Efficient Search for Inputs Causing High Floating-point Errors

Wei-Fan Chiang, Ganesh Gopalakrishnan, Zvonimir Rakamarić, and Alexey Solovyev
School of Computing, University of Utah,
Salt Lake City, UT

Supported in part by NSF grants ACI 1148127, CCF 1255776, CCF 1302449 and CCF 1346756.
Floating-point Computations in Sequential and Parallel Software

• Important applications such as weather prediction are accuracy-critical
• Everyday applications (e.g. cell-phone apps) run at lower FP precision
• Challenge: Knowing whether they give imprecise results for any input

Photo courtesy to droyospencer.com, aptito.com/blog, and itunes.apple.com.
Dangers of Inadequate or Inconsistent Precision

- **Patriot Missile Failure in 1991.**
  - Miscalculated distance due to floating-point error.
- **Inconsistent FP Calculations [Meng et al, XSEDE ‘13]**

\[
P = 0.42187499999994488848768742172978818416595458984375 \\
C = 0.0026041666666666665221063770019327421323396265506744384765625 \\
\text{Compute: } \text{floor}( P / C )
\]

- **Xeon**
  \[
P / C = 161.9999... \\
\text{floor}( P / C ) = 161
\]
  - Expecting 161 msgs

- **Xeon Phi**
  \[
P / C = 162 \\
\text{floor}( P / C ) = 162
\]
  - Sent 162 msgs
Problem Addressed

• How to tell which inputs maximize error?
• This is important for many reasons:
  – Characterize libraries precisely
  – Support tuning precision
  – Help decide where error-compensation is productive
Difficulties

• Large code-sizes
• Presence of non-linear operators
• Presence of data-dependent conditionals
• Concurrency (schedules may affect results)
Main Contribution

- A practical technique for reliable precision estimation for sequential and parallel programs.
  - Search based input generation.
  - Handles diverse operations.
  - Improves scalability.

- Usage scenarios:
  - Precision bottleneck detection.
  - Auto-tuning.
Previous Work

• Over-approximation based (false alarms likely):
  – Interval arithmetic: Examples
    • x in [-1, 2] and y in [2, 5]. Then (x * y) returned as [-5, 10].
    • x in [-1, 1]. Then (x – x) returned as [-2, 2] (must be 0)
  – Affine arithmetic: Basic idea
    • Each number is represented by a polynomial.
    • Linear approximation of non-linear operation.
  – SMT
    • Encodes error bound described in IEEE-754 standard.

• Under-approximation based (no false alarms):
  – Random testing.
Illustration of Interval Arithmetic

1. float $x_0, x_1, x_2$ in $[1.0, 2.0]$
2. float $p_0 = (x_0 + x_1) - x_2$
3. float $p_1 = (x_1 + x_2) - x_0$
4. float $p_2 = (x_2 + x_0) - x_1$
5. float $sum = (p_0 + p_1) + p_2$
6. Error? $sum \n/\!\!/ (x_0 + x_1) + x_2$

<table>
<thead>
<tr>
<th>Value of $sum$</th>
<th>Exact</th>
<th>Interval Arithmetic (Gappa)</th>
<th>Affine Arithmetic (SmartFloat)</th>
<th>SMT based</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[3.0, 6.0]$</td>
<td>$[0.0, 9.0]$</td>
<td>$[3.0, 6.0]$</td>
<td>$[3.0, 6.0]$</td>
<td>$[3.0, 6.0]$</td>
</tr>
<tr>
<td>Error on $sum$</td>
<td>?</td>
<td>Infinite</td>
<td>1.0362e-15</td>
<td>4.9960e-15</td>
</tr>
</tbody>
</table>
1. float $x_i$ in $[1.0, 3.0]$  // $0 \leq i \leq 7$
2. float sum = summation of $x_i$
3. Consider $x_i$ in $[1.0, 2.0]$
4. Error? sum

<table>
<thead>
<tr>
<th></th>
<th>Exact</th>
<th>Interval Arithmetic (Gappa)</th>
<th>Affine Arithmetic (SmartFloat)</th>
<th>SMT based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value of sum</strong></td>
<td>[8.0, 16.0]</td>
<td>[8.0, 16.0]</td>
<td>N/A</td>
<td>[8.0, 16.0]</td>
</tr>
<tr>
<td><strong>Error on sum</strong></td>
<td>?</td>
<td>7.7548e-16</td>
<td>N/A</td>
<td>Timeout</td>
</tr>
</tbody>
</table>
Previous Work

• Over-approximation:

<table>
<thead>
<tr>
<th></th>
<th>Interval Arithmetic</th>
<th>Affine Arithmetic</th>
<th>SMT based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor scalability</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Overly pessimistic results</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limited support for non-linear operation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Limited support for conditionals</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

• Our overall approach: Under-approximation based
  – Naïve Random Testing produces VERY LOOSE lower bounds
  – Our focus: How to produce tight lower-bounds?
Why do we base our approach on Guided Random Testing?

• Seems to be the only approach that can handle
  – Large Programs
  – Non-linear operators
  – Data dependent conditionals

No “closed form” solutions are possible

• At present, designers have no tools that can analyze programs with these features
  – Ours is the first practical tool in this area
Precision Measurement by Random Testing

Low Precision Program \rightarrow \text{Low Precision Result}

\{ \text{configuration} \}
\begin{align*}
X_0 &\leftarrow (\ldots) \\
X_1 &\leftarrow (\ldots) \\
X_2 &\leftarrow (\ldots)
\end{align*}

High Precision Program \rightarrow \text{High Precision Result}

Error Calculation* \rightarrow \text{Low Precision Result}

* “Error” = Relative Error (See paper for details)
Search Based Random Testing

• Our Contribution: Random Testing with Good Guidance Heuristics can Outperform Naïve Random

• We propose Binary Guided Random Testing
Search Based Random Testing

• Randomly sample inputs around “sour-spots!”
  – A “sour-spot” causes highly imprecise program output.
  – Definition of “Configuration:”
    An assignment from input variables to their probing intervals.

{ Configuration: 
  X0 ← [0.0 1.0] 
  X1 ← [1.1 2.2] 
  X2 ← [2.3 3.3] 
}

Program → Result
Search Based Random Testing

- Randomly sample inputs around "sour-spots!"
  - A "sour-spot" causes highly imprecise program output.
  - **Definition of “Configuration:”**
    
    An assignment from input variables to their probing intervals.

![Configuration Diagram]

Program

Imprecise Result

```
X0 \leftarrow \begin{bmatrix}
0.0 & 0.5 & 1.0 \\
1.1 & 1.5 & 2.2 \\
2.3 & 3.0 & 3.3 \\
\end{bmatrix}
```

```
X0 = 0.5 
X1 = 1.5 
X2 = 3.0 
```
Search Based Random Testing

- Randomly sample inputs around “sour-spots!”
  - A “sour-spot” causes highly imprecise program output.
  - **Definition of “Configuration:”**
    
    An assignment from input variables to their probing intervals.
Importance of Selecting Good Configurations

![Graph showing relative error vs number of samples for different configurations.]

- **Number of Samples**
- **Importance of Selecting Good Configurations**

Graph:
- **Good Conf.**
  - $x_0$ <-> 
  - $x_1$ <-> 

- **Original Conf.**
  - $x_0$ <-> 
  - $x_1$ <-> 

- **Bad Conf.**
  - $x_0$ <-> 
  - $x_1$ <-> 

Legend:
- **Orig.**
- **Bad**
- **Good**
Binary Guided Random Testing: Search and Test Around Sour-spots

• **Key Observations:**
  – “Sour spots” can be improved with more probing
  – Configurations can be ranked without too much probing

• **The optimization problem:**
  – Find a configuration that contains inputs causing high floating-point errors.
  – We propose Binary Guided Random Testing (BGRT).
  – We compared BGRT against other search methods, obtaining encouraging results
High-level View of BGRT

Original Conf.

Init

Derive Configuration to Generate Candidates

\{
\text{sub-conf. 1}
\}\quad \cdots \quad \{
\text{sub-conf. } \ n
\}\ni\text{Candidates}

Program
High-level View of BGRT

Original Conf. \{\}

Init

Derive Configuration to Generate Candidates

\{ sub-conf. 1 \} \ldots \{ sub-conf. n \}
Candidates

Choose the BEST Sub-conf.

Evaluate

Program
High-level View of BGRT

Original Conf.

Init

Derive Configuration to Generate Candidates

 Candidates

For each sub-conf., sample few inputs. Also Record the detected highest error.

Choose the BEST Sub-conf.

Evaluate

Program
High-level View of BGRT

Original Conf.\)
Init

Derive Configuration to Generate Candidates

\{ sub-conf. 1 \} \ldots \{ sub-conf. n \} Candidates

Evaluate

Choose the BEST Sub-conf.

sub-conf. k
The BEST among candidates

For each sub-conf., sample few inputs. Also Record the detected highest error.
High-level View of BGRT

Original Conf. → Restart?

Init

Derive Configuration to Generate Candidates

\{ \text{sub-conf. } 1 \} \ldots \{ \text{sub-conf. } n \} \quad \text{Candidates}

For each sub-conf., sample few inputs. Also Record the detected highest error.

Choose the BEST Sub-conf.

Evaluate

Program

Choose the BEST Sub-conf.

The BEST among candidates
High-level View of BGRT

Original Conf.

Init

Derive Configuration to Generate Candidates

Candidates

{sub-conf. 1} ...... {sub-conf. n}

Evaluate

Program

Choose the BEST Sub-conf.

The BEST among candidates

Restart?

OR

sub-conf. k

For each sub-conf., sample few inputs. Also Record the detected highest error.
A Closer View of BGRT

- Partition the variables (with their ranges).
A Closer View of BGRT

- Shrink variables’ ranges.
  - Each partition generates its “upper” and “lower” sub-partitions.
A Closer View of BGRT
A Closer View of BGRT

Candidates
These candidates are evaluated using random sampled inputs.

\[
\begin{align*}
&\{X_0 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_1 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_2 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_0 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_1 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_2 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_0 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_1 \leftarrow \bullet\bullet\bullet\bullet\bullet\} \\
&\{X_2 \leftarrow \bullet\bullet\bullet\bullet\bullet\}
\end{align*}
\]
Other Search Strategies We Investigated

• Iterated Local Search (ILS)
• Particle Swarm Optimization (PSO)
• Our results suggest BGRT as the better search strategy for precision measurement.
  – Focuses the search near sour-spots.
• Website for additional documents:
  – www.cs.utah.edu/fv/Gauss/Pages/grt
Experimental Results

• **Comparison among search strategies**
  – Unguided Random Testing (URT), BGRT, ILS, and PSO

• **Benchmarks**
  – Various reduction-tree shapes
  – Direct Quadrature Method of Moments (DQMOM)
  – GPU primitives
Evaluation of BGRT (Reductions)

• Imbalanced reduction (IBR)
• Balanced reduction (BR)
• Compensated imbalanced reduction (IBRK)
• Over-approximation techniques cannot report that the compensated reduction is the most precise.

Balanced Reduction

$$(((v_0 + v_1) + (v_2 + v_3))$$

$$= ((v_0 + v_1) + (v_2 + v_3))$$

$$= (v_0 + v_1) + (v_2 + v_3)$$

$$= v_0 + v_1 + v_2 + v_3$$

Imbalanced Reduction

$$(((v_0 + v_1) + v_2) + v_3)$$

$$= ((v_0 + v_1) + v_2) + v_3$$

$$= (v_0 + v_1) + v_2 + v_3$$

$$= v_0 + v_1 + v_2 + v_3$$

$$= v_0 + v_1$$
Evaluation of BGRT (Reductions)

- 2048 input variables
- *Exp1* and *Exp2* share all the same experiment settings except the seed for random number generation.
A Real-world Sequential Benchmark

- **Direct Quadrature Method of Moments (DQMOM):**
  - A sequential core function of a combustion simulation component of Uintah parallel computational framework.
Evaluation of BGRT (GPU Primitives)

- Fast Fourier Transform (FFT) from Parboil
- LU decomposition from MAGMA library
- QR decomposition from MAGMA library
- Matrix multiplication (MM) from MAGMA library
Evaluation of BGRT (GPU Primitives)

- Input size:
  - FFT: 2048. LU, QR: 1024. MM: 3074.
Challenges and Future Work

• **Improvements:**
  – Coverage
  – Scalability
  – Search strategy improvement

• **Applications:**
  – Combine with auto-tuning
  – Combine with precision bottleneck detection
  – Algorithm comparison
Conclusions

• Guided random testing can detect higher errors than pure random testing.
• Guided random testing overcomes some drawbacks of previous approaches:
  – Improves scalability
  – Handles diverse (e.g. non-linear) operations
  – Supports precision bottleneck detection and auto-tuning
• Our project website
  – http://www.cs.utah.edu/fv/Gauss/Pages/grt
A Comparison Among BGRT, Genetic Algorithm, and Delta Debugging

• **BGRT v.s. Genetic algorithm**
  – BGRT doesn’t have mutation.
  – BGRT only selects one of the best among current candidates to generate next candidates.

• **BGRT v.s. Delta debugging**
  – BGRT could restart the search from the initial conf.
  – Each conf. represents a set of inputs instead of a single input.
## Reductions

<table>
<thead>
<tr>
<th>Exp</th>
<th>Algo.</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IBR (2048)</td>
</tr>
<tr>
<td>Exp1</td>
<td>URT</td>
<td>3.6151e-03</td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>2.7132e-01</td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>2.5134e-02</td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>8.6183e-03</td>
</tr>
<tr>
<td>Exp2</td>
<td>URT</td>
<td>3.1396e-02</td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>2.9659e-01</td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>2.1614e-02</td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>3.1449e-02</td>
</tr>
</tbody>
</table>
**Direct Quadrature Method of Moments**

<table>
<thead>
<tr>
<th>Exp1</th>
<th>Algo.</th>
<th>Error of DQMOM (960)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>URT</td>
<td>8.8723e-03</td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>1.0000e+00</td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>2.0105e-02</td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>1.0133e-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp2</td>
<td>URT</td>
<td>2.4357e-03</td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>4.4318e-01</td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>2.7101e-03</td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>5.9729e-03</td>
</tr>
</tbody>
</table>
## GPU Primitives

<table>
<thead>
<tr>
<th></th>
<th>Algo.</th>
<th>Error</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FFT (2048)</td>
<td>LU (1024)</td>
<td>QR (1024)</td>
<td>MM (3074)</td>
<td></td>
</tr>
<tr>
<td><strong>Exp1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>URT</td>
<td>9.9671e-03</td>
<td>1.1942e-03</td>
<td>3.2723e-02</td>
<td>1.0016e-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>3.4312e-02</td>
<td>2.6197e-02</td>
<td>1.9540e-01</td>
<td>3.1161e+00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>6.8418e-02</td>
<td>3.3736e-03</td>
<td>2.1083e-02</td>
<td>1.6710e-01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>3.5419e-03</td>
<td>2.8987e-03</td>
<td>4.3618e-02</td>
<td>8.6908e-04</td>
<td></td>
</tr>
<tr>
<td><strong>Exp2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>URT</td>
<td>1.9560e-03</td>
<td>1.1742e-03</td>
<td><strong>1.6825e-01</strong></td>
<td>1.5422e-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BGRT</td>
<td>1.2580e-02</td>
<td>2.5969e-02</td>
<td>1.0213e-01</td>
<td><strong>1.7881e-01</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ILS</td>
<td>4.4445e-02</td>
<td>7.9298e-03</td>
<td>3.9839e-02</td>
<td>7.6199e-03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PSO</td>
<td>1.4056e-02</td>
<td>9.3751e-03</td>
<td>8.1161e-02</td>
<td>3.2531e-03</td>
<td></td>
</tr>
</tbody>
</table>