Additional experiments for rebuttal

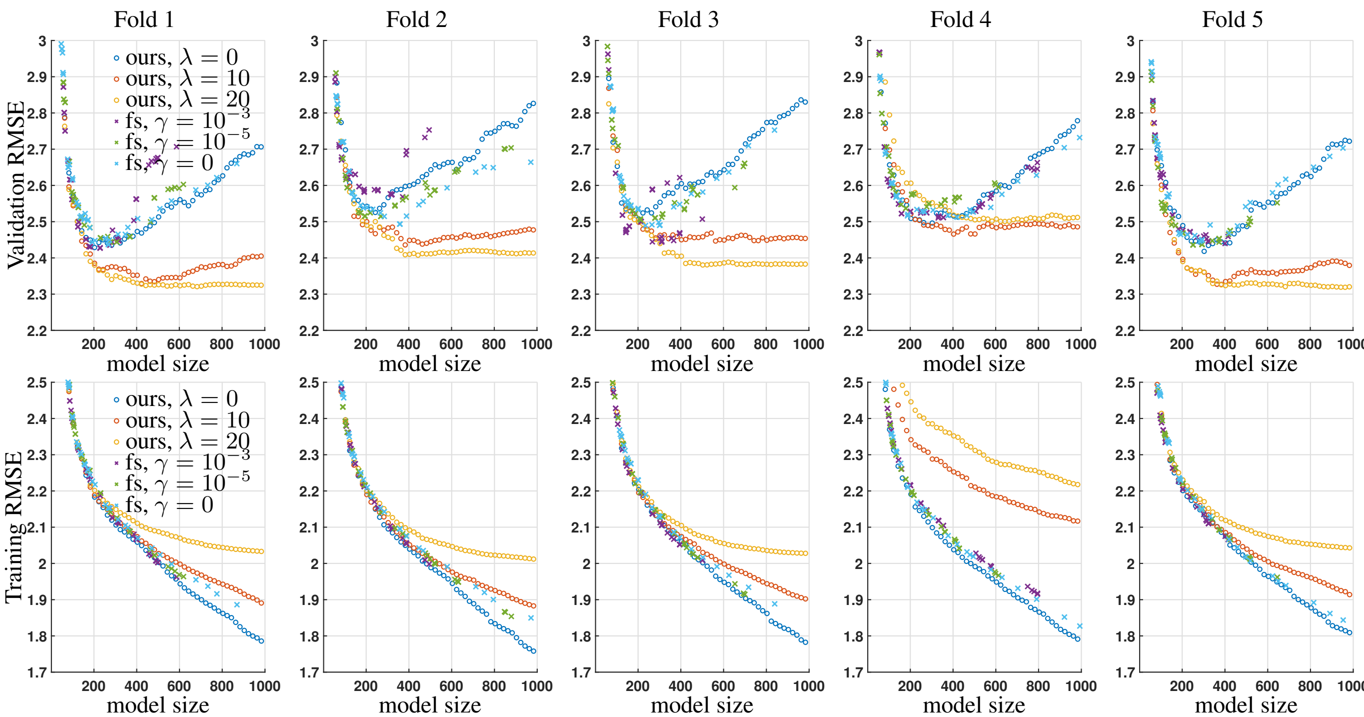

Figure 1. Five-fold cross-validation comparison on the CPU Activity regression dataset of the regularization paths generated by our method versus FastSparse (fs). In our approach, the regularization path is constructed by starting with an empty forest and incrementally adding 5 stumps at a time, re-optimizing the entire forest at each step using our algorithm. We present three curves corresponding to different values of the roughness penalty parameter $\lambda \in \{0, 10, 20\}$. For FastSparse, we show results for three different values of the parameter $\gamma \in \{0, 10^{-5}, 10^{-3}\}$. We set the `num_lambda` parameter to 100 and `max_support_size` to 1000, which generates a regularization path across 100 different `lambda_0` values. However, the actual number of unique models produced by FastSparse is typically lower, as multiple values of `lambda_0` often lead to the same model. In contrast, our method allows for more direct control over model size via the number of stumps T, effectively imposing an ℓ_0 constraint.Our regularization path consists of 100 distinct models, although only 50 are shown in the figure to avoid visual clutter. The model size here is defined as the number of thresholds times 2 (a constant piece value and a threshold) plus 1 for the bias.

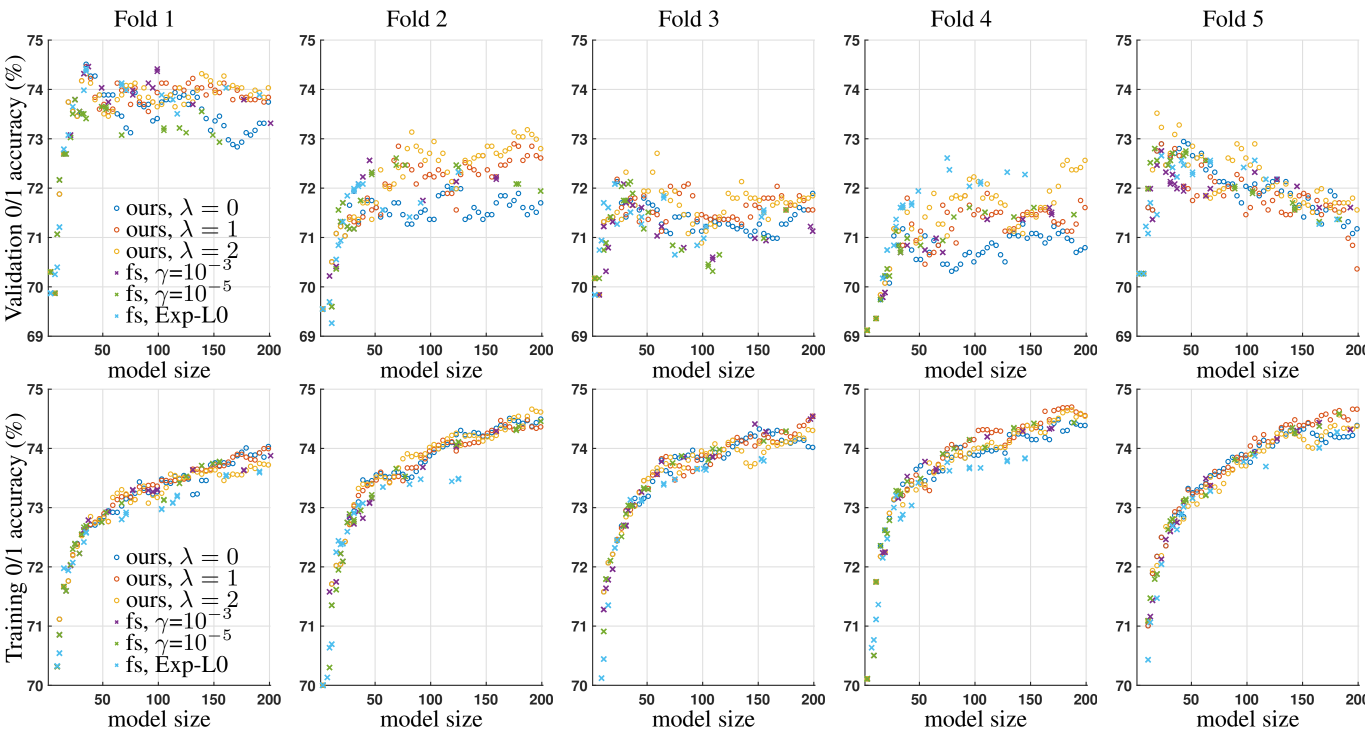

Figure 2. Same setup as Fig. 1, but applied to the FICO classification dataset. For our method, we generate the regularization path by incrementally adding one stump at a time and re-optimizing the current forest using our algorithm, continuing until the forest contains 100 stumps. To reduce visual clutter, we display only 50 points from our regularization path. For FastSparse (fs), following Fig.8 from their paper, we report results for logistic loss with two different values of $\gamma \in \{10^{-5}, 10^{-3}\}$, and for exponential loss with `penalty=L0`. We run the regularization path with `num_lambda=100`, and similarly as with fig. 1, we obtain a fewer number of unique models because multiple values `lambda_0` produce the same result.

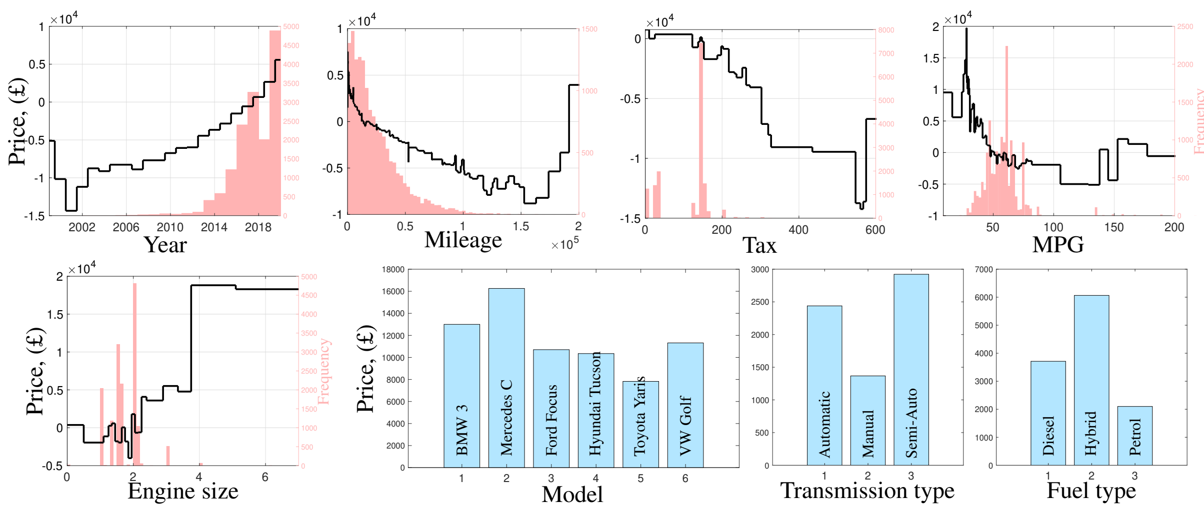

Figure 3. Visualization of the resulting additive model shape functions from our optimized stump forests for the UK used car dataset. For the numerical features, the light red bars show the histogram of the training points with the frequency values given on the right y-axis.

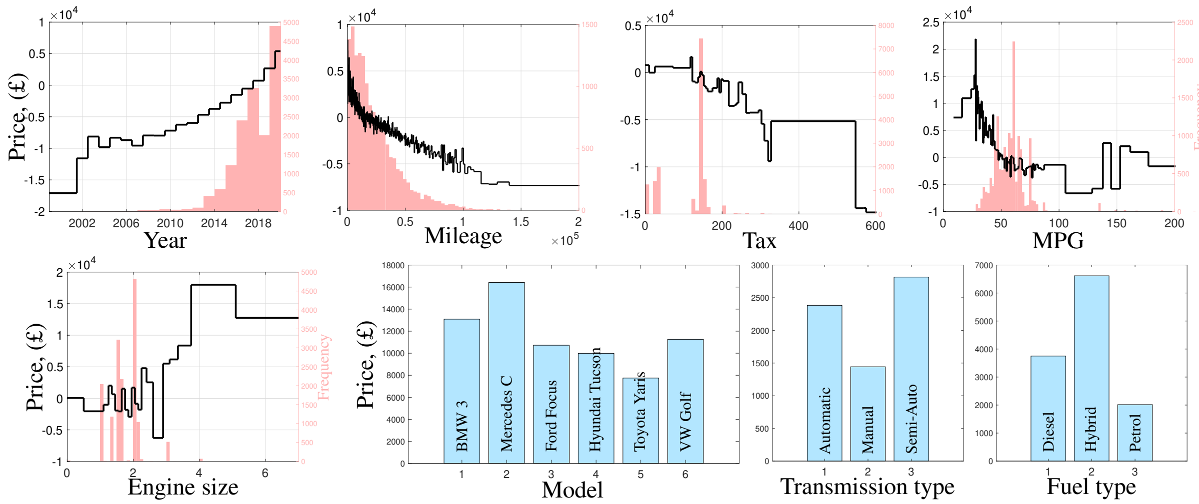

Figure 4. As fig. 3, but for EBM shape functions.

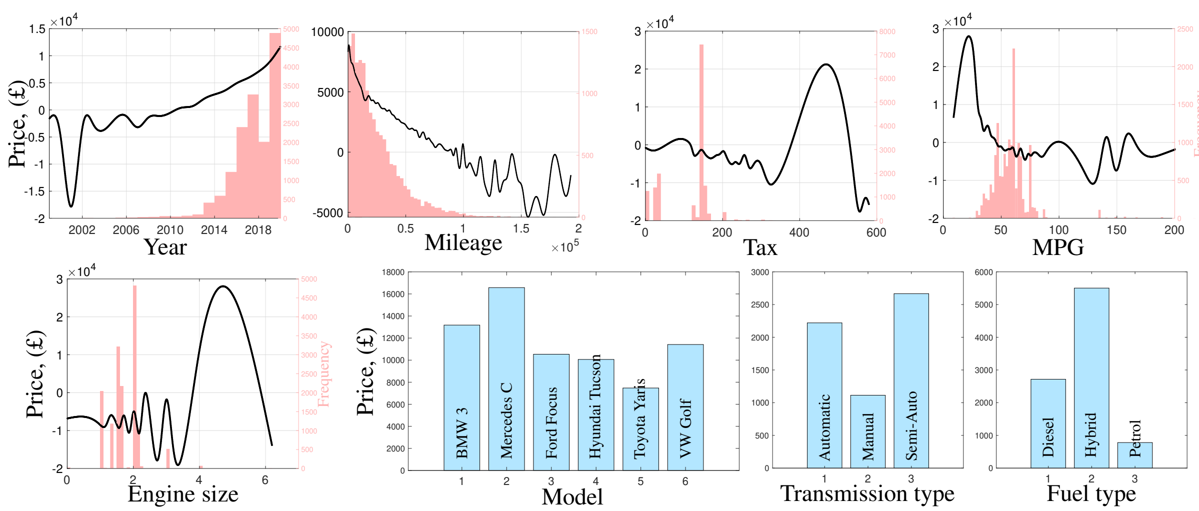

Figure 5. As fig. 3, but PyGAM shape functions.

Table 1. Train and test RMSE, model size (number of parameters) and training time (average $\pm$ standard deviation over 5 runs) for different GAMs. N refers to the dataset size, D is the feature dimension. Green color is the best test error, and blue is the second best. **Rebuttal update 1:** as requested by reviewer r73Z, we performed experiments with FastSparse. We use the Python package `fastsparsegams`. We use the method `convert_continuous_df_to_binary_df` to binarize the features. We perform grid-search on the following values of hyperparameters: `algorithm` = [`CD`, `CDPISI`], `max_support_size` = [10, 100, 1000, 10000], and the 100 λ values in regularization path (we set `num_lambda` = 100, and scan through the `model.lambda_0` values after training). For the `penalty` parameter, we use `L0`. Once we find the best combination of hyperparameters on a holdout set, we train the model 5 times on different training and test splits, consistent with all the performed experiments. **Rebuttal update 2:** as suggested by reviewer owkV, we change the encoding of the leaf fitting problem, and achieve substantial speedups in training time for our method as shown below of our old numbers.

Dataset		ORSF	GB	EBM	Splines	NAM	FLAM	FastSparse
Cpuact	train	2.12±0.01	2.20±0.04	2.19±0.02	2.53±0.02	3.38±0.26	2.88±0.01	2.76±0.03
N=8.2k	test	2.37±0.03	2.43±0.06	2.50±0.05	2.69±0.06	3.41±0.28	2.99±0.05	2.91±0.17
D=21	size	642±0	3.4k±133	16.6k±36	271±3	134k±0	77.9k±123	119±4
	time (s)	~~184±25~~ 9.4±0.3	46±17	39±2	37±0.03	99±1	85±2	3.8±0.5
Wine	train $\times 10^{-2}$	65.70±0.15	68.13±0.27	66.73±0.27	67.99±0.29	74.40±0.16	67.39±0.21	68.01±0.25
N=6.5k	test $\times 10^{-2}$	70.02±0.66	70.92±0.51	70.12±0.39	71.79±1.40	76.07±2.11	70.19±0.84	71.77±0.63
D=11	size	724±12	770±32	3.9k±11	197±7	70.1k±0	5041±11	182±4
	time (s)	~~64±3~~ 6.0±0.3	2.87±0.58	4.44±1.33	56±16	64±0	53±3	0.57±0.07
Housing	train $\times 10^{-2}$	51.84±0.16	54.24±0.27	52.70±0.04	53.37±0.21	71.56±0.30	55.08±0.20	54.62±0.20
N=21k	test $\times 10^{-2}$	54.80±0.65	56.15±0.58	55.23±0.68	55.49±0.61	72.23±0.88	56.24±0.74	56.29±0.65
D=8	size	1.4k±20	2.4k±31	7.2k±8	528±2	51.0k±0	118k±101	579±9
	time (s)	~~600±140~~ 13.6±0.4	42±8	36±2	37±2	175±2	73±2	3.94±0.73
Diamond	train $\times 10^{2}$	9.95±0.02	10.07±0.05	10.11±0.03	10.02±0.02	13.53±0.22	11.75±0.03	10.01±0.02
N=54k	test $\times 10^{2}$	10.15±0.08	10.19±0.08	10.23±0.06	10.96±1.45	13.59±0.25	11.70±0.12	10.17±0.09
D=26	size	934±16	1182±81	3.4k±7	273±24	86k±0	4139±12	516±11
	time (s)	~~648±20~~ 25.1±0.9	140±58	20±2	42±0.4	708±2	805±11	45±10
Year	train	9.12±0.03	9.30±0.03	7.53±0.02	9.14±0.03	10.22±0.05		9.14±0.03
N=423k	test	9.30±0.01	9.35±0.00	9.82±0.02	9.38±0.03	10.22±0.08	out of time	9.29±0.01
D=90	size	1379±0.8	1490±25	368k±0	2158±55	573k±0	> 2 days	2601±63
	time (s)	~~9681±205~~ 1402±43	4368±256	4262±437	3618±45	8858±88		973±65
FPS	train	55.40±0.09	55.48±0.09	55.42±0.09	55.41±0.09	56.23±0.10		55.41±0.09
N=401k	test	55.41±0.34	55.45±0.34	55.42±0.34	55.42±0.34	55.62±0.24	out of time	55.42±0.34
D=100	size	983±37	824±57	2372±12	411±1	288k±0	> 2 days	1250±17
	time (s)	~~6010±314~~ 798±21	1803±466	655±84	2043±2	4397±10		625±10

Table 2. As in Table 1 but for classification datasets. The error is a 0/1 misclassification (%). Similar updates during rebuttal as in Table 1. For FastSparse, we use `Logistic` as the `loss`, and perform hyperparameter search using the same grid search as in Table 1.

Dataset		ORSF	GB	EBM	Splines	NAM	FLAM	FastSparse
Letter	train	15.94±0.14	16.38±0.17	16.12±0.20	15.87±0.14	21.54±1.1	17.94±0.18	15.88±0.14
N=20k	test	16.40±0.52	16.88±0.41	16.63±0.42	16.55±0.70	22.53±1.88	17.95±0.51	16.57±0.67
D=16	size	403±13	420±15	502±2	224±1	68k±0	510±2	399±5
	time (s)	~~150±9~~ 14.9±0.2	32±3	31±1	58±2	153±0	71±1	18±2
Churn	train	18.88±0.19	19.00±0.23	18.84±0.08	18.78±0.15	22.59±2.13	19.85±0.18	18.88±0.11
N=7.0k	test	19.28±0.29	19.32±0.37	19.47±0.51	19.32±0.48	21.69±2.02	20.30±0.88	19.87±0.36
D=45	size	129±5	644±48	7292±11	40±0.04	120k±0	13.7k±15	105±8
	time (s)	~~36±8~~ 6.8±0.4	3±1	15±1	0.5±0.03	120±2	113±2	0.59±0.07
FICO	train	24.86±0.13	26.54±0.15	26.37±0.10	26.79±0.15	28.23±0.41	27.15±0.21	25.87±0.16
N=10k	test	27.33±0.04	27.62±0.30	27.43±0.31	27.35±0.17	28.08±0.61	27.64±0.52	27.80±0.33
D=23	size	550±28	1002±66	3680±9	83±1	130k±0	3791±11	196±10
	time (s)	~~231±18~~ 7.8±0.1	1.6±0.6	7±0.2	1.96±0.10	180±1	61±1	1.74±0.12
IJCNN	train	4.42±0.05	4.56±0.07	4.51±0.03	4.44±0.04	7.51±0.44	6.86±0.08	4.84±0.16
N=50k	test	4.95±0.14	5.10±0.15	5.00±0.14	4.92±0.20	7.48±0.55	7.14±0.15	5.52±0.21
D=22	size	414±23	918±21	12.3k±0	266±0.5	101k±0	828k±242	883±18
	time (s)	~~1090±200~~ 46±1	148±24	19±0	153±40	501±1	249±6	47±1
Covtype	train	22.50±0.03	22.56±0.02	22.46±0.02	22.48±0.02	26.16±0.50		22.49±0.02
N=581k	test	22.71±0.11	22.77±0.10	22.68±0.12	22.72±0.10	26.08±0.54	out of time	22.68±0.10
D=54	size	504±4	1090±32	6402±4	403±1	170k±0	> 2 days	841±15
	time (s)	~~4354±32~~ 1091±16	1202±49	325±5	15624±84	5373±16		2763±177
Bank	train	9.81±0.04	10.00±0.03	9.75±0.05	9.79±0.04	10.09±0.08	11.27±0.04	9.79±0.04
N=41k	test	9.83±0.17	9.99±0.13	9.91±0.17	9.88±0.12	9.87±0.26	11.23±0.15	9.86±0.14
D=62	size	231±4	530±15	1103±7	95±2	174k±0	1182±1	64±4
	time (s)	~~153±11~~ 47±2	34±7	40±3	22±2	662±10	916±3	19.6±3.3

Table 3. The results of Optimal Sparse Regression Trees (OSRT), as requested by reviewer r73Z. The algorithm did not finish within 2 hours for our smallest regression dataset, Wine. Therefore, we run experiments on 1000 sub-sampled training points for all datasets. Most features are continuous, and the algorithm by default seems to binarize using all possible thresholds, resulting in a very large number of features, and thus even slower training. Because of the short time period during this rebuttal, we naively binarize each continuous feature using median threshold. λ is the regularization hyperparameter. The time limit is set for 1 hour.

Dataset	λ	Train RMSE	Test RMSE	Leaves	Depth	Time (s)
diamond*	0.001	2604.93	2862.93	32	12	3600
diamond*	0.005	2589.88	2780.15	8	7	3600
diamond	0.01	2615.15	2741.46	5	5	273.13
diamond	0.05	2806.17	2857.79	2	2	0.70
housing	0.001	0.83	0.96	42	9	0.62
housing	0.005	0.91	0.93	8	5	0.28
housing	0.01	0.90	0.95	5	4	0.20
housing	0.05	0.96	0.99	2	2	0.12
wine	0.001	0.60	0.89	120	11	29.20
wine	0.005	0.72	0.83	26	10	7.87
wine	0.01	0.80	0.80	5	4	2.68
wine	0.05	0.79	0.82	2	2	0.17
cpuact*	0.001	10.74	19.17	55	10	3600
cpuact*	0.005	11.13	17.53	25	8	3600
cpuact	0.01	8.58	19.50	24	11	1658.88
cpuact	0.05	12.39	16.19	8	7	11.03

Table 4. The results of Generalized and Scalable Optimal Sparse Decision Trees (GOSDT), as requested by reviewer r73Z. The algorithm did not finish within 2 hours for our smallest classification dataset, Churn. Similar to OSRT above, we run experiments on 1000 subsampled training points and naively binarize each continuous feature using median threshold for all datasets. λ is the regularization hyperparameter. The time limit is set for 1 hour.

Dataset	λ	Train Error	Test Error	Leaves	Depth	Time (s)
bank	0.001	out-of-time				
bank	0.005	out-of-time				
bank	0.01	out-of-time				
bank	0.05	12.50	11.16	1	1	0.07
fico	0.001	out-of-time				
fico	0.005	out-of-time				
fico	0.01	30.40	30.35	2	2	363.54
fico	0.05	30.40	30.74	2	2	0.65
ijcnn	0.001	6.70	10.27	27	9	18.01
ijcnn	0.005	10.90	9.84	1	1	1.61
ijcnn	0.01	9.40	9.84	1	1	0.12
ijcnn	0.05	9.10	9.84	1	1	0.05
letter	0.001	10.30	23.72	88	12	1272.58
letter	0.005	22.40	25.85	10	6	166.80
letter	0.01	29.70	33.85	4	4	41.59
letter	0.05	33.20	34.08	2	2	0.19
telco	0.001	out-of-time				
telco	0.005	out-of-time				
telco	0.01	out-of-time				
telco	0.05	29.40	26.23	1	1	0.53