Timing-Driven Global Routing with Efficient Buffer Insertion

Jingyu Xu, Xianlong Hong, Tong Jing
Dept. of Computer Science & Technology, Tsinghua Univ., Beijing, P. R. China

Abstract -- **Timing optimization is an important goal of global routing in deep submicron era. To guarantee the timing performance of the circuit, merely adopting topology optimization becomes inadequate. In this paper, we present an efficient timing-driven global routing algorithm with buffer insertion. Our approach is capable of applying topological-based timing optimization and buffer insertion simultaneously with routablity considerations. Compared with previous works, we efficiently solve the timing issues under a limited buffer usage. The experimental results have demonstrated significant delay improvement within short runtime with very small number of buffers inserted.**

I. INTRODUCTION

Global routing is an important stage in VLSI physical design. In the past, congestion was the major concern in global routing. With the exponential reduction in scaling of feature size, higher performance design brings substantial advantage over the competition and interconnect becomes a performance bottleneck. In many systems designed today, as much as 50% of the clock cycle is consumed by interconnect delay. As technology advances scale device dimensions, the significance of interconnect delay is expected to increase further in the future. Therefore, timing optimization becomes a crucial task of global routing in order to maximize the overall chip performance.

To deal with this trend, many helpful researches have been performed on timing-driven global routing[1-3]. Most of these works are based on topology optimization. In deep submicron era, with the dramatic increase in chip density, the connections between the component modules constitute a directed acyclic graph, and signals need to meet a large number of timing constraints. Merely adopting topology optimization technique becomes inadequate.

Buffer insertion is an effective technique for reducing interconnect delay in both theory and practice. Several works studied delay-driven buffer insertion for 2-pin nets [4-5]. For buffer placement in distributed RC trees, Van Ginneken [6] proposed a classic dynamic programming (DP) algorithm. It has since been generalized to other applications---low power [7], wire segmenting[8], noise optimization[9], buffered Steiner tree construction, etc. Most of these previous works targeted earlier design stages such as floorplanning and placement, where a large number of buffer resources are available and the final buffer solution is always assumed to be feasible for later design phases. Therefore, such approaches focus on finding the ideal buffer location precisely with a considerable number of buffers used.

Since global routing is performed on fixed placement, a large buffer usage is likely to bring disadvantages: first, it adds to the difficulty of ECO placement for realizing those buffer locations; second, with routablity considerations, it is hard to achieve routablity/timing solution convergence because lots of routing resources being occupied by buffers increases congestion. Further, most earlier approaches either operate on individual routing trees, or are time consuming to achieve a feasible global buffering.

Facing these problems, the designers naturally desire a buffer solution consuming less buffer resources while efficiently solves the timing issues. In this paper, we present an efficient global routing algorithm with buffer insertion. We conduct global buffering in two iterative steps: selecting timing-critical nets and finding buffer location for each single net. In the first step, we compute timing/congestion information and find nets which have greatest impact on circuit timing, while the routing tree selected for buffer insertion is of minimal cost of routablity deterioration. In the second step, we restrict one net to be inserted with one buffer such that buffers are distributed to as many critical nets to benefit as many delay paths as possible in the network. We then derive some new properties to explore the optimal location for single buffer insertion for a multi-sink RC tree structure. Our work has the following contributions.

- Different from existing ones, our approach is capable of applying topological-based timing optimization and buffer insertion simultaneously with routablity considerations.
- Our approach handles nets simultaneously and reduces buffer usage greatly. Taking circuit timing as a whole, our approach can lead to maximal delay/congestion tradeoff with minimal number of buffers inserted.

Our experiments have demonstrated significant delay improvement within short runtime. A network of about 20000 multi-pin nets takes less than 100 seconds to complete the routablity/timing optimization with buffer insertion.

The remainder of this paper is organized as follows. Section 2 introduces the delay models and defines the problem. Section 3 presents theoretical properties that determine the appropriate location for buffer insertion. Section 4 describes our global routing algorithm with efficient buffer insertion. We present experimental results in Section 5 and summarize in Section 6.

II. PRELIMINARIES

A. Delay Models

The analytical Elmore delay and Sakurai's heuristic delay formula [10] have been widely used in delay estimation. In Elmore delay, the basic model for wiring is modeled as a voltage source with the on-resistance of the transistor R_s, distributed RC lines of resistance r_e, capacitance c_e, and loading capacitance C_z. Sakurai [10] also gave delay calculations for the distributed RC line. Rewrite these delay forms into a uniform expression. We have,

$$T_{DZ} = \beta R_s(c_e + C_z) + \alpha r_e c_e + \beta r_e C_z \qquad (1)$$

For 63.2% threshold Elmore delay, α =0.5 and β = 1.0; for 90% threshold Elmore delay, α =1.15 and β = 2.3; for 90% threshold Sakurai delay, α =1.02 and β = 2.21. As in most previous work, we

* This work was supported in part by the NSFC under Grant No.60373012, the SRFDP of China under Grant No.20020003008, and the Hi-Tech Research and Development (863) Program of China under Grant No.2004AA1Z1050.

use the RC model for buffers. Buffer consists of three element, intrinsic delay d_b, output resistance r_b and input capacitance c_b.

B. Problem Formulation

In global routing graph (GRG)[3] $G = (V, E)$, each GRG edge is associated with a number called edge capacity, which indicates the available tracks between two adjacent vertices. Each output port p_i has a required arrival time $RAT(p_i)$. For the circuit to function properly, we must have $Delay(p_i) \le RAT(p_i)$ for every $p_i \in PO$, where PO is the set of circuit output ports and $Delay(p_i)$ denotes the delay of p_i from corresponding input port of the circuit. Thus, the timing-driven global routing problem can be formulated as follows.

Minimize $\max_{pi\in PO}(Delay(p_i)-RAT(p_i))$

Subject to $f_j = \sum_{i=1}^{N_n} u_{ij} \le c_j$

$M_{buffer} \le AV_{buffer}$

where $u_{ij} = \begin{cases} 1, when\ net\ n_i\ passes\ edge\ e_j \\ 0, elsewise \end{cases}$

Let N_n be the number of nets within a design and f_j the total demand of the nets using edge e_j. f_j should be no greater than the edge capacity c_j. M_{buffer} denotes the number of buffers inserted and AV_{buffer} is the number of available buffers. In practice, AV_{buffer} is usually between 10%-20% of the number of the cells.

III. Optimal Buffer Insertion for Single Net

To minimize the buffer usage, we conduct global buffering in two iterative steps. In this section, we first try to find the optimal buffer location for a given signal net under one buffer restriction. In section 4, we then discuss how to obtain a reasonable global buffering with maximal delay/congestion tradeoff.

A. Buffer Insertion for 2-Pin Nets

Theorem 1 Given a 2-pin net with source s and sink t connected by a single wire, the optimal location for the placement of a buffer b on the wire is at distance $x = \frac{\beta r(C_t - c_b) - \beta c(R_s - r_b) + 2\alpha rcl}{4\alpha rc}$ from the source.

We omit the proof due to space limitation.

Corollary 1 Given a 2-pin net with source s and sink t connected by a single wire and a buffer b that $C_t=c_b$ and $R_s=r_b$, it is worthwhile to insert b if and only if $l > \sqrt{\frac{2(\beta R_s c_b + d_b)}{\alpha rc}}$. The optimal location for insertion is at distance $x=l/2$ from the source.

Corollary 1 can be obtained by applying Theorem 1 with $C_t=c_b$ and $R_s=r_b$. For 63.2% threshold Elmore delay, C. Alpert *et al* [8] has proved that the threshold length is $l > 2\sqrt{\frac{R_s c_b + d_b}{rc}}$. Substituting α=0.5 and β= 1.0 into Corollary 1, the result exactly corresponds with their solution.

B. Buffer Insertion for Multi-Sink Nets

Previous theoretical results for optimal buffer insertion are based on 2-pin wires. Van Ginneken's algorithm can be used to compute buffer solution for multi-sink RC trees, while the DP formulation is quadratic in time and space usage and limited to inserting buffers at tree vertices. We derive new properties to explore the optimal buffer insertion for a multi-sink RC tree. Although these analytical results target the case of "one net one buffer", they are very helpful in finding provably good buffer solution under small buffer usage. Fig.1(a) illustrates the case, where T denotes the buffer location.

To tackle the optimal location, we transform the tree structure

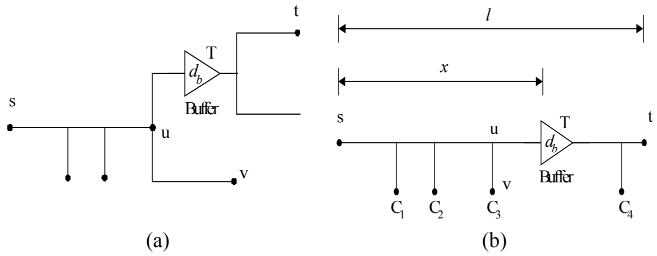

Figure 1. Insert a buffer into a multi-sink net (a) The original tree structure, (b) Transform the tree structure into a multi-branched path

into a branched path structure from source s to the critical sink t (Fig.1(b)). We denote the total capacitance of each branch on the path as C_1, C_2, ... ,C_n with coordinates x_1, x_2, ..., x_n. Let C_{ll} be the summation of branch capacitances after insertion point T and load capacitance of critical sink C_t. Then the load capacitance of T is given by $C_T = (l-x)c + C_{ll}$, where l is the length of the s-t path.

Theorem 2 Given the tree structure of a multi-sink net with source s and critical sink t, the optimal location for the placement of a buffer b on the tree is on the path from s to t at distance $x = \frac{r(lc + C_{ll} - c_b) - (R_s - r_b)c}{2rc}$ from s, where $C_{ll} = \sum_{x<x_i} C_i + C_t$, or on a branching point u of branches incident on the path from s to t which yields maximum $\beta R_s(C_u - c_b) + \sum_{xy\in path(s,u)} \beta rl_{xy}(C_u - c_b)$, where C_u is the total capacitance of the branch.

Proof: we conduct the proof in two steps. In step1, we prove that the optimal location should be either on the path from s to t defined above, or on one of the branching points to decouple the branch. Then, we prove the optimal location given by Theorem2.

Step1. We partition all the nodes of the tree into three groups according to their locations and analyze the impact of buffer on their delays respectively.

Group1: nodes that are in the upstream of insertion point T, and not on the path from s to t, such as node v in Fig.1(b). We first find the intersection of branch containing v with the s-t path and denote it as u' (in Fig.1(b) $u = u'$). According to (1), after buffer is inserted, the terms that can be eliminated from the delay expression of v are:

$$\Delta_+ = \beta R_s C_T + \sum_{xy\in path(s,u')} \beta rl_{xy} C_T \tag{2}$$

The terms that should be added are:

$$\Delta_- = \beta R_s c_b + \sum_{xy\in path(s,u')} \beta rl_{xy} c_b \tag{3}$$

Group2: nodes that are in the upstream of insertion point T, and on the path from s to t, such as node u in Fig.1(b). According to (1), the delay variation of node u is also given by (2) and (3).

Group3: nodes that are in the downstream of insertion point T, such as the critical sink t in Fig.1(b). After buffer is inserted, the terms that can be eliminated from the delay expression of t are:

$$\Delta_+ = \beta R_s C_T + \sum_{xy\in path(s,T)} \beta rl_{xy} C_T \tag{4}$$

The terms that should be added are:

$$\Delta_- = \beta R_s c_b + \beta r_b C_T + d_b + \sum_{xy\in path(s,T)} \beta rl_{xy} c_b \tag{5}$$

Suppose the optimal buffer location for critical sink is on the branch other than on the path. We illustrate the case in Fig.1(b) by taking v as critical sink instead of t, so that the path from s to v is the main path. After a buffer is inserted on the branch, the delay variation of v is given by (2) and (3). Since u' is fixed, $\Delta\_$ is a constant. Only C_T in (2) is a variable. To maximize (2), we should maximize C_T. It is clear that the insertion point T should be moved to u, which is on the branching point. From (2) and (3), the maximum delay reduction is $\beta R_s(C_u - c_b) + \sum_{xy\in path(s,u)} \beta r l_{xy}(C_u - c_b)$, where C_u is the total capacitance of the branch. Thus we can determine the optimal branching point for buffer placement.

Step2. Suppose the delay of sink t before and after buffer insertion is D and D', respectively. According to (4) and (5),

$$D - D' = \left[\beta(R_s - r_b) + \sum_{xy\in path(s,T)} \beta r l_{xy}\right]\left[(l-x)c + C_{ll}\right] - (\beta R_s c_b + d_b + \sum_{xy\in path(s,T)} \beta r l_{xy} c_b) > 0 \quad (6)$$

Substituting $x = \sum_{xy\in path(s,T)} l_{xy}$ into (6), setting the derivative of $D\text{-}D'$ with respect to x to 0 and solving for x yields

$$\frac{d(D-D')}{dx} = 0 \Rightarrow x = \frac{r(lc + C_{ll} - c_b) - (R_s - r_b)c}{2rc} \quad (7)$$

For each sub region $[x_i, x_{i+1}]$ bounded by two adjacent branches, the candidate optimal x can be obtained by (7)(if x does not fall into the region, assign one of the two terminals x_i, x_{i+1} that is nearer to x as the candidate location). The one yielding maximum delay reduction on the s-t path is then compared with the optimal branching point to determine the final location for insertion.

IV. Global Routing with Efficient Buffer Insertion

In global interconnect, simply buffering each routing tree will be extremely wasteful in terms of buffers used. To characterize the timing of the whole circuit to determine where and how to do buffer insertion remains a complicated problem. In this section, we first review the critical network concept[11] which has shown to be efficient in characterizing the most crucial parts of the circuit. Following this concept, we then present a global routing algorithm with efficient buffer insertion.

A. *Review of the Critical Network Concept*

In a circuit, the timing arcs between the component modules constitute a directed acyclic graph. If we treat all the input and output pins of cells as vertices and timing arcs between them as edges, with a virtual primary source s and a sink t added, the transmission network can be represented by a 5-tuple $N=(V, E, w, s, t)$, where edge weight w denotes the delay of the arc. Each vertex k in the network has a required arrival time $RAT(k)$ and an actual arrival time $t_E(k)$. Given a vertex $i\in V$, i is called a critical vertex if $t_E(i) > RAT(i)$. Given an edge $(i,k)\in E$, (i,k) is called a critical edge if $t_E(i) + delay(i,k) > RAT(k)$. T_L is the maximum timing constraint of the circuit, while the actual arrival time T_E of the primary sink determines the circuit speed. For output ports with different timing constraints, the weight of edge connecting the primary sink and each output port can be defined.

If $T_E > T_L$, there exist a network $N' = (V', E', w, s, t)$ consisting of and only consisting of critical vertices and critical edges. N' is called a *critical network*, where V' is the set of critical vertices and E' is the set of critical edges. Critical network has following advantages. First, it precisely describes the parts of the circuit that have crucial impact on circuit delay, avoiding the combinational explosion problem caused by enumeration of all the delay paths. Second, critical network dynamically reflects the change of timing criticality to guide optimization such as rip-up and buffer insertion.

B. *The Buffer-Insertion Global Routing Algorithm*

To achieve a feasible global buffering, we want to place less buffers to gain as much delay reduction as possible. According to Max-flow Min-cut theorem[12], given a cut C of a network, every directed path from primary source s to primary sink t passes through at least one edge in C. While in the critical network, every edge on the cut turns out to be a critical edge. Therefore reducing the delay of every edge on a cut leads to an overall delay reduction of circuit. To take account of routablity, we modify the edge weight w to be a delay improvement cost function and construct a new critical network $N'' = (V', E', w', s, t)$. Edge weight w' indicates the cost of routablity deterioration per unit delay improvement. Clearly, a min-cut of N'' corresponds to a set of nets that have maximal delay/congestion tradeoff after rerouting. To guarantee maximum delay improvement of these nets, we compare the delay performance of every historical routing tree obtained during congestion optimization to determine an optimal delay tree for each net in timing optimization.

The algorithm takes three major operations to evaluate the delay improvement cost for $\forall e\in E'$: 1) compute the routablity deterioration cost 2) determine the optimal delay tree 3) compute delay improvement cost.

(1) Routablity Deterioration Cost

Given a net that contains one or more edges on the cut of N', a candidate solution set S is built and inserted with all the historical routing trees ever obtained by the net. For each tree in S, we evaluate its congestion impact on the overall solution as corresponding routablity deterioration cost. For each GRG edge s_i that the tree passes, the increase of congestion overflow after adding a unit edge usage is computed. The summation of overflow increase on all the edges passed by the tree turns out to be the routablity deterioration cost.

(2) Determine Optimal Delay Tree

For each tree in S, we compute the number of critical and non-critical sinks and sort them by negative slack from timing constraints violation. We compare the delay performance of candidate trees in terms of their critical sink delays. If delay performance of two candidates is similar, non-critical sinks and the congestion cost are then compared.

(3) Delay Improvement Cost

Suppose reroute a net with its optimal delay tree leads to delay reduction Δd($\Delta d>0$ indicates a delay improvement), and congestion reduction Δc($\Delta c>0$ indicates a routablity improvement). We evaluate the delay improvement cost for each $e\in E'$ as follows.

i) if current routing tree is the optimal delay tree, set $w' = 1$, which

```
ALGORITHM BufferInsertion-GR
1.  Evaluate timing and routability;
2.  WHILE congested OR timing constraints violated DO
3.          Congestion optimization;
4.          Update timing information based on transmission queues;
5.          WHILE timing constraints violated DO
6.                  Construct critical network N'=(V', E', w, s, t);
7.                  FOR each edge e∈ E' DO
8.                          Find net i containing edge e;
10.                         Determine the optimal delay tree $T_i$ of net i;
11.                         Set the improvement weight of the edge w';
12.                 ENDFOR
13.                 N''=(V', E', w', s, t);
14.                 calculate the maximum flow and min-cut of N'';
15.                 FOR each edge e on the min-cut DO
16.                         Find net i containing edge e;
17.                         Reroute net i with optimal delay tree $T_i$;
18.                         Insert a buffer on $T_i$;
19                          Num_buffer = Num_buffer +1;
20.                 ENDFOR
21.         ENDWHILE
22. ENDWHILE
ENDPROC
```

Figure 2. The global routing algorithm with Buffer Insertion

TABLE. 1 BENCHMARK DATA

Testcase	#Nets	#Cells	#Grids
C2	745	590	9*11
C5	1764	1586	16*18
C7	2356	2150	16*18
S13207	4953	4267	24*26
AVQ	21851	22119	65*67

TABLE. 2 PARAMETER LIST

	Description	Value
r	Wire resistance per unit length(Ω/μm)	0.075
c	Wire capacitance per unit length(fF/μm)	0.118
d_b	Intrinsic buffer delay(ps)	36.4
c_b	Buffer capacitance(fF)	8.0
r_b	Buffer output resistance(Ω)	200
α	Parameter in delay expression	1.02
β	Parameter in delay expression	2.21

means that simply rerouting will not improve delay whereas buffer insertion is applicable.

ii) if $\Delta d>0$ and $\Delta c>0$, set $w' = 0$, which means that rerouting will improve timing and routablity simultaneously. if $0<\Delta d<\delta$, where δ is a predefined small constant, a random disturbing is applied to choose between current routing tree and optimal delay tree.

iii) if $\Delta d>0$ and $\Delta c<0$, set $w' = \Delta c^2/\Delta d$, which means that rerouting improves delay at the expense of routablity deterioration.

After above operations, we construct a new critical network $N'' = (V', E', w', s, t)$ and compute the min-cut of N''. For each edge on the min-cut, we find corresponding net and reroute it with optimal delay tree. Buffers are then inserted based on the theoretical results presented in section 3. We restrict one net to be inserted with one buffer such that buffers are distributed to as many nets to benefit as many sinks as possible in the network. The description of the algorithm is given in Fig.2.

V. EXPERIMENTAL RESULTS

We have implemented the timing-driven global routing algorithm in C language and tested it on a Sun Enterprise 450 workstation. The MCNC (Microelectronics Center of North Carolina) benchmarks are used in the experiments. Table.1 summarizes the benchmark data sets. The parameters(Table.2) are based on 0.18 μm technology in [13]. The experiments compare the circuit delay performance and running time of our algorithm with the method proposed by [11], which adopts topological optimization based on critical network concept. We will see that with our efficient buffer insertion scheme, significant delay improvement and runtime speedups can be achieved under very small buffer usage. Table.3 shows the performance of the two algorithms for 5 test cases. "TGR" indicates the solution obtained by using method in [11] that applies topological optimization merely, "buffer" indicates our global routing algorithm.

The test results are record by: 1) Max violation: the maximum timing violation among the output ports of the circuit, or equivalently, the maximum negative slack; 2) #buffer: the number of buffers inserted by our algorithm; 3) #buffer / #cell: the ratio of the number of buffers inserted to the total number of cells; 4) Overflow edges: the number of congested GRG edges; 5) Runtime; 6) Wire length and 7) Wire length off: the wire length comparison of the two algorithms.

We can see from Table.3 that our algorithm is capable of dilivering a substantial delay reduction to satisfy the timing constaints successfully. The number of buffers used is small with respect to the number of cells. For large scale circuits, our algorithm achieves extremely high tradeoff between the delay reduction and buffer usage. The running time comparison of the two algorithms is also given in Table.3. It is clear from the table that our algorithm achieves speedups over method TGR.

Comparison on total wire length are given in the last two columns of Table.3. We can see that the wire length performance of the two algorithms is comparable. In some of the test cases, our algorithm even achieves better wire length than TGR.

VI. CONCLUSIONS

In this paper, we propose a timing-driven global routing algorithm based on critical network concept to obtain efficient global buffer insertion. Our approach is capable of applying topological-based timing optimization and efficient buffer insertion simultaneously with routablity considerations. The experiments have demonstrated significant delay improvement within short runtime with very small number of buffers inserted.

REFERENCE

[1] X. L. Hong, T. X. Xue, J. Huang, C. K. Cheng, E. S. kuh, "TIGER: An Efficient Timing-Driven Global Router for Gate Array and Standard Cell Layout Design", IEEE Trans. on CAD, 16(11): 1323-1330, 1997.

[2] J. Hu, S. S. Sapatnekar, "A Timing-constrained Algorithm for Simultaneous Global Routing of Multiple Nets", In: Proc. of IEEE/ACM ICCAD, San Jose, CA, pp.99-103, 2000.

[3] T. Jing, X. L. Hong, J. Y. Xu, H. Y. Bao, C. K. Cheng, J. Gu, "UTACO: A Unified Timing and Congestion Optimization Algorithm for Standard Cell Global Routing", IEEE Trans. on CAD, 2004, 23(3): 358-365.

[4] H. Zhou, D. F. Wong, I.-M. Liu, and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer locations", in Proc. ACM/IEEE DAC, 1999, pp. 96–99.

[5] M. Lai and D. F. Wong, "Maze routing with buffer insertion and wiresizing", in: Proc. ACM/IEEE DAC, 2000, pp. 374–378.

[6] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," in Proc. Int. Symposium on Circuits and Systems, pp. 865-868, 1990.

[7] J. Lillis, C. K. Cheng and T. -T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model", IEEE Trans. Solid-State Circuits, 31(3), 1996, 437-447.

[8] C. Alpert and A. Devgan, "Wire Segmenting for Improved Buffer Insertion," in: Proc. ACM/IEEE DAC, pp. 588-589, 1997.

[9] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion for noise and delay optimization", in: Proc. DAC, 1998, pp. 362–367.

[10] Sacurai, T., "Approximation of Wiring Delay in MOSFET LSI", IEEE J. of Solid-State Circuits, 1983, 18(4): 418-426.

[11] T. Jing, X. L. Hong, H. Y. Bao et al "A Novel and Efficient Timing-Driven Global Router for Standard Cell Layout Design Based on Critical Network Concept" in: Proc. IEEE ISCAS'02, I 165-I 168.

[12] L. R. Ford, Jr. and D. R. Fulkerson. Flows in Networks. Princeton University Press, 1962.

[13] Semiconductor Industry Association, National Technology Roadmap for Semiconductors. San Jose, CA: SIA, 1997.

TABLE. 3 COMPARISON ON DELAY, ROUTABLITY AND RUNTIME PERFORMANCE

Test case	Max Violation(*ns*)		#Buffer	%#Buffer /#Cell	Overflow Edges		Runtime (s)		Wire Length(μm)		%Wire Length Off
	TGR	*Buffer*			*TGR*	*Buffer*	*TGR*	*Buffer*	*TGR*	*Buffer*	
C2	-0.6135	0.0672	42	7.1%	0	2	11.44	3.85	47143	47154	0.02%
C5	-0.3082	0.4154	36	2.3%	1	0	27.69	11.84	132423	134120	1.28%
C7	-2.2618	0.0929	76	3.5%	0	1	323.60	94.23	157106	157081	-0.02%
S13207	-0.1711	0.2775	89	2.1%	0	0	53.87	45.13	1031458	1032265	0.08%
Avq	-3.3831	0.9034	36	0.2%	2	0	157.34	67.57	1369619	1343240	-1.93%