Expectation-Oriented Framework for Automating Approximate Programming

Jongse Park, Kangqi Ni, Xin Zhang,
Hadi Esmaeilzadeh, Mayur Naik

Alternative Computing Technologies (ACT) Lab

Georgia Institute of Technology

Approximate Programming

Programmer's

manual/explicit specification

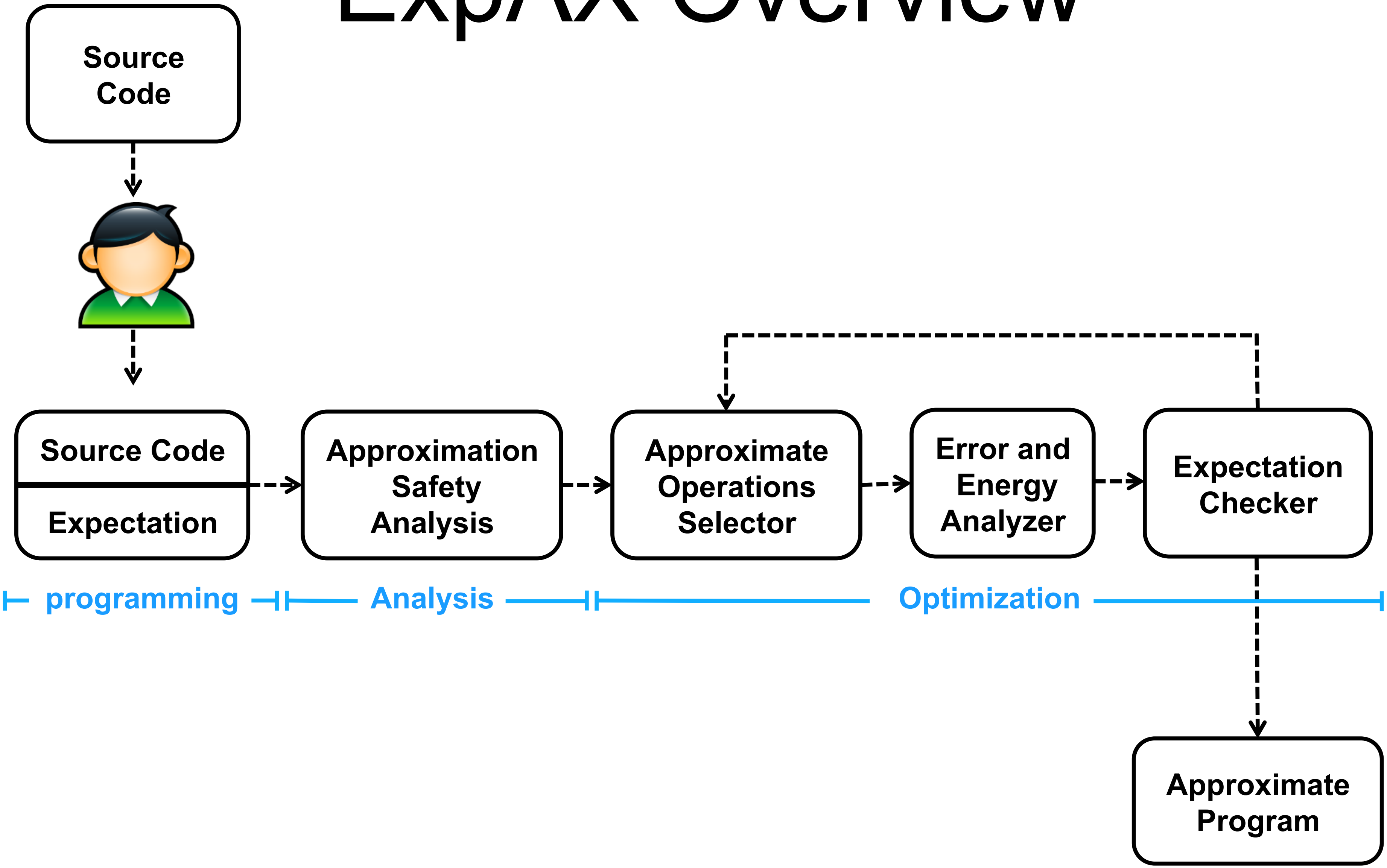
[EnerJ PLDI'11, Rely OOPSLA'13]

AUTOMATE

approximate programming

Where? How much?

ExpAX Overview



Programming Model

Programmer's Annotations with **Expectation**

1. accept rate(v) < c

   e.g. rate(v) < 0.2

2. accept magnitude(v) < c using f

   e.g. magnitude(v) < 0.1

3. accept magnitude(v) > c using f with rate < c'

   e.g. magnitude(v) > 0.9 with rate < 0.3

Approximation Safety Analysis

Find possible safe-to-approximate variables

Unsafe-to-approximate variables

1. Variables violating memory safety
2. Variables violating functional correctness

Approximation Safety Analysis

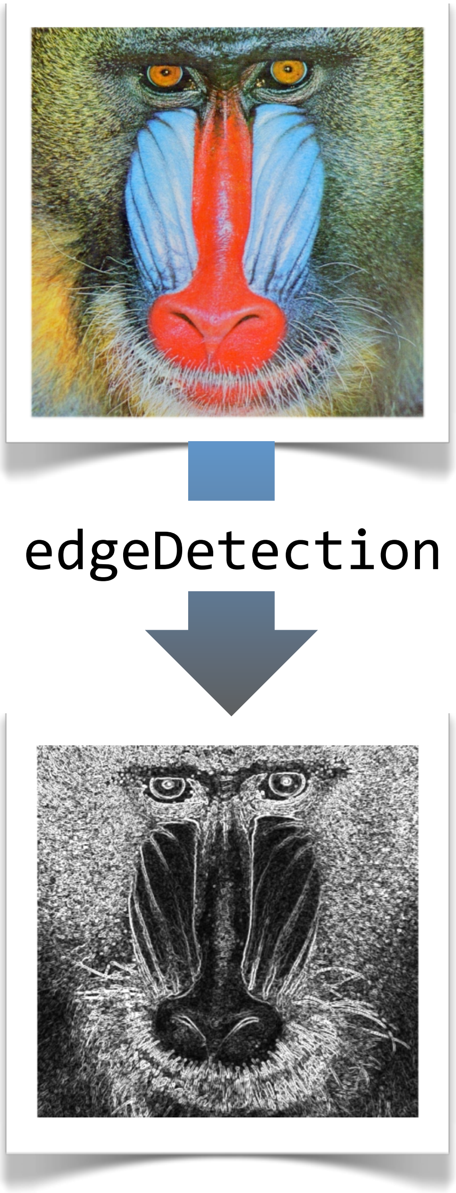
Backslicing Analysis

For each variable **v** in program,
find all variables **contributing** to the variable **v**

unsafe-to-approximate variables
should be **precise**

Everything else should be **precise** variables

Example



edgeDetection

```
Float sobel (float[3][3] p) {
    float x, y, gradient;
    x = (p[0][0] + 2 * p[0][1] + p[0][2]);
    x += (p[2][0] + 2 * p[0][1] + p[2][2]);
    y = (p[0][2] + 2 * p[1][2] + p[2][2]);
    y += (p[0][0] + 2 * p[1][1] + p[2][0]);
    gradient = sqrt(x * x + y * y);
    ...
    return gradient;
}
```

```
void edgeDetection(Image &src, Image &dst) {
    grayscale(src);

    for (int y = …)
        for (int x = …)
            dst[x][y] = sobel(window(src, x, y));

    accept rate(dst) < 0.1;
}
```

Optimization

Find a subset of safe-to-approximate operations

- Minimize error
- Maximize energy saving

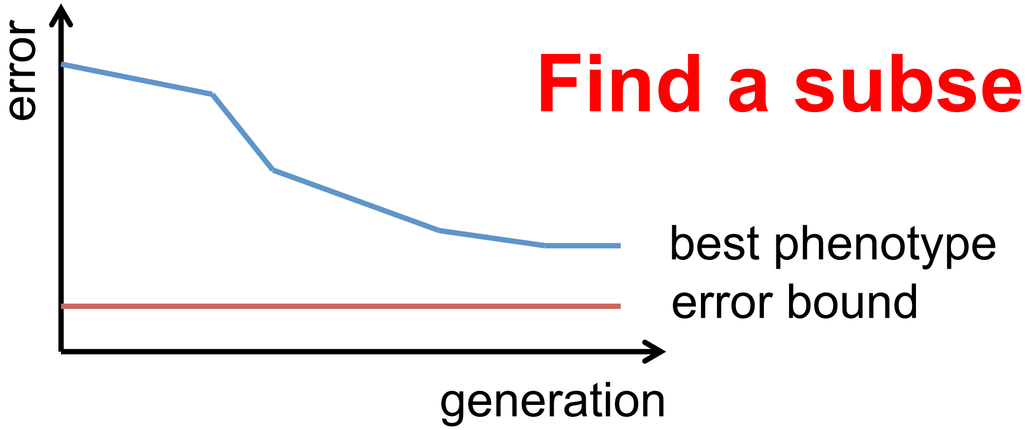
Objective function

$$f(subset) = \left(\alpha \times error + \beta \times energy\right)^{-1}$$

Genetic algorithm

phenotype: a bitvector representing a subset (approximate('0') or precise('1'))

Statistical Guarantee



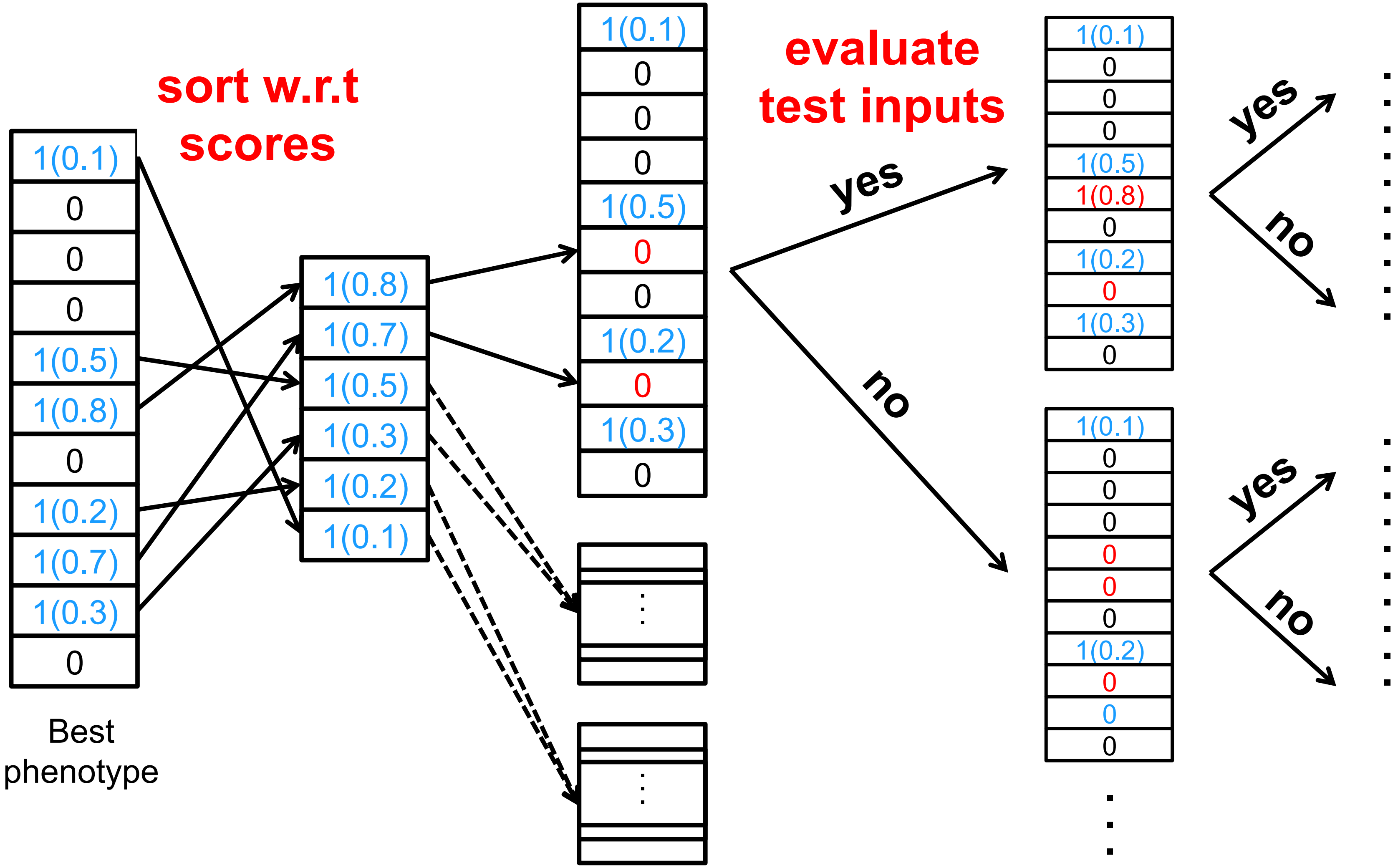
Find a subset of best phenotype!

For each eval in genetic algorithm:

calculate a score for each operation

$$f(operation) = \sum_{eval \in Eval} \left(\frac{\alpha \times error + \beta \times energy}{n(approx)} \right) / \; n(Eval)$$

Space Exploration with Transformed Best Phenotype



Evaluation

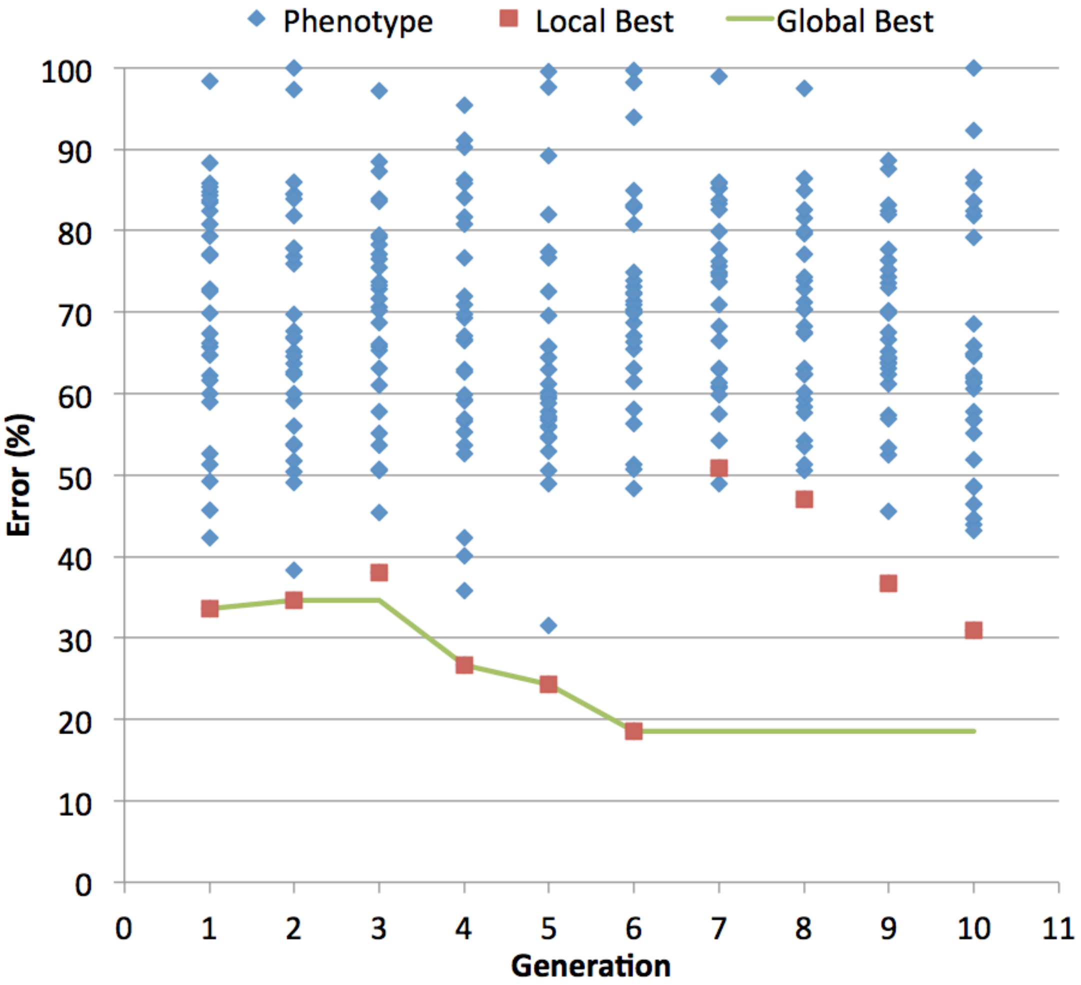
Benchmarks:
scimark2 – FFT, LU, SOR, MonteCarlo, SMM
Imagefill, raytracer, jmeint, zxing

Simulator:
Open-source simulator provided by EnerJ

Analysis Result

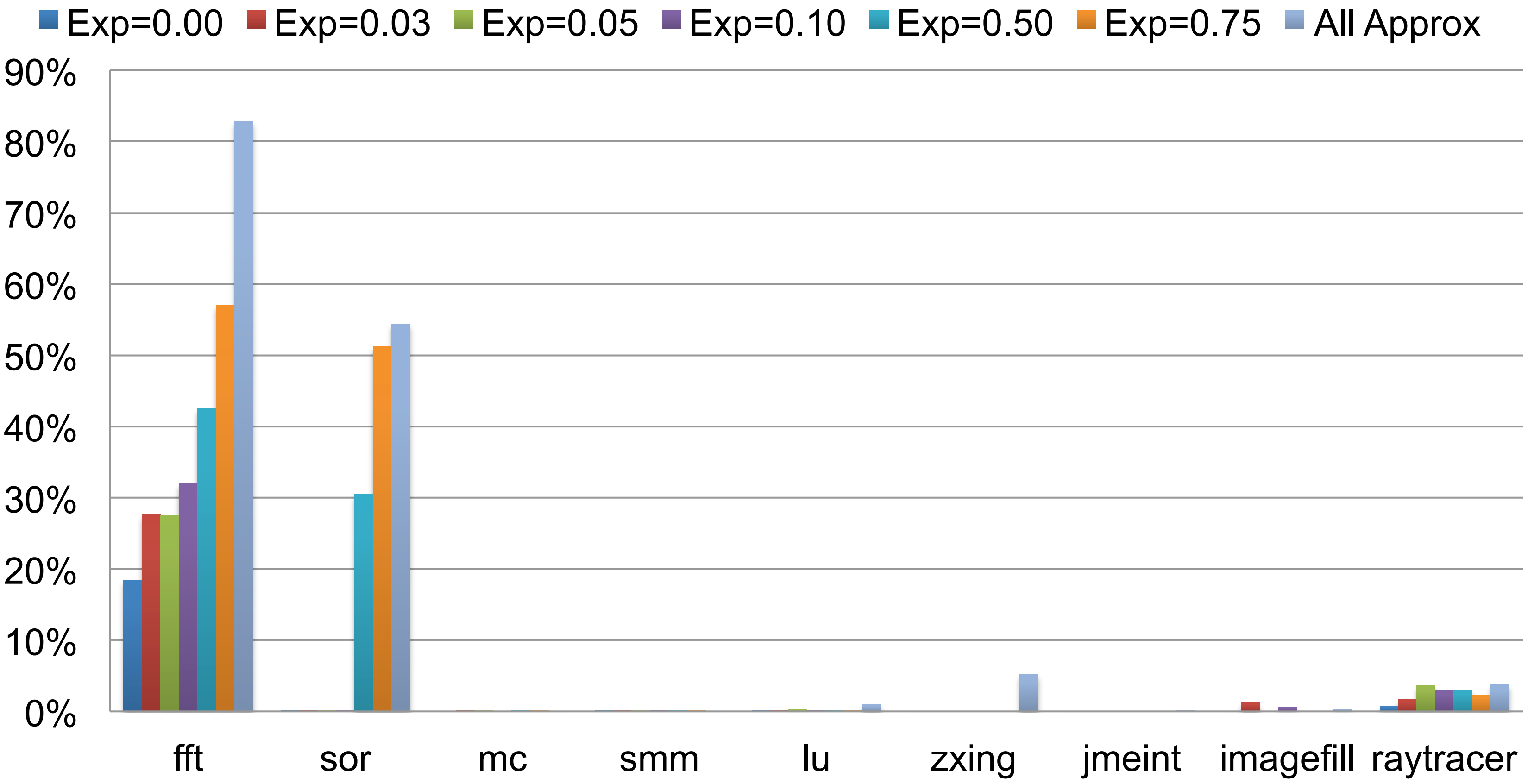
BenchName	Enerj: # of Annotations	ExpAX: # of Expectations
FFT	**27**	**1**
LU	**20**	**1**
SOR	**9**	**1**
MonteCarlo	**3**	**1**
SMM	**8**	**1**
imagefill	**28**	**7**
RayTracer	**27**	**2**
jmeint	**113**	**1**
zxing	**172**	**15**

Genetic Algorithm Results



LU on aggressive system specification

Error


Exp=0.00
Exp=0.03
Exp=0.05
Exp=0.10
Exp=0.50
Exp=0.75
All Approx
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
fft
sor
mc
smm
lu
zxing
jmeint
imagefill
raytracer

Conclusion

Expax:
an expectation-oriented framework for
automating approximate programming

1. Programming model with a new program specification
2. Approximation safety analysis
3. Optimization framework
   with heuristics for statistical guarantee