AutoSense: A Framework for Automated Sensitivity Analysis of Program Data

Bernard Nongpoh[1], Rajarshi Ray[1], Saikat Dutta[2], Ansuman Banerjee[3]

[1] Department of Computer Science & Engineering, National Institute of Technology Meghalaya, Shillong, India.
[2] Department of Computer Science & Engineering, Jadavpur University, Kolkata, India.
[3] Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India.

Sensitivity Analysis: A Motivating Example

```
bool binsearch(int lo, int hi)
{
    unsigned int size = hi-lo + 1;
    unsigned int mid = (lo+hi)/2;
    if(lo>hi) return false;
    if (size >= 1){
        if(a[mid] == key) return true;
        else if(a[mid]>key)
         return binsearch(lo, mid-1);
  else return binsearch(mid+1, hi);
  }
    return false;
}
```

Sensitive Insensitive

Any inexact value that **size** may take other than 0, the binary search procedure will return an acceptable output.

Problem

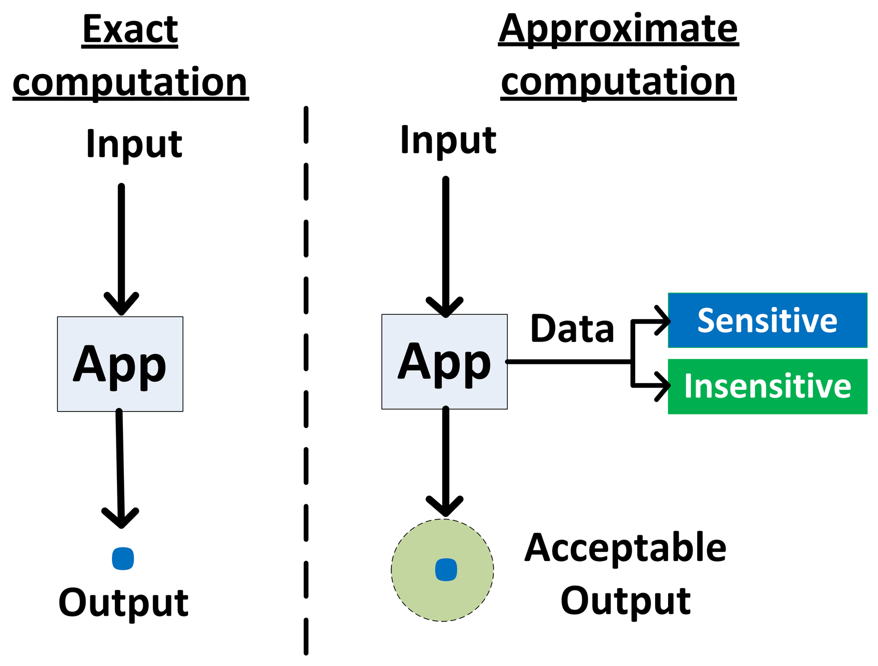


Figure 1: Data classification in approximate computing

Contributions

A collection of systematic methods for program data classification with quantitative confidence guarantee. The contributions are:

- A Dynamic analysis to automatically classify program data as sensitive or insensitive.
- A Static-Dynamic combined analysis for efficiency.

Definition: Sensitive Data

Given an acceptable QoS band for a program $\mathcal{P}$ and a sensitivity threshold probability θ, a program data $v \in \mathcal{D}$ is called sensitive if and only if $\forall e \in E$, the probability that the program output remains in the acceptable QoS band when every instance (v_e, l) in e is replaced with some (v_{approx}, ℓ), is less than θ.

$$\mathcal{SD} = \{v \in \mathcal{D} \mid \forall e \in E, \forall \ell \in \ell_v^e, (v_e, \ell) \rightarrow (v_{approx}, \ell) \implies Pr(\mathcal{R} \in QoS) < \theta\} \quad (1)$$

where $(v_e, \ell) \rightarrow (v_{approx}, \ell)$ means the substitution of (v_{approx}, ℓ) in place of (v_e, ℓ). The set of *insensitive* data is defined as $\overline{\mathcal{SD}} = \mathcal{D} - \mathcal{SD}$.

Sensitivity Analysis Using Hypothesis Testing

For every $v \in \mathcal{D}$, we propose a hypothesis that $\forall e \in E, \forall \ell \in \ell_v^e, (v_e, \ell) \rightarrow (v_{approx}, \ell) \implies \mathcal{R} \in QoS$, where E, ℓ_v^e, (v_e, ℓ) and (v_{approx}, ℓ). Let us denote such an hypothesis by K. Test the following null and contrary hypothesis:

$$H : Pr(K) < \theta \qquad H' : Pr(K) \geq \theta \quad (2)$$

where $Pr(K)$ is the probability that the hypothesis K is true.

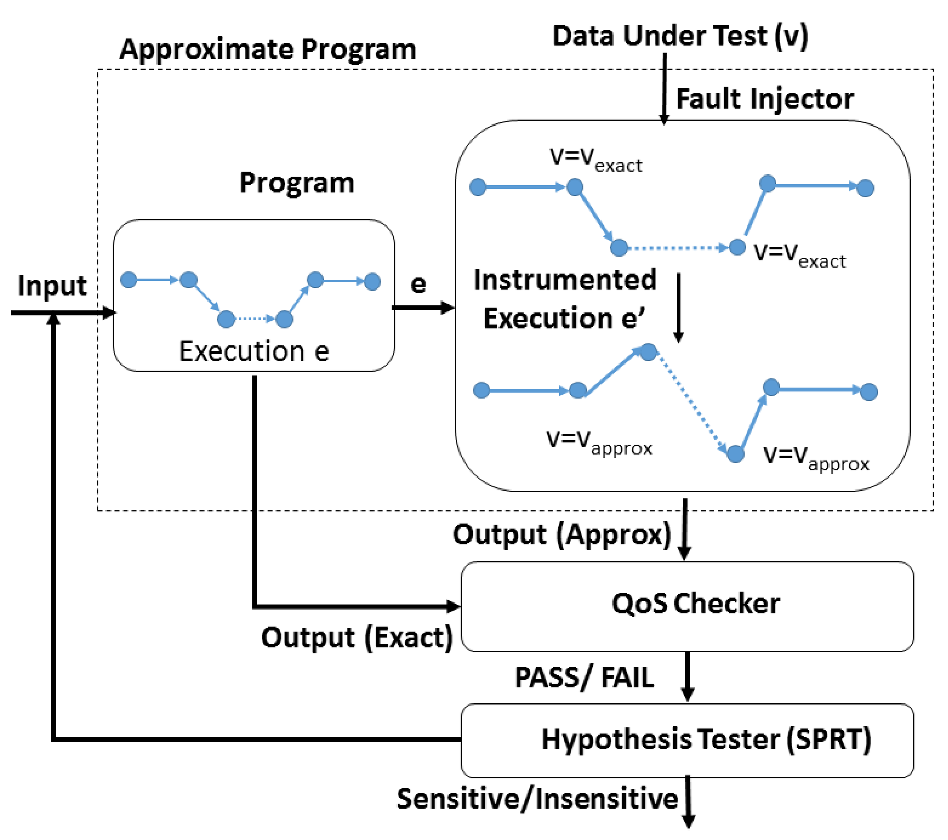


Figure 2: Framework of Dynamic Sensitivity Analysis with Hypothesis Testing

Sequential Probability Ratio Test

- SPRT is to decide whether additional experiments need to be performed to accept or reject a hypothesis on the basis of the previously observed outcomes.

Limitation of Dynamic Analysis

- Compute and data intensive programs may take a long time to terminate, making each trial during the hypothesis testing expensive.
- Generating random inputs for many applications can be challenging.

Static-Dynamic Combined Analysis

The elements of the complete lattice L of our analysis are mappings $\sigma : \mathcal{D} \rightarrow \{\bot, S, I, \top\}$. $\sigma(x) = \bot$ denotes that no information is known about the data x whereas $\sigma(x) = \top$ denotes that x may be *sensitive* or *insensitive*. $\sigma(x) = S$ and $\sigma(x) = I$ denotes x to be *sensitive* and *insensitive* respectively. We define a *data sensitivity lattice* over the range of σ, i.e., $\{\bot, S, I, \top\}$

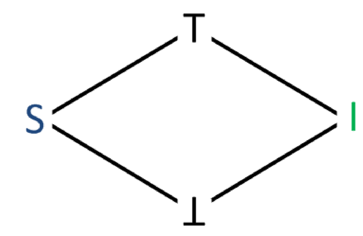


Figure 3: Data Sensitivity Lattice

The partial order on σ is defined :

$$\forall \sigma : \bot \sqsubseteq \sigma \qquad \forall \sigma_1, \sigma_2 : \sigma_1 \sqsubseteq \sigma_2 \text{ iff } \forall x, \sigma_1(x) \sqsubseteq_D \sigma_2(x). \quad (3)$$

where $\bot \in \sigma$ maps every $x \in \mathcal{D}$ to $\bot$, $\sqsubseteq$ denotes the partial order relation on σ and $\sqsubseteq_D$ denotes the partial order relation of the *data sensitivity lattice*. The *join* operation over σ is defined in Eq. 4.

$$(\sigma_1 \sqcup \sigma_2)(x) = \sigma_1(x) \sqcup \sigma_2(x) \quad (4)$$

Considering a general assignment statement block $[x := a]$, a being any expression, we define the transfer functions of our analysis as:

$$[x = a] : f(\sigma) = \begin{cases} \sigma(x \rightarrow I) & \text{if } \forall v \in FV(a), \sigma(v) = I \\ \sigma(x \rightarrow S) & \text{if } \forall v \in FV(a), \sigma(v) = S \\ \sigma(x \rightarrow \top) & \text{if } \exists u, v \in FV(a) \\ & \quad s.t.\ \sigma(u) = S, \sigma(v) = I \\ \sigma & \text{if } FV(a) = \emptyset \end{cases} \quad (5)$$
$$[\cdots] : f(\sigma) = \sigma$$

where $[\cdots]$ is to denote any program statement which is not an assignment statement and $FV(a)$ is the set of all free variables of the expression a.

Example

```
double average(int N,int a[])
{
 double sum=0;
 for(int i=0;i<N;i++)
    sum=sum+a[i];
 avg=sum/N; //avg is I as both sum,N are I
 return avg;
}
```

Reliability of Sensitivity Analysis

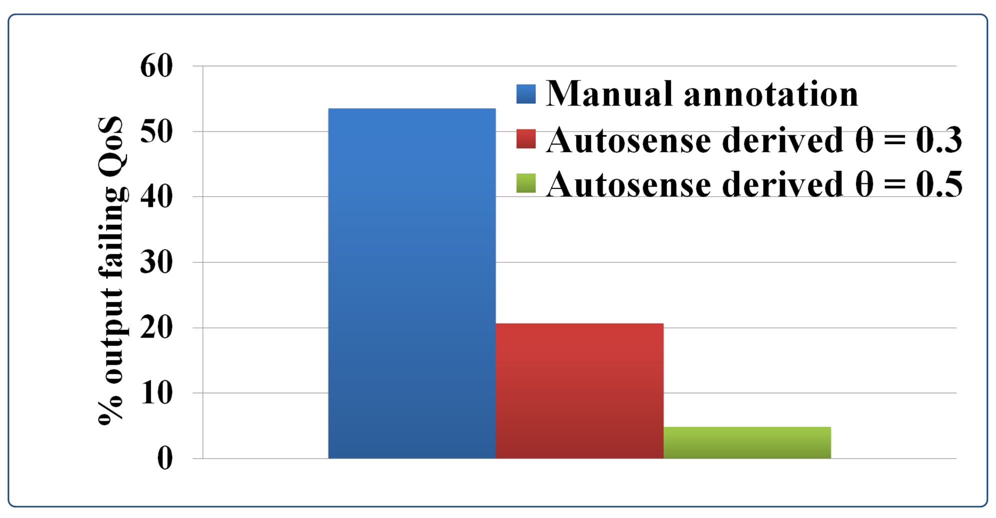
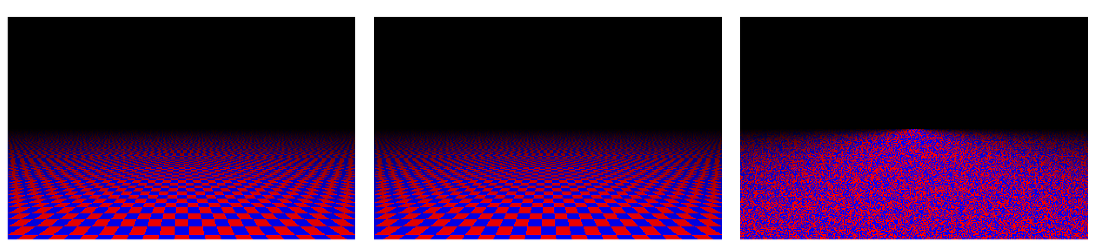


Figure 4: Percent output failing QoS with confidence $\theta = 0.3$ and $\theta = 0.5$



(a) Original image | (b) AutoSense guided approximation | (c) Manual guided approximation

Figure 5: Raytracer rendered image with AutoSense guided approximation

Evaluation of Dynamic Sensitivity Analysis

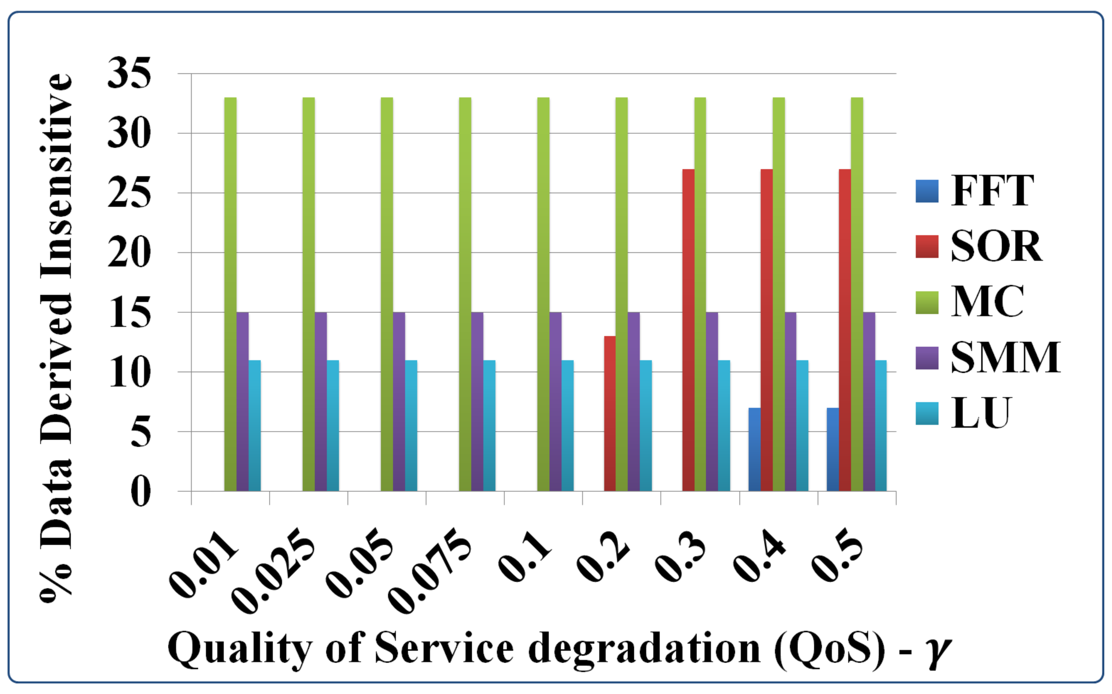


Figure 6: The percent insensitive data reported by a AutoSense on varying QoS γ and fixed probability factor $\theta = 0.5$

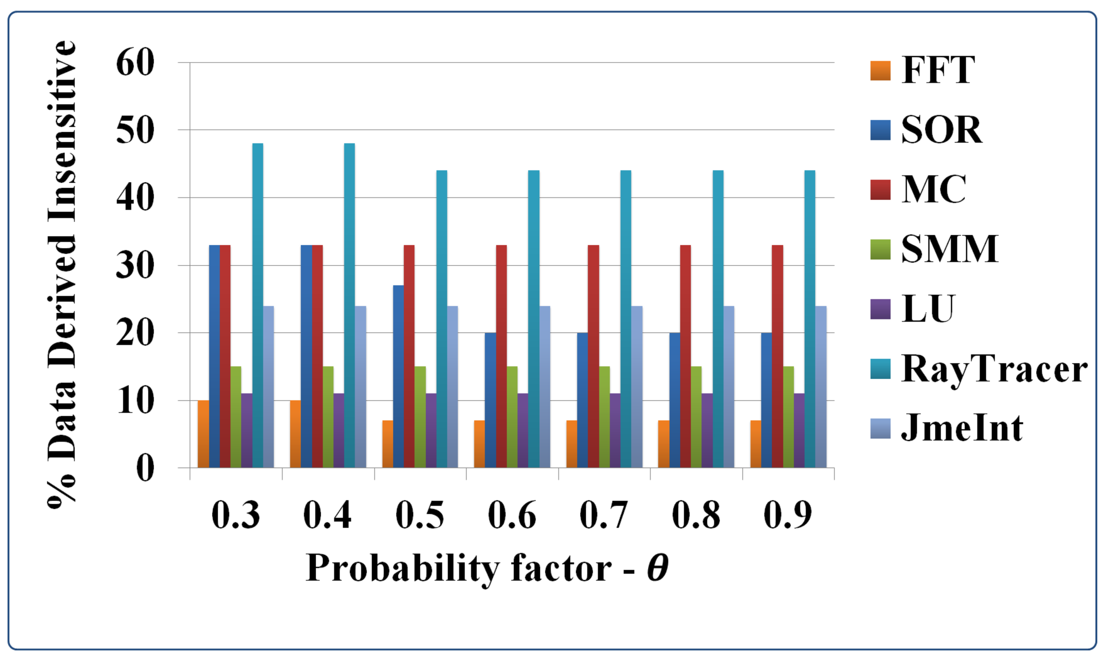


Figure 7: The percent insensitive data reported by a AutoSense on varying θ and fixed QoS $\gamma = 0.5$ (scimark2), PSNR=10.5 (raytracer) and exact (jmeint)

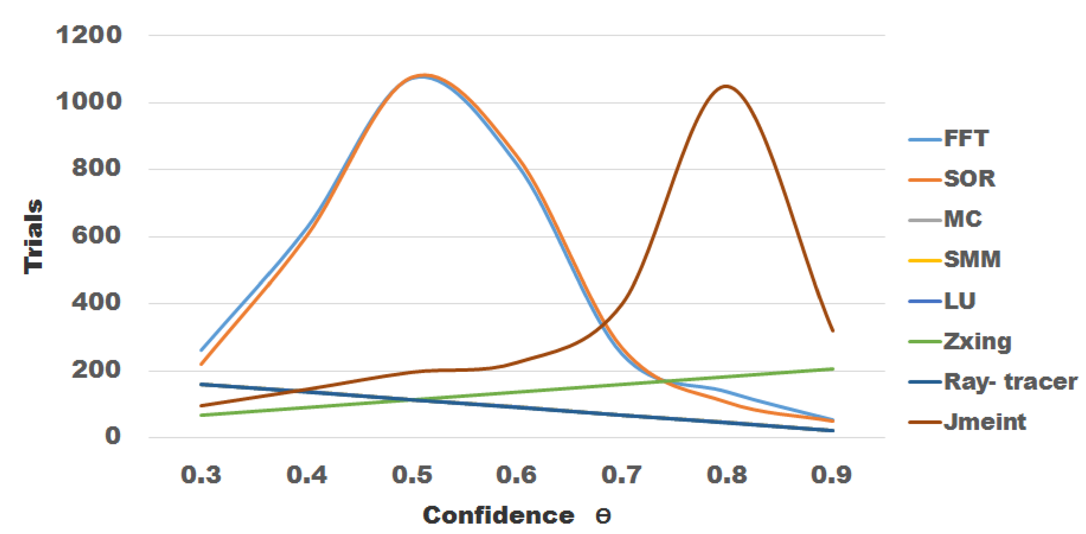


Figure 8: Number of Trials vs. Confidence θ

Evaluation of Combined Analysis

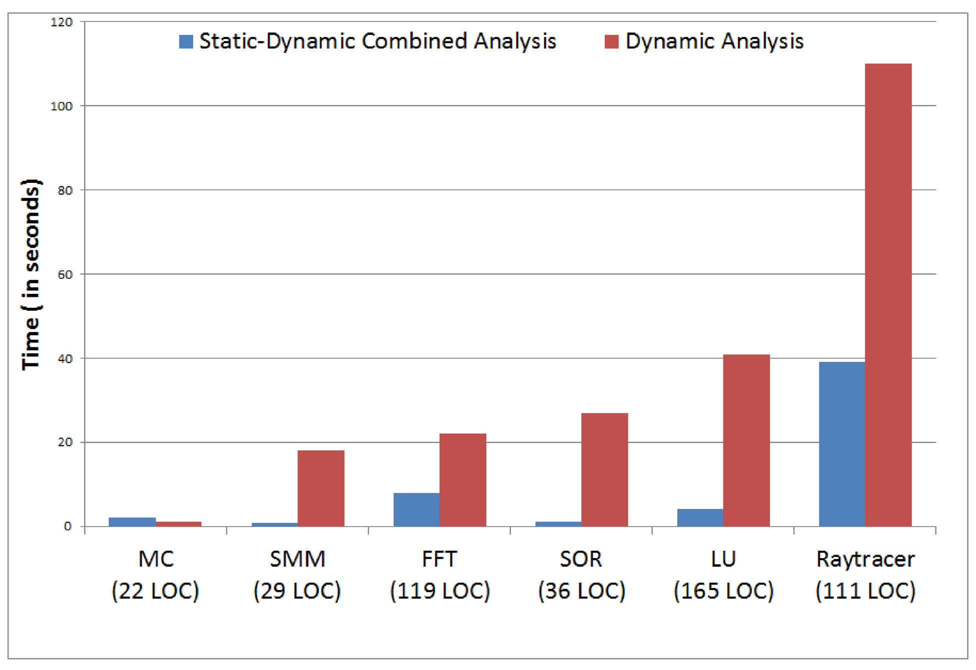


Figure 9: Performance of Static-dynamic combined vs. Dynamic analysis

Application	TP	FP	FN	Precision (%)	Recall (%)
FFT	0	0	3	0	0
SOR	3	0	0	100	100
MC	1	0	1	100	50
SMM	2	0	0	100	100
LU	0	0	9	0	0
Raytracer	0	1	2	0	0

Table 1: Precision, Recall of the Combined Analysis w.r.t. Dynamic Analysis

Conclusions

- Identifying insensitive data of an application is non-trivial, especially when the application is large and has substantial data and control dependencies.
- Illustrated that a systematic study of the effect of inaccuracy in program data with statistical methods like hypothesis testing can lead to automatic classification of insensitive and sensitive data.