whose updated gain is increased on the bucket. Because CLIP also reflects this reduced amount for a cell whose updated gain is

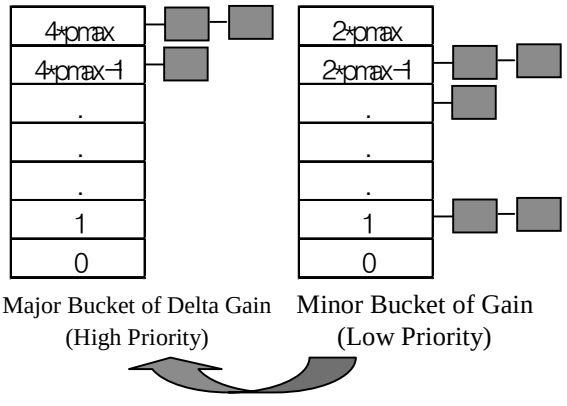
decreased, but it is not related to obtain cluster-removal effect. That is, the cluster-removal effect is obtained when encouraging cell movement in cluster located on cutset regardless of total gain. Accordingly, the decrease of updated gain serves only to limit movement, but have no role in promoting it. It means that since the cluster-removal effect is reflected in the increased amount of updated gain, the decreased amount should not be reflected on cell choice for removing clusters. The increase in updated gain by locked-nets does not obtain the cluster-removal effect, so it is excluded from subject selected with priority.

```
while(there exists free cells)
   c = pick cell with maxmum-
       upated gain from Major bucket;
   if(c == NULL)
       c = pick cell with maxmum-
           total gain from Minor bucket;
   move and lock cell c;
   for(each net n incident to cell c)
       update gain of cells in net n
       and move the cell to major bucket
       if (the cell is updated gain > 0 and not in Locked-net);
endwhile
```

[Figure 3] Structure of the One Pass of the Proposed Algorithm

With using these methods, it operates as follows. Only when there is a cluster-removal effect, that is, a cell exists on a Major bucket, is cell choice technique by updated gain used and since most cells exist in Minor bucket other than in the former case, cell selection by general total gain can be used.

Figure 4 shows the Major/Minor bucket structure in applying the proposed cell choice technique. The Major bucket is $4*p_{max}$ in size, and the Minor bucket is $2*p_{max}$ in size; and at that time, p_{max} is the maximum of the sum of net weight values connected to a specific cell in a given circuit. This bucket maintenance is very effective in decreasing tie-breaking, because the frequency of tie-breaking can be reduced proportionally to the buckets area. In case of the Major bucket, if tie-breaking occurs, a cell can be selected by total gain, and in the case of Minor bucket, it can be selected by updated gain. In this case, if tie-breaking occurs, since secondary selecting conditions are proportional (Major bucket) or inversely proportional(Minor bucket) to the initial gain, the best cell can be selected without additional calculation.



[Figure 4] Structure of Suggested Hybrid Bucket

4. Experimental Results

As shown in Figure 3, the proposed algorithm has the same complexity as the FM and CLIP. When the updated gain is calculated, the proposed algorithm needed only a few additional steps to assign the cell to the bucket according to the conditions which could be immediately determined. Although some calculation was necessary to keep the two kinds of buckets, the calculation hardly affects the performance.

The proposed algorithm was experimented on ACM/SIGDA 21 benchmark circuits. The experimental results using the FM, LA-3(Look Ahead level 3) and CLIP applied to the FM are represented in Table 1. They were compared with results using the proposed algorithm (HYIP) applied to the FM. The experimental results showed the performance of the each IIP techniques with the same initial partition. As shown in the results of experiment, HYIP improves up to 33-44% and 45-50% respectively in cutsize over the FM and LA-3. In addition, it reduced cutsize by 10-12% compared with the CLIP algorithm that achieved the best cutsize result with linear time complexity among all the previously published IIP algorithms. In particular, we need to note that lager the circuit sizes yield better results.

In Table 2, we compare the results of the proposed algorithm with the results of the Paraboli and MELO represented constructive methods. We executed the HYIP, FM and CLIP 100 times; despite these numerous executions at faster CPU times, the HYIP outperforms Paraboli and MELO by about 12% and 24%, respectively.

Finally, the CPU execution times for the experiments above shown are given in Table 3. The Paraboli method was performed on a DEC 3000 Model 500 AXP; the MELO and HYIP were performed on a SUN SPARC 10, and the remainder on a SUN SPARC Model 85. Although these were executed at different machines, it is reasonable to assume that these machines perform similarly. As shown in Table 3, the execution time for the proposed algorithm is less than in existing constructive methods, and its speed compares favorably to that of FM and CLIP of existing IIP based methods.

5. Conclusions

This paper presents a fast and efficient Iterative Improvement Partitioning (IIP) technique for VLSI circuits and hybrid bucket structures on its implementation. Although relying on the updated gain in choosing the best cell efficiently removes clusters from the cutset in the early stage of the improvement, after the clusters have been removed, relying on total gain occasionally achieves the better results. To supplement the previous choice technique, we propose the hybrid bucket structures, and varying cell choice techniques may be applied according to the processing situation of partition improvement. We also point out that choosing and moving of cells with the highest updated gain do not always get the cluster removal effect. To obtain this effect, we must take into account the two sides: updated gain and the distribution of locked cells on the clusters. We also discuss that the proposed hybrid bucket structure is very effective in decreasing tie-breaking, because such frequency could be reduced proportionally in the buckets area.

[References]

[1] B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs”, Bell System Tech. Journal, vol. 49, Feb. 1970, pp. 291-307.

[2] D. G. Schweikert and B. W. Kernighan, “A Proper Model for the Partitioning of Electrical Circuits”, Proc. 9th Design automation workshop, 1972, pp. 57-62.

[3] C. M. Fiduccia and R. M. Mattheyses, “A linear-time heuristic for improving network partitions”, Proc. ACM/IEEE Design Automation Conf., 1982, pp. 175-181.

[4] B. Krishnamurthy, “An improved min-cut algorithm for partitioning VLSI networks”, IEEE Trans. on Comput., vol. C-33, May 1984, pp. 438-446.

[5] Y. C. Wei and C. K. Cheng, “Towards efficient hierarchical designs by ratio cut partitioning”, Proc. Int’l. Conf. Computer-Aided Design, 1989, pp. 298-301.

[6] Y. C. Wei and C. K. Cheng, “An Improved Two-way Partitioning Algorithm with Stable Performance”, IEEE Trans. on Computer-Aided Design, 1990, pp. 1502-1511.

[7] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", Proc. Int'l. Conf. Computer-Aided Design, 1991, pp. 10-13.
[8] J. Cong and M. Smith, "A bottom-up clustering algorithm with applications to circuit partitioning in VLSI designs", Proc. ACM/IEEE Design Automation Conf., 1993, pp. 755-760.
[9] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering", Proc. Int'l. Conf. Computer-Aided Design, 1994, pp. 63-67.
[10] B. M. Riess, K. Doll and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", Proc. ACM/IEEE Design Automation Conf., 1994, pp. 646-651.
[11] C. J. Alpert and S-Z Yao, "Spectral Partitioning: The More Eigenvectors, the Better", Proc. ACM/IEEE Design Automation Conf., 1995.
[12] L. W. Hagen, D. J. Hwang and A. B. Kahang, "On implementation choices for iterative improvement partitioning methods", Proc. European Design Automation Conf., Sept. 1995, pp. 144-149
[13] S. Dutt and W. Deng, "A Probability-Based Approach to VLSI Circuit Partitioning", Proc. ACM/IEEE Design Automation Conf., 1996, pp. 100-105.
[14] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", Proc. Int'l Conf. Computer-Aided Design, 1996, pp. 194-200.
[15] C. J. Alpert, J-H Huang and A. B. Kahng, "Multilevel Circuit Partitioning", Proc. ACM/IEEE Design Automation Conf., 1997, pp. 530-533.
[16] J. Cong, H. P. Li, S. K. Lim, T. Shibuya and D. Xu, "Large Scale Circuit Partitioning With Loose/Stable Net Removal And Signal Flow Based Clustering", Proc. Int'l Conf. Computer-Aided Design, 1997, pp. 441-446.

[Table 1] Comparison of HYIP and other IIP based algorithms

Circuit	Minimum Cut Size of 20 Runs							Average Cut Size of 20 Runs						
	Cut Size				Improvement(%)			Cut Size				Improvement(%)		
	FM	LA-3	CLIP FM	HYIP FM	HYIP (FM)	HYIP (LA)	HYIP (CLIP)	FM	LA-3	CLIP FM	HYIP FM	HYIP (FM)	HYIP (LA)	HYIP (CLIP)
P1	47	52	52	47	0.0	9.6	9.6	74.9	68.5	65.8	60.6	19.1	11.5	7.9
Bm1	54	53	49	47	13.0	11.3	4.1	79.5	67.5	65.0	58.3	26.7	13.6	10.3
t4	87	82	56	54	37.9	34.2	3.6	129.3	117.2	77.0	66.6	48.5	43.2	13.5
t3	75	80	57	58	22.7	27.5	- 1.7	106.8	106.2	72.3	65.4	38.8	38.4	9.5
t2	149	126	89	90	40.0	28.6	- 1.1	182.1	148.1	105.2	102.8	43.6	30.6	2.3
t6	67	70	60	63	6.0	10.0	- 4.8	94.2	84.4	70.1	73.3	22.2	13.2	-4.4
struct	46	44	37	33	28.3	25.0	10.8	58.0	49.6	45.6	41.6	28.3	16.1	8.8
t5	127	99	75	75	41.0	24.2	0.0	183.6	165.0	89.0	85.6	53.4	48.1	3.8
19ks	140	130	119	107	23.6	17.7	10.1	171.7	169.0	150.3	128.1	25.4	24.2	14.8
p2	212	149	149	143	32.5	4.0	4.0	273.9	233.4	233.2	211.6	22.8	9.3	9.3
s9324	59	43	49	45	23.7	- 4.4	8.2	84.7	81.1	89.5	66.1	22.0	18.5	26.2
biomed	83	90	84	84	- 1.2	6.7	0.0	117.4	170.7	108.4	98.9	15.8	42.1	8.8
s13207	98	85	98	70	28.6	17.7	28.6	122.6	118.9	123.5	101.6	17.1	14.6	17.7
s15850	109	87	80	85	22.0	2.3	- 5.9	176.9	140.4	140.9	121.0	31.6	13.8	14.1
ind2	264	422	260	185	29.9	56.2	28.9	627.5	732.4	369.6	298.9	52.4	59.2	19.1
ind3	272	504	261	241	11.4	52.2	7.7	506.0	758.0	376.6	338.7	33.1	55.3	10.1
s35932	85	168	102	89	- 4.5	47.0	12.8	210.3	231.4	127.1	123.6	41.2	46.6	2.8
s38584	100	85	49	52	48.0	38.8	- 5.8	299.8	271.2	83.2	88.5	70.5	67.4	-6.0
avq.sml	347	608	223	187	46.1	69.2	16.1	578.6	815.5	335.1	274.2	52.6	66.4	18.2
s38417	240	284	78	75	68.8	73.6	3.9	384.1	408.2	136.7	112.1	70.8	72.5	18.0
avq.lar	350	398	216	184	47.4	53.8	14.8	755.0	693.4	305.2	272.3	63.9	60.7	10.8
Total	3011	3659	2243	2014	33.1	45.0	10.2	5217	5630	3169	2789	46.5	50.4	12.0
Average of % Improvement					26.4	28.8	6.9					38.6	36.4	10.3

[Table 2] Comparison of HYIP and other constructive algorithms

Circuit	Cut Size				
	Paraboli	MELO	FM	CLIP FM	HYIP FM
P1	53	64	47	47	47
bm1		48	49	47	47
t4		61	80	53	53
t3		60	62	56	56
t2		109	124	87	88
t6		90	60	60	60
struct	40	38	41	33	32
t5		102	104	74	72
19ks		119	130	109	105
p2	146	169	182	148	143
s9324	74	79	51	44	45
biomed	135	115	83	83	83
s13207	91	104	78	76	70
s15850	91	52	104	75	70
ind2	193	319	264	174	185
ind3	267		263	241	241
s35932	62		85	83	58
s38584	55		63	47	47
avq.sml	224		297	200	177
s38417	49		147	66	63
avq.lar	139		350	185	169
Total	1619	1529	2664	1988	1430 1156 1911
Improv -ement	*	*			11.7(%) 24.4(%)

[Table 3] Comparison of CPU times of various algorithms

Circuit	CPU Times				
	Paraboli	MELO	FM (x100)	CLIP FM (x100)	HYIP FM (x100)
p1	18.3	8	0.33	0.44	0.18
bm1		9	0.27	0.43	0.20
t4		24	0.58	0.90	0.50
t3		27	0.76	0.96	0.50
t2		29	0.58	0.99	0.58
t6		31	0.57	0.88	0.52
struct	35.2	38	0.54	0.69	0.35
t5		67	1.13	1.62	1.09
19ks		79	1.59	2.23	1.03
p2	137.4	89	1.81	2.37	1.38
s9324	490.3	516	2.78	3.14	2.49
biomed	710.9	496	3.89	3.34	3.97
s13207	2060.4	710	4.23	5.01	5.11
s15850	2730.9	1197	4.24	6.87	6.96
ind2	1367.3	1855	9.10	13.17	17.02
ind3	760.7		11.07	16.35	19.40
s35932	2626.7		11.82	13.21	15.37
s38584	6517.5		13.70	15.73	19.37
avq.sml	4098.9		18.44	20.50	23.40
s38417	2041.5		15.36	17.70	21.71
avq.lar	4135.0		19.49	24.07	28.00
Total	27731	5175	12228	15060	16471 4188 16913