Problems of Dependence Analysis

- Traditional analysis techniques oversimplify the dependence problem ignoring:
  - IF-Statements
  - Coupled Subscripts
  - Complex Loop Bounds etc.

- The vast majority of these techniques apply only to linear expressions
- Non-Linear expressions are found very frequently in actual source code
- Large amount of potential parallelism remain unexploited
Improvements in Data Dependence Analysis

- The I-Test
  - Conclusively proves integer solutions
  - Has near linear time complexity

- The Omega Test
  - Considers if-statement constraints
  - Can provide simultaneous solution across coupled subscripts
  - Can handle complex loop regions (triangular/trapezoidal/symbolic)
  - Can prove dependences

- The Range Test
  - Can handle non-linear expressions
  - Can handle complex loop regions
Limitations of Data Dependence Analysis Techniques

- **The I-Test**
  - Cannot handle non-linear expressions
  - Does not consider if-statements
  - Cannot handle coupled subscripts
  - Cannot handle complex loops regions

- **The Omega Test**
  - Cannot handle non-linear expressions
  - Has exponential time complexity

- **The Range Test**
  - Does not produce complete direction vector information
  - Cannot handle coupled subscripts
  - Cannot prove dependences
The NLVI-Test

- Can handle non-linear expressions
- Can handle simple if-statement constraints
- Can minimize or eliminate coupling across equations
- Can handle complex loop regions such as trapezoidal/symbolic
- Can produce accurate and complete direction vector information
- Can conclusively prove or disprove dependences
- Has polynomial time complexity
Variable Interval

- **Integer Interval**
  
  \[ [L, U] = \{L, L + 1, \ldots, U\} \]

- **Single Variable Integer Interval**
  
  \[ [L(X), U(X)] \text{ where } P \leq X \leq Q = [L(P), U(P)] \cup [L(P + 1), U(P + 1)] \cup \ldots \cup [L(Q), U(Q)] \]

- **General Variable Integer Interval**
  
  \[ [L(x), U(x)] \text{ where } x \text{ in } \mathbb{R} = \bigcup_{x_i \in \mathbb{R}} [L(x_i), U(x_i)] \]

  - e.g. \([1, 4] = \{1, 2, 3, 4\}\)

  - e.g. \([X, 3X], 1 \leq X \leq 3 = [1, 3] \cup [2, 6] \cup [3, 9] = [1, 9] = \{1, 2, \ldots, 9\}\)

  - e.g. \([2X_1X_2 - 1, 3X_1X_2 + X_1], \text{ where } 1 \leq X_1 \leq 5, 1 \leq X_2 \leq X_1^2\)
Variable Interval Theory

- Applies to linear and non-linear expressions
- It relies on the monotonicity of multivariable functions
- Provides the conditions under which a variable integer interval can be contiguous
Fundamental Theorem

Given a single variable integer interval \([L(X), U(X)]\), where \(P \leq X \leq Q\):

a. If \(L\) is decreasing and \(U\) is increasing, then \([L(X), U(X)]\)
is equal to the integer interval \([L(Q), U(Q)]\).

b. If \(L\) is increasing and \(U\) is decreasing, then \([L(X), U(X)]\)
is equal to the integer interval \([L(P), U(P)]\).

c. If \(L\) and \(U\) are increasing and \(U(X_i) - L(X_i + 1) + 1 \geq 0\)for all \(X_i, P \leq X_i \leq Q - 1\), then \([L(X), U(X)]\) is equal tothe integer interval \([L(P), U(Q)]\).

d. If \(L\) and \(U\) are decreasing and \(U(X_i + 1) - L(X_i) + 1 \geq 0\)for all \(X_i, P \leq X_i \leq Q - 1\), then \([L(X), U(X)]\) is equal tothe integer interval \([L(Q), U(P)]\).
Intuition Behind Theorem

\[ L(Q) \quad L(X_i + 1) \quad L(X_i) \quad U(X_i) \quad U(X_i + 1) \quad U(Q) \]

\[ L(P) \quad L(X_i) \quad L(X_i + 1) \quad U(X_i + 1) \quad U(X_i + 1) \quad U(P) \]

\[ a. \ L \downarrow U \uparrow \]

\[ L(P) \quad L(X_i) \quad L(X_i + 1) \quad U(X_i + 1) \quad U(X_i) \quad U(P) \]

\[ b. \ L \uparrow U \downarrow \]

\[ L(P) \quad L(X_i) \quad L(X_i + 1) \quad U(X_i) \quad U(X_i + 1) \quad U(Q) \]

\[ c. \ L \uparrow U \uparrow \]

\[ L(Q) \quad L(X_i + 1) \quad L(X_i) \quad U(X_i) \quad U(X_i + 1) \quad U(P) \]

\[ d. \ L \downarrow U \downarrow \]
Monotonicity of Multi-variable Functions

- Increasing for variable function $F(x, X)$ for variable $X$
  - $F(x_i, X_j) \leq F(x_i, X_j + 1)$ for any $x_i \in Z^n$ and for any $X_j \in Z$

- Decreasing for variable function $F(x, X)$ for variable $X$
  - $F(x_i, X_j) \geq F(x_i, X_j + 1)$ for any $x_i \in Z^n$ and for any $X_j \in Z$

- If $F$ is defined in a region subset of $Z^{n+1}$ then the conditions need only apply for values of $(x, X)$ in that region

- $F$ can be proven increasing if $\min(F(x, X + 1) - F(x, X)) \geq 0$

- $F$ can be proven decreasing if $\max(F(x, X + 1) - F(x, X)) \leq 0$

- For more efficiency we can use the first partial derivative
  - $\min(dF(x, X)/dX) \geq 0$ : Increasing
  - $\max(dF(x, X)/dX) \leq 0$ : Decreasing
Given a variable integer interval \([L(x, X), U(x, X)]\) subject to a set of constraints on \(x\) in \(\mathbb{R}\) and the constraint \(P(x) \leq X \leq Q(x)\), where \(X\) does not appear in any constraints in \(\mathbb{R}\):

a. If \(L\) is decreasing and \(U\) is increasing for variable \(X\), then \([L(x, X), U(x, X)]\) is equal to the variable integer interval \([L(x, Q(x)), U(x, Q(x))]\), where \(x\) in \(\mathbb{R}\) and \(P(x) \leq Q(x)\).

b. If \(L\) is increasing and \(U\) is decreasing for variable \(X\), then \([L(x, X), U(x, X)]\) is equal to the variable integer interval \([L(x, P(x)), U(x, P(x))]\), where \(x\) in \(\mathbb{R}\) and \(P(x) \leq Q(x)\).

c. If \(L\) and \(U\) are increasing for variable \(X\) and \(\min(U(x, X) - L(x, X + 1) + 1) \geq 0\), then \([L(x, X), U(x, X)]\) is equal to the variable integer interval \([L(x, P(x)), U(x, Q(x))]\), where \(x\) in \(\mathbb{R}\) and \(P(x) \leq Q(x)\).

d. If \(L\) and \(U\) are decreasing for variable \(X\) and \(\min(U(x, X + 1) - L(x, X) + 1) \geq 0\), then \([L(x, X), U(x, X)]\) is equal to the variable integer interval \([L(x, Q(x)), U(x, P(x))]\), where \(x\) in \(\mathbb{R}\) and \(P(x) \leq Q(x)\).
General Loop Structure

for \( I_1 = p_1 \) to \( q_1 \) do
  for \( I_2 = p_2(I_1) \) to \( q_2(I_1) \) do
    for \( I_3 = p_3(I_1, I_2) \) to \( q_3(I_1, I_2) \) do
      \[ \ldots \]
    \[ \ldots \]
    for \( I_n = p_n(I_1, I_2, \ldots, I_{n-1}) \) to \( q_n(I_1, I_2, \ldots, I_{n-1}) \) do
      \( s_1: \ A[f(I_1, I_2, \ldots, I_n)] = \ldots \)
      \( s_2: \ldots = A[g(I_1, I_2, \ldots, I_n)] \)
    endfor
  endfor
endfor
Handling General Loop Regions

- General loop nest dependence constraints
  \[ f(I_1, I_2, \ldots, I_n) = g(I'_1, I'_2, \ldots, I'_n) \]
  \[ p_k(I_1, I_2, \ldots, I_{k-1}) \leq I_k \leq q_k(I_1, I_2, \ldots, I_{k-1}) \]
  \[ p_k(I'_1, I'_2, \ldots, I'_{k-1}) \leq I'_k \leq q_k(I'_1, I'_2, \ldots, I'_{k-1}), \quad 1 \leq k \leq n. \]

- The can also be represented as

\[ F(x) = 0 \]
\[ P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x) \quad 1 \leq k \leq n \]
\[ P_{2k}(x) \leq X_{2k} \leq Q_{2k}(x) \quad x = (X_1, X_2, \ldots, X_{2n}), \]
Handling General Loop Regions

- Start from variable interval equation
  \[ F(x) = [0, 0] \]
- Eliminate one variable at a time from
  \[ F(x) = [L(x, X), U(x, X)], P(x) \leq X \leq Q(x) \]
- Use a DAG to determine the order of variable elimination
- Check for Accuracy Condition 1
  \[ L, U \] should have opposite monotonicity for \( X \) or else
  - if they are both increasing, then \( \min(L(x, X) - U(x, X + 1) + 1) \geq 0 \)
  - if they are both decreasing, then \( \min(L(x, X + 1) - U(x, X) + 1) \geq 0 \)
- Check for Accuracy Condition 2
  \[ \min(Q(x) - P(x)) \geq 0 \]
- Compare the final integer interval with zero
Handling Direction Vector
Constraints

- **Additional direction vector constraint**
  \[ P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x) \]
  \[ P_{2k}(x) \leq X_{2k} \leq Q_{2k}(x) \]
  \[ X_{2k-1} < X_{2k} \]

- **Two orders of elimination**
  \[ P_{2k-1}(x) \leq X_{2k-1} \leq \min(Q_{2k-1}(x), Q_{2k}(x) - 1) \]
  \[ \max(P_{2k}(x), X_{2k-1} + 1) \leq X_{2k} \leq Q_{2k}(x) \]

  OR

  \[ P_{2k-1}(x) \leq X_{2k-1} \leq \min(Q_{2k-1}(x), X_{2k} - 1) \]
  \[ \max(P_{2k}(x), P_{2k-1}(x) + 1) \leq X_{2k} \leq Q_{2k}(x) \]

- **Select bounds using the table**
- **Proceed with the NLVI-Test algorithm**
### Conditions for Selecting Direction Vectors

#### $v_k = \ll$:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Bounds</th>
<th>Conditions</th>
<th>Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{2k}(x) \leq P_{2k-1}(x) + 1$</td>
<td></td>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td></td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td>$P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x)$</td>
<td>$P_{2k-1}(x) \leq P_{2k-1}(x) + 1$</td>
<td>$P_{2k-1}(x) \leq X_{2k-1} \leq X_{2k} - 1$</td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td>$X_{2k-1} + 1 \leq X_{2k} \leq Q_{2k}(x)$</td>
<td>$P_{2k}(x) \leq P_{2k-1}(x) + 1$</td>
<td>$P_{2k-1}(x) + 1 \leq X_{2k} \leq Q_{2k}(x)$</td>
</tr>
</tbody>
</table>

#### $v_k = \gg$:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Bounds</th>
<th>Conditions</th>
<th>Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{2k-1}(x) \leq P_{2k}(x) + 1$</td>
<td></td>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td></td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td>$X_{2k} + 1 \leq X_{2k-1} \leq Q_{2k-1}(x)$</td>
<td>$P_{2k}(x) \leq P_{2k}(x) + 1$</td>
<td>$P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x)$</td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x) - 1$</td>
<td>$P_{2k}(x) \leq X_{2k} \leq Q_{2k}(x)$</td>
<td>$P_{2k-1}(x) \leq P_{2k}(x) + 1$</td>
<td>$P_{2k}(x) \leq X_{2k} \leq X_{2k-1} - 1$</td>
</tr>
</tbody>
</table>

#### $v_k = \ll$:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Bounds</th>
<th>Conditions</th>
<th>Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{2k}(x) \leq P_{2k-1}(x)$</td>
<td></td>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x)$</td>
<td></td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x)$</td>
<td>$P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x)$</td>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x)$</td>
<td>$P_{2k}(x) \leq X_{2k} \leq Q_{2k-1}(x)$</td>
</tr>
<tr>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x)$</td>
<td>$P_{2k-1}(x) \leq X_{2k-1} \leq Q_{2k-1}(x)$</td>
<td>$Q_{2k}(x) \leq Q_{2k-1}(x)$</td>
<td>$P_{2k}(x) \leq X_{2k} \leq Q_{2k-1}(x)$</td>
</tr>
</tbody>
</table>
Computation of Minimum and Maximum

- Utilized throughout the NLVI-Test algorithm
- Used to compare expressions.
  \[
  \min(Q(x) - P(x)) \geq 0 \iff Q(x) \geq P(x) \text{ for all } x \text{ in } \mathbb{R}
  \]
  \[
  \max(Q(x) - P(x)) \leq 0 \iff Q(x) \leq P(x) \text{ for all } x \text{ in } \mathbb{R}
  \]
- Utilize a Variable Substitution Algorithm
- Use the monotonicity of the expression
  - if \( F(x, X) \) and is \( \uparrow \) for \( X \) where \( P(x) \leq X \leq Q(x) \)
    then \( \max(F(x, X)) = \max(F(x, Q(x))) \)
    and \( \min(F(x, X)) = \min(F(x, P(x))) \)
  - if \( F(x, X) \) and is \( \downarrow \) for \( X \) where \( P(x) \leq X \leq Q(x) \)
    then \( \max(F(x, X)) = \max(F(x, P(x))) \)
    and \( \min(F(x, X)) = \min(F(x, Q(x))) \)
Computation of Minimum and Maximum

- The computation of monotonicity relies on the computation of min and max
- Each procedure calls the other recursively
  - $F(x, X)$ and is $\uparrow$ for $X$ iff $\min(dF(x, X)) \geq 0$
  - $F(x, X)$ and is $\downarrow$ for $X$ iff $\max(dF(x, X)) \leq 0$
- The halting condition depends on the type of functions used
  - For polynomials the level of recursion is the degree of the polynomial
- A DAG is used to determine the order of substitution
  - Any topological short is a valid sequence
Coupled Subscripts and If-Statements

- Apply the equation propagation technique while considering if-statement constraints:
  
  ```
  if(N > 1) then
    for I = 1 to N do
      for J = 1 to I*I do
      endfor
    endfor
  endif
  ```

- The dependence problem constraints:
  
  \[
  \begin{align*}
  2NX_1 - X_3 - 2X_4 + N &= 0 & 2 \leq N \leq +\infty \\
  NX_1 - X_3 - X_4 &= -1 & 1 \leq X_1 \leq N, \quad 1 \leq X_2 \leq N \\
  1 \leq X_3 \leq X_1^2, \quad 1 \leq X_4 \leq X_2^2
  \end{align*}
  \]

- Equation propagation produces
  
  \[N + X_3 = 2\]

- Dependence is disproved
DO n0 = 1, n, 1
    t(n0) = t(n0) + (-c(n0))*c0
    t(n+n0) = t(n+n0) + (-c(n0))*c1
    t(n0+2*n) = t(n0+2*n) + (-c(n0))*c2
    t(n0+3*n) = t(n0+3*n) + (-c(n0))*c3
    t(n0+4*n) = t(n0+4*n) + (-c(n0))*c4
    t(n0+5*n) = t(n0+5*n) + (-c(n0))*c5
ENDDO

- **BDNA of Perfect**
  - Symbolic Loop Bounds
  - I-Test cannot parallelize
  - NLVI-Test, Omega and Range can parallelize loop and increase speedup
Real Code Examples

```
DO mrs = 1, (num*(num+1))/2, 1
  DO mi = 1, num, 1
    DO mj = 1, mi, 1
      xrsij(mj+(mi**2-mi-num-num**2+mrs*num+mrs*num**2)/2) = xij(mj)
    ENDDO
  ENDDO
ENDDO
ENDDO
ENDDO
```

- **TRFD of Perfect**
  - Non-Linear Expressions
  - I-Test, Omega cannot parallelize
  - NLVI-Test and Range can parallelize loops and produce speedup
Real Code Examples

DO k = 2, lmi-1, 1
   DO i3 = 1, mm(k), 1
      DO i2 = 1, mm(k), 1
         DO i1 = 1, mm(k), 1
            u(-1+ir(k)+i1-mm(k)-mm(k)**2+mm(k)*i2+i3*mm(k)**2) = 0
         ENDDO
      ENDDO
   ENDDO
ENDDO
ENDDO
ENDDO
ENDDO

● MGRID of SPEC
   - Non-Affine loop bounds and Non-Linear expression
   - I-Test, Omega and Range cannot parallelize loop i3
   - NLVI-Test can parallelize i3 and increase speedup
Real Code Examples

DO j = 3, n-1, 1
   DO i = 2, n-1, 1
      r = aa(i,j)*d(i,j-1)
      d(i,j) = 1.D0/(dd(i,j)-aa(i,j-1)*r)
      rx(i,j) = rx(i,j)-rx(i,j-1)*r
      ry(i,j) = ry(i,j)-ry(i,j-1)*r
   ENDDO
ENDDO

- **TOMCATV of SPEC**
  - Dependence with (<, =) direction vector
  - Range test detects only (<, *)
  - Range test cannot interchange loops
  - NLVI-Test, I-Test and Omega interchange loop and increase speedup
We implemented a portable library that contains several data dependence tests. The PLATO library contains:
- Banerjee Test
- I-Test
- VI-Test for linear expressions
- NLVI-Test for polynomial and rational polynomial expressions

We used the Polaris Compiler Infrastructure

Programs were transformed
- Constant Propagation
- Induction variable recognition
- Reduction
- Scalar and array privatization
- Loop Normalization (etc)

We also implemented a simple loop interchange transformation
Experiments

- We compare 4 Data Dependence tests
  - I-Test
  - NLVI-Test
  - Omega
  - Range

- We compare in terms of
  - Data Dependence Accuracy
  - Compilation efficiency
  - Loop Parallelization
  - Program Execution Performance

- We use two Benchmark Suites
  - The Perfect Benchmarks
  - The SPEC Benchmarks
Data Dependence Accuracy in the Perfect Benchmarks

- I-Test: 30% Independent, 8% Dependent, 62% Maybe
- NLVI-Test: 56% Independent, 12% Dependent, 32% Maybe
- Omega: 35% Independent, 11% Dependent, 54% Maybe
- Range: 29% Independent, 71% Dependent

PR: 64% Independent, 61% Dependent
DV: 67% Independent, 67% Dependent
Data Dependence Accuracy in the SPEC Benchmarks

- **I-Test**
  - Independent: 45%
  - Dependent: 7%
  - Maybe: 8%

- **NLVI-Test**
  - Independent: 48%
  - Dependent: 42%
  - Maybe: 8%

- **Omega**
  - Independent: 50%
  - Dependent: 8%
  - Maybe: 8%

- **Range**
  - Independent: 53%
  - Dependent: 52%
  - Maybe: 48%

- **I-Test**
  - Independent: 32%
  - Dependent: 1%
  - Maybe: 67%

- **NLVI-Test**
  - Independent: 31%
  - Dependent: 1%
  - Maybe: 68%

- **Omega**
  - Independent: 31%
  - Dependent: 1%
  - Maybe: 68%

- **Range**
  - Independent: 47%
  - Dependent: 1%
  - Maybe: 53%
Compilation Efficiency in the Perfect and SPEC Benchmarks

<table>
<thead>
<tr>
<th>Test</th>
<th>Comp Time</th>
<th>DD Time</th>
<th>Test Time</th>
<th>I-Test</th>
<th>Comp Time</th>
<th>DD Time</th>
<th>Test Time</th>
<th>Omega</th>
<th>Comp Time</th>
<th>DD Time</th>
<th>Test Time</th>
<th>Omega</th>
<th>Comp Time</th>
<th>DD Time</th>
<th>Test Time</th>
<th>Range</th>
<th>Comp Time</th>
<th>DD Time</th>
<th>Test Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLVI-Test</td>
<td>100%</td>
<td>41%</td>
<td>100%</td>
<td>NLVI-Test</td>
<td>100%</td>
<td>31%</td>
<td>100%</td>
<td>Omega</td>
<td>100%</td>
<td>20%</td>
<td>100%</td>
<td>90%</td>
<td>100%</td>
<td>89%</td>
<td>100%</td>
<td>Range</td>
<td>100%</td>
<td>12%</td>
<td>100%</td>
</tr>
<tr>
<td>I-Test</td>
<td>40.5</td>
<td>6.7</td>
<td>1.0</td>
<td>I-Test</td>
<td>14.0</td>
<td>2.8</td>
<td>1.5</td>
<td>Omega</td>
<td>109.9</td>
<td>16.7</td>
<td>15.4</td>
<td>90%</td>
<td>109.9</td>
<td>98.6</td>
<td>95.3</td>
<td>Omega</td>
<td>109.9</td>
<td>16.7</td>
<td>15.4</td>
</tr>
<tr>
<td>Omega</td>
<td>122.9</td>
<td>89.2</td>
<td>83.4</td>
<td>Omega</td>
<td>122.9</td>
<td>89.2</td>
<td>83.4</td>
<td>Omega</td>
<td>122.9</td>
<td>89.2</td>
<td>83.4</td>
<td>Omega</td>
<td>122.9</td>
<td>89.2</td>
<td>83.4</td>
<td>Omega</td>
<td>122.9</td>
<td>89.2</td>
<td>83.4</td>
</tr>
<tr>
<td>Range</td>
<td>41.6</td>
<td>7.8</td>
<td>2.1</td>
<td>Range</td>
<td>14.1</td>
<td>2.8</td>
<td>1.5</td>
<td>Range</td>
<td>14.1</td>
<td>2.8</td>
<td>1.5</td>
<td>Range</td>
<td>14.1</td>
<td>2.8</td>
<td>1.5</td>
<td>Range</td>
<td>14.1</td>
<td>2.8</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Loop Parallelization in the Perfect and SPEC Benchmarks

<table>
<thead>
<tr>
<th>Test</th>
<th>Total Loops</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>4118 (100%)</td>
<td>2253 (55%)</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>4118 (100%)</td>
<td>2369 (58%)</td>
</tr>
<tr>
<td>Omega</td>
<td>4118 (100%)</td>
<td>2295 (56%)</td>
</tr>
<tr>
<td>Range</td>
<td>4118 (100%)</td>
<td>2335 (57%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>Total Loops</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>1729 (100%)</td>
<td>1040 (60%)</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>1729 (100%)</td>
<td>1163 (67%)</td>
</tr>
<tr>
<td>Omega</td>
<td>1729 (100%)</td>
<td>1069 (62%)</td>
</tr>
<tr>
<td>Range</td>
<td>1729 (100%)</td>
<td>1094 (63%)</td>
</tr>
</tbody>
</table>
Execution Performance in MDG of the Perfect Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc</th>
<th>sp</th>
<th>3-proc</th>
<th>sp</th>
<th>4-proc</th>
<th>sp</th>
<th>5-proc</th>
<th>sp</th>
<th>6-proc</th>
<th>sp</th>
<th>7-proc</th>
<th>sp</th>
<th>8-proc</th>
<th>sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>serial</td>
<td>51.33</td>
<td>28.76</td>
<td>1.8</td>
<td>19.48</td>
<td>2.6</td>
<td>14.90</td>
<td>3.4</td>
<td>12.08</td>
<td>4.2</td>
<td>10.23</td>
<td>5.0</td>
<td>9.26</td>
<td>5.5</td>
<td>7.96</td>
<td>6.4</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>serial</td>
<td>51.33</td>
<td>28.97</td>
<td>1.8</td>
<td>19.63</td>
<td>2.6</td>
<td>15.01</td>
<td>3.4</td>
<td>12.16</td>
<td>4.2</td>
<td>10.31</td>
<td>5.0</td>
<td>9.35</td>
<td>5.5</td>
<td>8.00</td>
<td>6.4</td>
</tr>
<tr>
<td>MDG</td>
<td>Omega</td>
<td>51.33</td>
<td>28.69</td>
<td>1.8</td>
<td>19.43</td>
<td>2.6</td>
<td>14.88</td>
<td>3.4</td>
<td>12.09</td>
<td>4.2</td>
<td>10.22</td>
<td>5.0</td>
<td>9.27</td>
<td>5.5</td>
<td>7.97</td>
<td>6.4</td>
</tr>
<tr>
<td>MDG</td>
<td>Range</td>
<td>51.33</td>
<td>29.03</td>
<td>1.8</td>
<td>19.65</td>
<td>2.6</td>
<td>15.00</td>
<td>3.4</td>
<td>12.16</td>
<td>4.2</td>
<td>10.31</td>
<td>5.0</td>
<td>9.33</td>
<td>5.5</td>
<td>8.00</td>
<td>6.4</td>
</tr>
</tbody>
</table>
Execution Performance in BDNA of the Perfect Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc</th>
<th>sp</th>
<th>3-proc</th>
<th>sp</th>
<th>4-proc</th>
<th>sp</th>
<th>5-proc</th>
<th>sp</th>
<th>6-proc</th>
<th>sp</th>
<th>7-proc</th>
<th>sp</th>
<th>8-proc</th>
<th>sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>serial</td>
<td>8.73</td>
<td>5.71</td>
<td>1.5</td>
<td>4.22</td>
<td>2.1</td>
<td>3.49</td>
<td>2.5</td>
<td>3.07</td>
<td>2.8</td>
<td>2.79</td>
<td>3.1</td>
<td>2.61</td>
<td>3.3</td>
<td>2.47</td>
<td>3.5</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>serial</td>
<td>8.73</td>
<td>5.58</td>
<td>1.6</td>
<td>4.11</td>
<td>2.1</td>
<td>3.35</td>
<td>2.6</td>
<td>2.93</td>
<td>3.0</td>
<td>2.64</td>
<td>3.3</td>
<td>2.45</td>
<td>3.6</td>
<td>2.39</td>
<td>3.7</td>
</tr>
<tr>
<td>BDNA</td>
<td>Omega</td>
<td>8.73</td>
<td>5.59</td>
<td>1.6</td>
<td>4.15</td>
<td>2.1</td>
<td>3.34</td>
<td>2.6</td>
<td>2.89</td>
<td>3.0</td>
<td>2.66</td>
<td>3.3</td>
<td>2.42</td>
<td>3.6</td>
<td>2.35</td>
<td>3.7</td>
</tr>
<tr>
<td>Range</td>
<td>serial</td>
<td>8.73</td>
<td>5.72</td>
<td>1.5</td>
<td>4.21</td>
<td>2.1</td>
<td>3.46</td>
<td>2.5</td>
<td>2.98</td>
<td>2.9</td>
<td>2.68</td>
<td>3.3</td>
<td>2.49</td>
<td>3.5</td>
<td>2.38</td>
<td>3.7</td>
</tr>
</tbody>
</table>
Execution Performance in ARC2D of the Perfect Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc</th>
<th>sp</th>
<th>3-proc</th>
<th>sp</th>
<th>4-proc</th>
<th>sp</th>
<th>5-proc</th>
<th>sp</th>
<th>6-proc</th>
<th>sp</th>
<th>7-proc</th>
<th>sp</th>
<th>8-proc</th>
<th>sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>serial</td>
<td>29.93</td>
<td>23.10</td>
<td>1.3</td>
<td>17.97</td>
<td>1.7</td>
<td>13.94</td>
<td>2.1</td>
<td>12.19</td>
<td>2.5</td>
<td>11.12</td>
<td>2.7</td>
<td>10.35</td>
<td>2.9</td>
<td>9.84</td>
<td>3.0</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>serial</td>
<td>29.93</td>
<td>23.15</td>
<td>1.3</td>
<td>17.93</td>
<td>1.7</td>
<td>13.96</td>
<td>2.1</td>
<td>12.30</td>
<td>2.4</td>
<td>11.08</td>
<td>2.7</td>
<td>10.40</td>
<td>2.9</td>
<td>9.86</td>
<td>3.0</td>
</tr>
<tr>
<td>ARC2D</td>
<td>Omega</td>
<td>29.93</td>
<td>23.18</td>
<td>1.3</td>
<td>17.92</td>
<td>1.7</td>
<td>13.96</td>
<td>2.1</td>
<td>12.22</td>
<td>2.4</td>
<td>11.13</td>
<td>2.7</td>
<td>10.36</td>
<td>2.9</td>
<td>9.91</td>
<td>3.0</td>
</tr>
<tr>
<td>Range</td>
<td>serial</td>
<td>29.93</td>
<td>23.10</td>
<td>1.3</td>
<td>17.88</td>
<td>1.7</td>
<td>13.85</td>
<td>2.2</td>
<td>12.02</td>
<td>2.5</td>
<td>10.98</td>
<td>2.7</td>
<td>10.36</td>
<td>2.9</td>
<td>10.00</td>
<td>3.0</td>
</tr>
</tbody>
</table>
### Execution Performance in FLO52 of the Perfect Benchmarks

**Program** | **Test** | **serial** | **2-proc** | **sp** | **3-proc** | **sp** | **4-proc** | **sp** | **5-proc** | **sp** | **6-proc** | **sp** | **7-proc** | **sp** | **8-proc** | **sp** |
--- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
I-Test | 3.77 | 3.61 | 1.0 | 3.40 | 1.1 | 2.86 | 1.3 | 2.85 | 1.3 | 2.80 | 1.3 | 2.78 | 1.4 | 2.78 | 1.4 |
RVI-Test | 3.77 | 3.62 | 1.0 | 3.40 | 1.1 | 2.85 | 1.3 | 2.86 | 1.3 | 2.77 | 1.4 | 2.77 | 1.4 | 2.74 | 1.4 |
FLO52 | 3.77 | 3.60 | 1.0 | 3.40 | 1.1 | 2.84 | 1.3 | 2.86 | 1.3 | 2.80 | 1.3 | 2.80 | 1.3 | 2.77 | 1.4 |
Omega | 3.77 | 3.59 | 1.1 | 3.40 | 1.1 | 2.84 | 1.3 | 2.84 | 1.3 | 2.81 | 1.3 | 2.80 | 1.3 | 2.78 | 1.4 |
Range | 3.77 | 3.59 | 1.1 | 3.40 | 1.1 | 2.84 | 1.3 | 2.84 | 1.3 | 2.81 | 1.3 | 2.80 | 1.3 | 2.78 | 1.4 |
Execution Performance in TRFD of the Perfect Benchmarks

Program | Test        | serial | 2-proc | sp  | 3-proc | sp  | 4-proc | sp  | 5-proc | sp  | 6-proc | sp  | 7-proc | sp  | 8-proc | sp
---|-------------|--------|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|
NLVI-Test | serial      | 4.68   | 3.42   | 1.4 | 2.45   | 1.9 | 2.00   | 2.3 | 1.72   | 2.7 | 1.53   | 3.1 | 1.39   | 3.4 | 1.29   | 3.6|
TRFD Range | serial      | 4.68   | 3.43   | 1.4 | 2.44   | 1.9 | 2.01   | 2.3 | 1.72   | 2.7 | 1.53   | 3.1 | 1.40   | 3.3 | 1.29   | 3.6|
### Execution Performance in HYDRO2D of the SPEC Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc sp</th>
<th>3-proc sp</th>
<th>4-proc sp</th>
<th>5-proc sp</th>
<th>6-proc sp</th>
<th>7-proc sp</th>
<th>8-proc sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td></td>
<td>172.99</td>
<td>109.27</td>
<td>78.72</td>
<td>62.86</td>
<td>53.39</td>
<td>47.74</td>
<td>43.65</td>
<td>40.70</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td></td>
<td>172.99</td>
<td>104.11</td>
<td>74.11</td>
<td>58.30</td>
<td>49.01</td>
<td>43.21</td>
<td>39.31</td>
<td>36.29</td>
</tr>
<tr>
<td>HYDRO2D</td>
<td>Omega</td>
<td>172.99</td>
<td>109.67</td>
<td>78.58</td>
<td>62.70</td>
<td>53.26</td>
<td>47.86</td>
<td>43.69</td>
<td>40.70</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>172.99</td>
<td>104.34</td>
<td>74.15</td>
<td>58.54</td>
<td>48.93</td>
<td>43.37</td>
<td>39.28</td>
<td>36.18</td>
</tr>
</tbody>
</table>
## Execution Performance in MGRID of the SPEC Benchmarks

### Program Test Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc sp</th>
<th>3-proc sp</th>
<th>4-proc sp</th>
<th>5-proc sp</th>
<th>6-proc sp</th>
<th>7-proc sp</th>
<th>8-proc sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>serial</td>
<td>119.20</td>
<td>66.39 1.8</td>
<td>58.44 2.0</td>
<td>47.67 2.5</td>
<td>42.95 2.8</td>
<td>40.26 3.0</td>
<td>38.78 3.1</td>
<td>37.28 3.2</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>serial</td>
<td>119.20</td>
<td>62.84 1.9</td>
<td>54.07 2.2</td>
<td>43.05 2.8</td>
<td>37.83 3.2</td>
<td>34.95 3.4</td>
<td>32.63 3.7</td>
<td>30.82 3.9</td>
</tr>
<tr>
<td>MGRID</td>
<td>Omega</td>
<td>119.20</td>
<td>66.61 1.8</td>
<td>58.34 2.0</td>
<td>47.64 2.5</td>
<td>42.90 2.8</td>
<td>40.35 3.0</td>
<td>38.68 3.1</td>
<td>37.43 3.2</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td>119.20</td>
<td>65.89 1.8</td>
<td>57.80 2.1</td>
<td>47.08 2.5</td>
<td>42.61 2.8</td>
<td>39.99 3.0</td>
<td>38.79 3.1</td>
<td>37.15 3.2</td>
</tr>
</tbody>
</table>

### Graph: Execution Performance in MGRID of the SPEC Benchmarks

The graph above illustrates the execution performance of various programs and tests on MGRID. The X-axis represents the number of processors used, ranging from serial to 8-processors (sp). The Y-axis represents the execution time in seconds. Different tests and programs are color-coded for easy comparison.

- **I-Test**: Demonstrates the performance of the I-Test with decreasing execution time as the number of processors increases.
- **NLVI-Test**: Shows the performance of the NLVI-Test with a similar trend to I-Test.
- **Omega**: Illustrates the Omega test's performance, indicating a consistent decrease in execution time.
- **Range**: Represents the Range test, also showing a steady drop in execution time with increased processors.

The data points for each test across different processor counts provide a clear visualization of how execution time varies under different conditions.
Execution Performance in SU2COR of the SPEC Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc</th>
<th>3-proc</th>
<th>4-proc</th>
<th>5-proc</th>
<th>6-proc</th>
<th>7-proc</th>
<th>8-proc</th>
<th>8-proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td></td>
<td>119.33</td>
<td>69.51</td>
<td>51.54</td>
<td>41.85</td>
<td>37.82</td>
<td>35.07</td>
<td>34.34</td>
<td>32.86</td>
<td>3.6</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td></td>
<td>119.33</td>
<td>69.79</td>
<td>51.35</td>
<td>41.80</td>
<td>37.87</td>
<td>35.14</td>
<td>34.19</td>
<td>33.05</td>
<td>3.6</td>
</tr>
<tr>
<td>SU2COR</td>
<td>Omega</td>
<td>119.33</td>
<td>69.72</td>
<td>51.62</td>
<td>41.83</td>
<td>37.76</td>
<td>35.07</td>
<td>34.17</td>
<td>32.94</td>
<td>3.6</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td>119.33</td>
<td>70.15</td>
<td>51.60</td>
<td>42.21</td>
<td>38.14</td>
<td>35.43</td>
<td>34.27</td>
<td>33.35</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Execution Performance in SWIM of the SPEC Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc sp</th>
<th>3-proc sp</th>
<th>4-proc sp</th>
<th>5-proc sp</th>
<th>6-proc sp</th>
<th>7-proc sp</th>
<th>8-proc sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>86.69</td>
<td>44.12</td>
<td>28.98</td>
<td>21.72</td>
<td>17.11</td>
<td>14.18</td>
<td>11.86</td>
<td>9.84</td>
<td></td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>86.69</td>
<td>44.44</td>
<td>29.13</td>
<td>21.74</td>
<td>17.16</td>
<td>14.27</td>
<td>11.88</td>
<td>9.88</td>
<td></td>
</tr>
<tr>
<td>SWIM</td>
<td>Omega</td>
<td>86.69</td>
<td>44.24</td>
<td>28.84</td>
<td>21.88</td>
<td>17.15</td>
<td>14.22</td>
<td>11.92</td>
<td>9.82</td>
</tr>
<tr>
<td>Range</td>
<td>86.69</td>
<td>44.39</td>
<td>29.17</td>
<td>21.78</td>
<td>17.16</td>
<td>14.21</td>
<td>11.82</td>
<td>9.89</td>
<td></td>
</tr>
</tbody>
</table>
Execution Performance in TOMCATV of the SPEC Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Test</th>
<th>serial</th>
<th>2-proc</th>
<th>3-proc sp</th>
<th>4-proc sp</th>
<th>5-proc sp</th>
<th>6-proc sp</th>
<th>7-proc sp</th>
<th>8-proc sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Test</td>
<td>serial</td>
<td>30.86</td>
<td>32.59</td>
<td>0.9</td>
<td>24.22</td>
<td>1.3</td>
<td>19.43</td>
<td>1.6</td>
<td>16.89</td>
</tr>
<tr>
<td>NLVI-Test</td>
<td>serial</td>
<td>30.86</td>
<td>31.85</td>
<td>1.0</td>
<td>24.14</td>
<td>1.3</td>
<td>19.40</td>
<td>1.6</td>
<td>16.87</td>
</tr>
<tr>
<td>TOMCATV</td>
<td>Omega</td>
<td>30.86</td>
<td>31.77</td>
<td>1.0</td>
<td>24.19</td>
<td>1.3</td>
<td>19.42</td>
<td>1.6</td>
<td>16.87</td>
</tr>
<tr>
<td>Range</td>
<td>serial</td>
<td>30.86</td>
<td>25.29</td>
<td>1.2</td>
<td>20.69</td>
<td>1.5</td>
<td>18.36</td>
<td>1.7</td>
<td>17.49</td>
</tr>
</tbody>
</table>
Conclusions

- Traditional data dependence analysis techniques assume simple source code when in fact real source is complex.
- Large amount of potential parallelism remain unexploited.
- Other techniques have been proposed to address most of the issues.
- Each technique has its own advantages and limitations.
- We created a new data dependence test that manages to combine the advantages and overcome the limitations.
Conclusions

- Defined conditions that guarantee integer solution to a data dependence problem even in presence of non-linear constraints.
- It can address the issues of:
  - Non-Linear expressions
  - If-Statement conditions
  - Coupled array subscripts
  - Complex loop bounds
  - Prove Definite Dependence
  - Produce complete direction vector information
  - Has polynomial time complexity
Conclusions

- **NLVI-Test**
  - Is more accurate than all polynomial time tests
  - Is much faster than the Omega test
  - Produced the highest degree of parallelization
  - Increased the execution performance in several benchmarks

- In order to achieve better parallelization non-linear expressions need to be taken into account

- Compilation efficiency is essential