

Efficient Techniques for Advanced Data Dependence Analysis

PACT 2005

Konstantinos Kyriakopoulos and
Kleanthis Psarris

Department of Computer Science
The University of Texas at San Antonio
San Antonio, TX 78249



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

$$[L, U] = \{L, L + 1, \dots, U\}$$

e.g. $[1, 4] = \{1, 2, 3, 4\}$

● Single Variable Integer Interval

$$[L(X), U(X)] \text{ where } P \leq X \leq Q = \\ [L(P), U(P)] \cup [L(P + 1), U(P + 1)] \cup \dots \cup \\ [L(Q), U(Q)]$$

e.g. $[X, 3X], 1 \leq X \leq 3 = \\ [1, 3] \cup [2, 6] \cup [3, 9] = \\ [1, 9] = \{1, 2, \dots, 9\}.$

● General Variable Integer Interval

$$[L(x), U(x)] \text{ where } x \text{ in } \mathfrak{R} = \\ \bigcup_{x_i \in \mathfrak{R}} [L(x_i), U(x_i)]$$

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 to the 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 to the integer interval $[L(Q), U(P)]$.

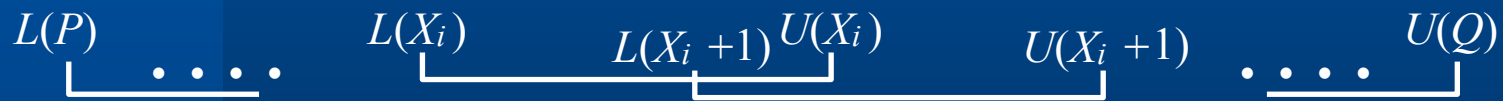
Intuition Behind Theorem



a. $L \downarrow U \uparrow$



b. $L \uparrow U \downarrow$



c. $L \uparrow U \uparrow$



d. $L \downarrow U \downarrow$

Monotonicity of Multi-variable Functions

- Increasing for variable function $F(\mathbf{x}, X)$ for variable X
 - $F(\mathbf{x}_i, X_j) \leq F(\mathbf{x}_i, X_j + 1)$ for any $\mathbf{x}_i \in Z^n$ and for any $X_j \in Z$
- Decreasing for variable function $F(\mathbf{x}, X)$ for variable X
 - $F(\mathbf{x}_i, X_j) \geq F(\mathbf{x}_i, X_j + 1)$ for any $\mathbf{x}_i \in Z^n$ 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 (\mathbf{x}, X) in that region
- F can be proven increasing if $\min(F(\mathbf{x}, X + 1) - F(\mathbf{x}, X)) \geq 0$
- F can be proven decreasing if $\max(F(\mathbf{x}, X + 1) - F(\mathbf{x}, X)) \leq 0$
- For more efficiency we can use the first partial derivative
 - $\min(dF(\mathbf{x}, X)/dX) \geq 0$: Increasing
 - $\max(dF(\mathbf{x}, X)/dX) \leq 0$: Decreasing

Mutli-variable Interval Theorem

Given a variable integer interval $[L(\mathbf{x}, X), U(\mathbf{x}, X)]$ subject to a set of constraints on \mathbf{x} in \mathfrak{R} and the constraint $P(\mathbf{x}) \leq X \leq Q(\mathbf{x})$, where X does not appear in any constraints in \mathfrak{R} :

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

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
      .
      .
      for  $I_n = p_n(I_1, I_2, \dots, I_{n-1})$  to  $q_n(I_1, I_2, \dots, I_{n-1})$  do
 $s_1:$            $A[f(I_1, I_2, \dots, I_n)] = \dots$ 
 $s_2:$            $\dots = A[g(I_1, I_2, \dots, I_n)]$ 
      endfor
    .
    .
  endfor
endfor
endfor
```

Handling General Loop Regions

- **General loop nest dependence constraints**

$$f(I_1, I_2, \dots, I_n) = g(I'_1, I'_2, \dots, I'_n)$$

$$p_k(I_1, I_2, \dots, I_{k-1}) \leq I_k \leq q_k(I_1, I_2, \dots, I_{k-1})$$

$$p_k(I'_1, I'_2, \dots, I'_{k-1}) \leq I'_k \leq q_k(I'_1, I'_2, \dots, I'_{k-1}), \quad 1 \leq k \leq n.$$

- **The can also be represented as**

$$F(\mathbf{x}) = 0$$

$$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x}) \quad 1 \leq k \leq n$$

$$P_{2k}(\mathbf{x}) \leq X_{2k} \leq Q_{2k}(\mathbf{x}) \quad \mathbf{x} = (X_1, X_2, \dots, X_{2n}),$$

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**

$$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$$

$$P_{2k}(\mathbf{x}) \leq X_{2k} \leq Q_{2k}(\mathbf{x})$$

$$X_{2k-1} < X_{2k} \quad 1 \leq k \leq n$$

- **Two orders of elimination**

$$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq \min(Q_{2k-1}(\mathbf{x}), Q_{2k}(\mathbf{x}) - 1)$$

$$\max(P_{2k}(\mathbf{x}), X_{2k-1} + 1) \leq X_{2k} \leq Q_{2k}(\mathbf{x})$$

OR

$$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq \min(Q_{2k-1}(\mathbf{x}), X_{2k} - 1)$$

$$\max(P_{2k}(\mathbf{x}), P_{2k-1}(\mathbf{x}) + 1) \leq X_{2k} \leq Q_{2k}(\mathbf{x})$$

- **Select bounds using the table**
- **Proceed with the NLVI-Test algorithm**

Conditions for Selecting Direction Vectors

$v_k = "<"$			
Conditions	Bounds	Conditions	Bounds
$P_{2k}(\mathbf{x}) \leq P_{2k-1}(\mathbf{x}) + 1$		$Q_{2k-1}(\mathbf{x}) \leq Q_{2k}(\mathbf{x}) - 1$	
$Q_{2k-1}(\mathbf{x}) \leq Q_{2k}(\mathbf{x}) - 1$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k}(\mathbf{x}) - 1$ $X_{2k-1} + 1 \leq X_{2k} \leq Q_{2k}(\mathbf{x})$	$P_{2k}(\mathbf{x}) \leq P_{2k-1}(\mathbf{x}) + 1$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq X_{2k} - 1$ $P_{2k-1}(\mathbf{x}) + 1 \leq X_{2k} \leq Q_{2k}(\mathbf{x})$
$Q_{2k-1}(\mathbf{x}) \leq Q_{2k}(\mathbf{x}) - 1$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$ $X_{2k-1} + 1 \leq X_{2k} \leq Q_{2k}(\mathbf{x})$	$P_{2k}(\mathbf{x}) \leq P_{2k-1}(\mathbf{x}) + 1$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq X_{2k} - 1$ $P_{2k}(\mathbf{x}) \leq X_{2k} \leq Q_{2k}(\mathbf{x})$
$v_k = ">"$			
Conditions	Bounds	Conditions	Bounds
$P_{2k-1}(\mathbf{x}) \leq P_{2k}(\mathbf{x}) + 1$		$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x}) - 1$	
$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x}) - 1$	$X_{2k} + 1 \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$ $P_{2k}(\mathbf{x}) \leq X_{2k} \leq Q_{2k-1}(\mathbf{x}) - 1$	$P_{2k-1}(\mathbf{x}) \leq P_{2k}(\mathbf{x}) + 1$	$P_{2k}(\mathbf{x}) + 1 \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$ $P_{2k}(\mathbf{x}) \leq X_{2k} \leq X_{2k-1} - 1$
$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x}) - 1$	$X_{2k} + 1 \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$ $P_{2k}(\mathbf{x}) \leq X_{2k} \leq Q_{2k}(\mathbf{x})$	$P_{2k-1}(\mathbf{x}) \leq P_{2k}(\mathbf{x}) + 1$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$ $P_{2k}(\mathbf{x}) \leq X_{2k} \leq X_{2k-1} - 1$
$v_k = "="$			
Conditions	Bounds	Conditions	Bounds
$P_{2k}(\mathbf{x}) \leq P_{2k-1}(\mathbf{x})$		$P_{2k}(\mathbf{x}) \leq P_{2k-1}(\mathbf{x})$	
$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x})$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$	$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x})$	$P_{2k}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k-1}(\mathbf{x})$
$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x})$	$P_{2k-1}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k}(\mathbf{x})$	$Q_{2k}(\mathbf{x}) \leq Q_{2k-1}(\mathbf{x})$	$P_{2k}(\mathbf{x}) \leq X_{2k-1} \leq Q_{2k}(\mathbf{x})$

Computation of Minimum and Maximum

- Utilized throughout the NLVI-Test algorithm
- Used to compare expressions.

$$\min(Q(x) - P(x)) \geq 0 \Leftrightarrow Q(x) \geq P(x) \text{ for all } x \text{ in } \mathcal{R}$$

$$\max(Q(x) - P(x)) \leq 0 \Leftrightarrow Q(x) \leq P(x) \text{ for all } x \text{ in } \mathcal{R}$$

- Utilize a Variable Substitution Algorithm
- Use the monotonicity of the expression

– if $F(x, X)$ and is \uparrow for X where $P(x) \leq X \leq Q(x)$

$$\text{then } \max(F(x, X)) = \max(F(x, Q(x)))$$

$$\text{and } \min(F(x, X)) = \min(F(x, P(x)))$$

– if $F(x, X)$ and is \downarrow for X where $P(x) \leq X \leq Q(x)$

$$\text{then } \max(F(x, X)) = \max(F(x, P(x)))$$

$$\text{and } \min(F(x, X)) = \min(F(x, Q(x)))$$

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 sort 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
      A[2*N*I - J + N, N*I - J + 1] = A[2*J, J]
    endfor
  endfor
endif
```

- The dependence problem constraints:

$$\begin{array}{lll} 2NX_1 - X_3 - 2X_4 + N = 0 & 2 \leq N \leq +\infty & \\ NX_1 - X_3 - X_4 = -1 & 1 \leq X_1 \leq N, & 1 \leq X_2 \leq N \\ & 1 \leq X_3 \leq X_1^2, & 1 \leq X_4 \leq X_2^2 \end{array}$$

- Equation propagation produces

$$N + X_3 = 2$$

- Dependence is disproved

Real Code Examples

```
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
```

- **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
```

- **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

Implementation Details

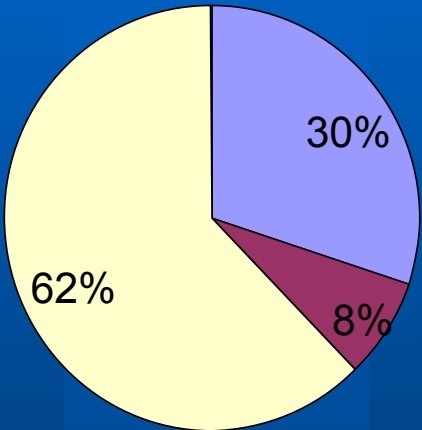
- **We implemented a portable library that contains several data dependence tests. The PLATO library contains:**
 - Banerjee Test
 - I-Test
 - VI-Test for linear expressions
 - NLVI-Test for polynomial and rational polynomial expressions
- **We used the Polaris Compiler Infrastructure**
- **Programs were transformed**
 - Constant Propagation
 - Induction variable recognition
 - Reduction
 - Scalar and array privatization
 - Loop Normalization (etc)
- **We also implemented a simple loop interchange transformation**

Experiments

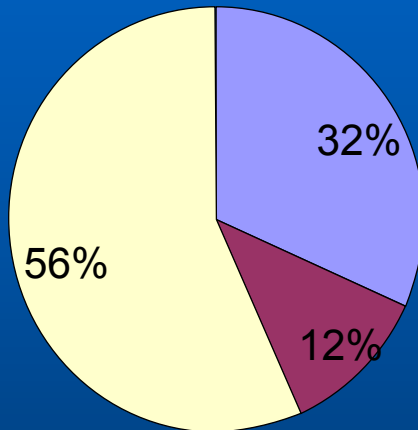
- **We compare 4 Data Dependence tests**
 - I-Test
 - NLVI-Test
 - Omega
 - Range
- **We compare in terms of**
 - Data Dependence Accuracy
 - Compilation efficiency
 - Loop Parallelization
 - Program Execution Performance
- **We use two Benchmark Suites**
 - The Perfect Benchmarks
 - The SPEC Benchmarks

Data Dependence Accuracy in the Perfect Benchmarks

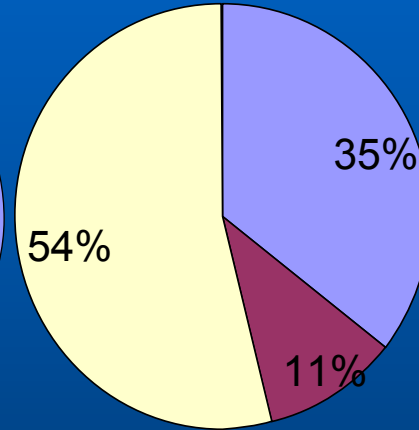
I-Test



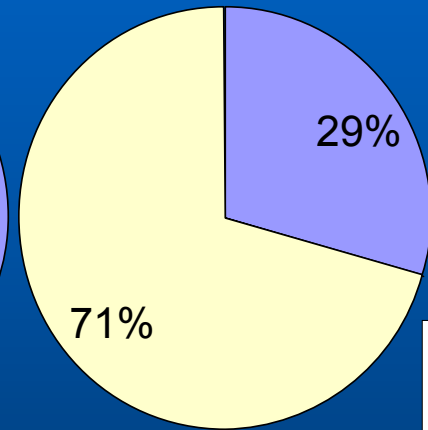
NLVI-Test



Omega



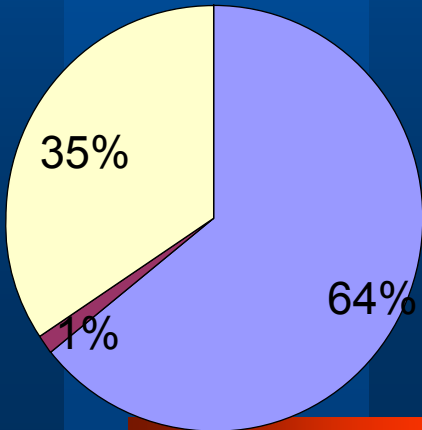
Range



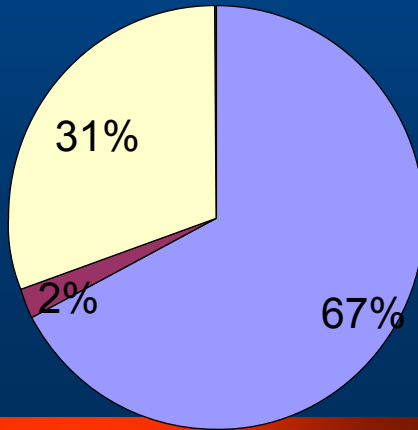
PR



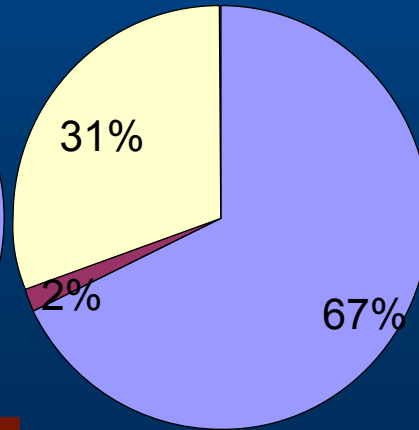
I-Test



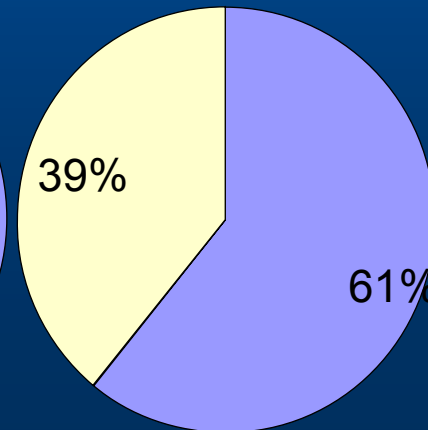
NLVI-Test



Omega



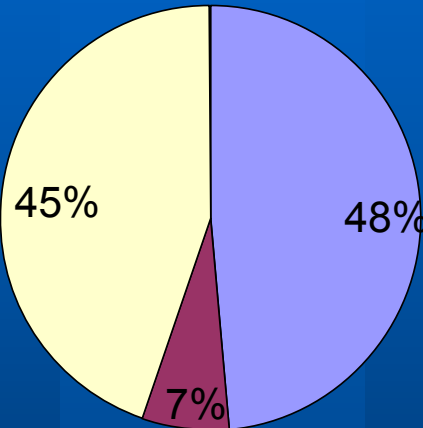
Range



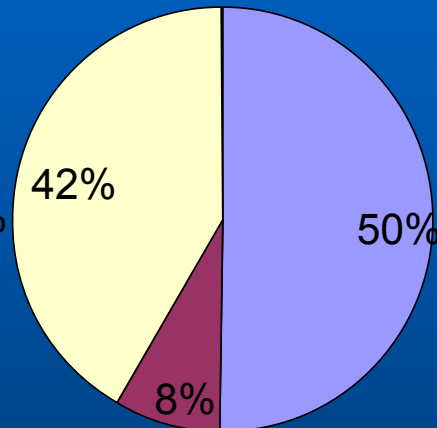
DV

Data Dependence Accuracy in the SPEC Benchmarks

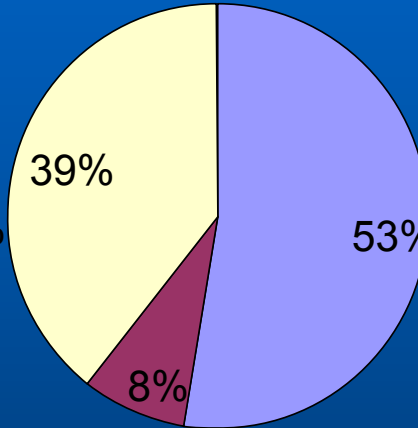
I-Test



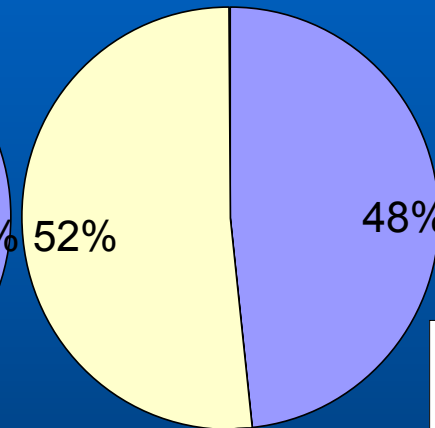
NLVI-Test



Omega



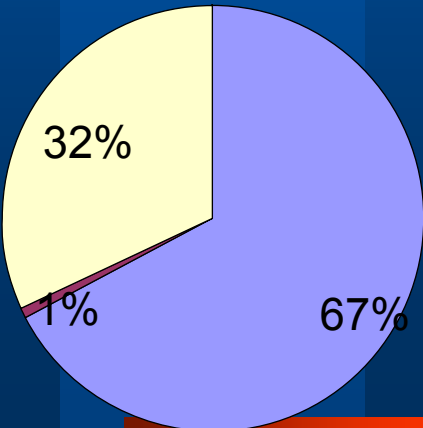
Range



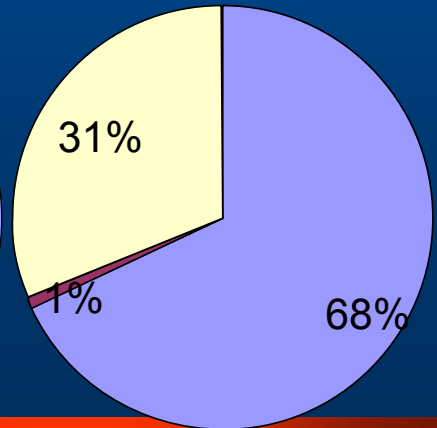
PR



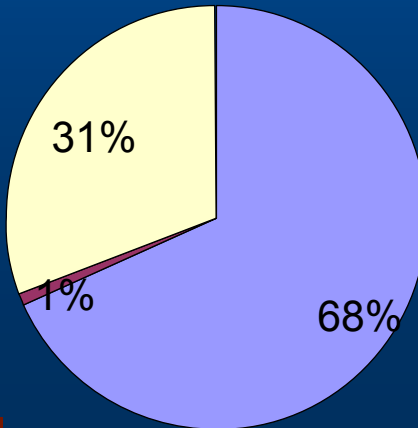
I-Test



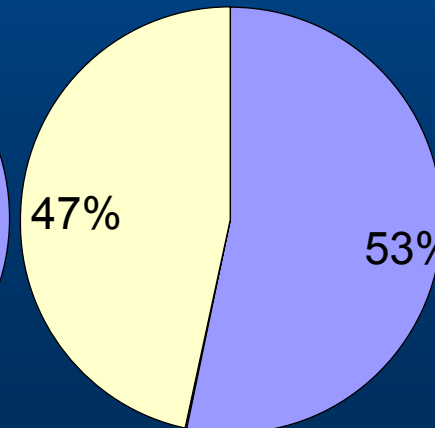
NLVI-Test



Omega

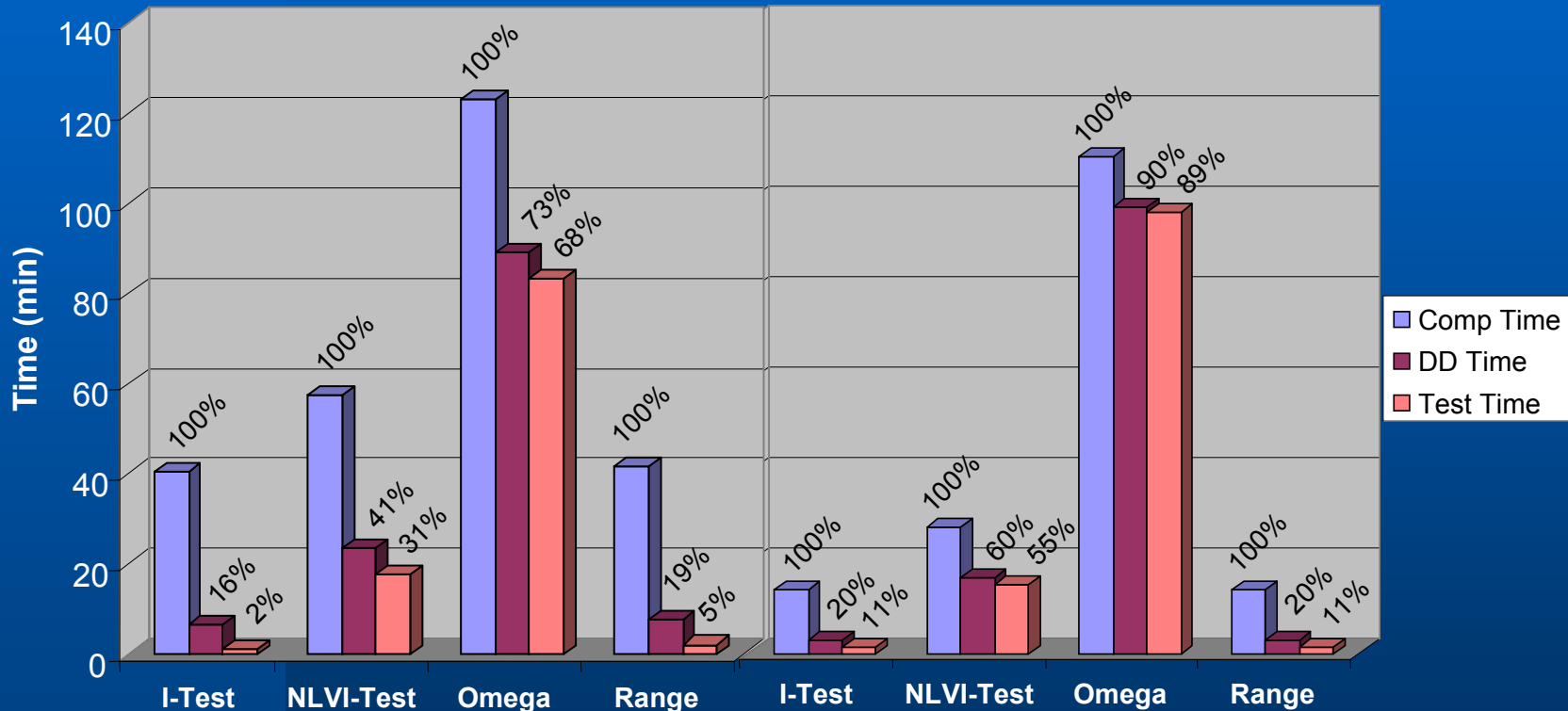


Range



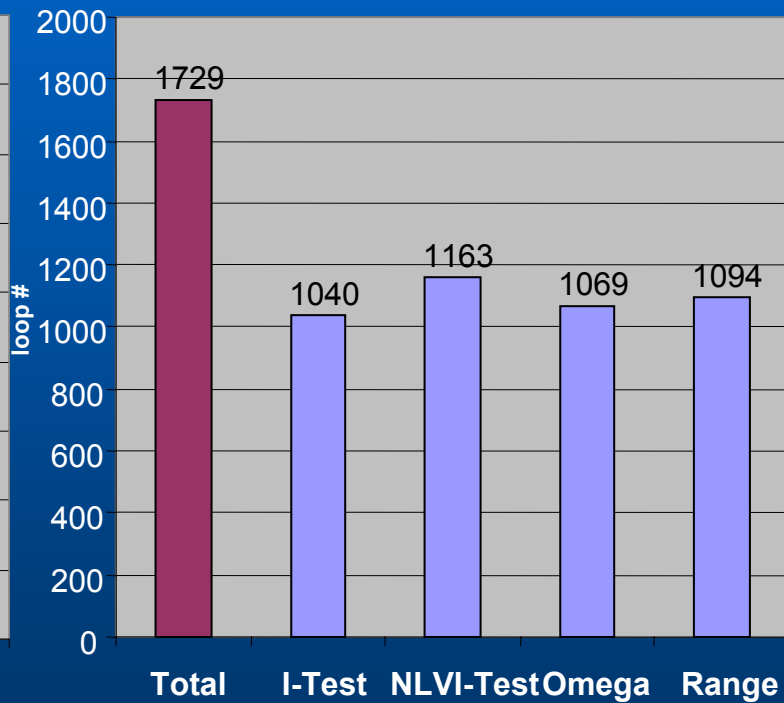
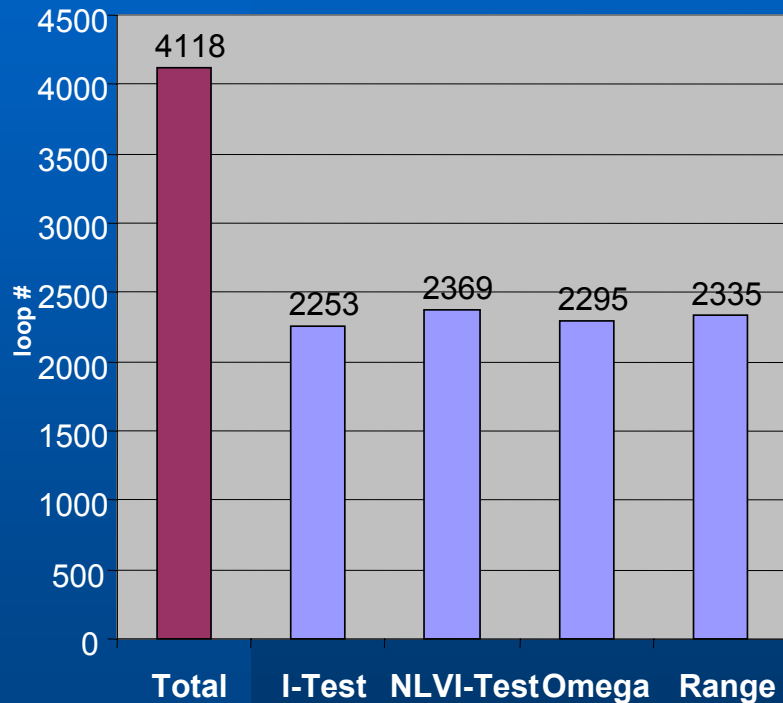
DV

Compilation Efficiency in the Perfect and SPEC Benchmarks



	Test	Comp Time	DD Time	Test Time		Test	Comp Time	DD Time	Test Time
Perfect	I-Test	40.5 (100%)	6.7 (16%)	1.0 (2%)	SPEC	I-Test	14.0 (100%)	2.8 (20%)	1.5 (11%)
	NLVI-Test	57.3 (100%)	23.5 (41%)	17.8 (31%)		NLVI-Test	28.0 (100%)	16.7 (60%)	15.4 (55%)
	Omega	122.9 (100%)	89.2 (73%)	83.4 (68%)		Omega	109.9 (100%)	98.6 (90%)	97.3 (89%)
	Range	41.6 (100%)	7.8 (19%)	2.1 (5%)		Range	14.1 (100%)	2.8 (20%)	1.5 (11%)

Loop Parallelization in the Perfect and SPEC Benchmarks



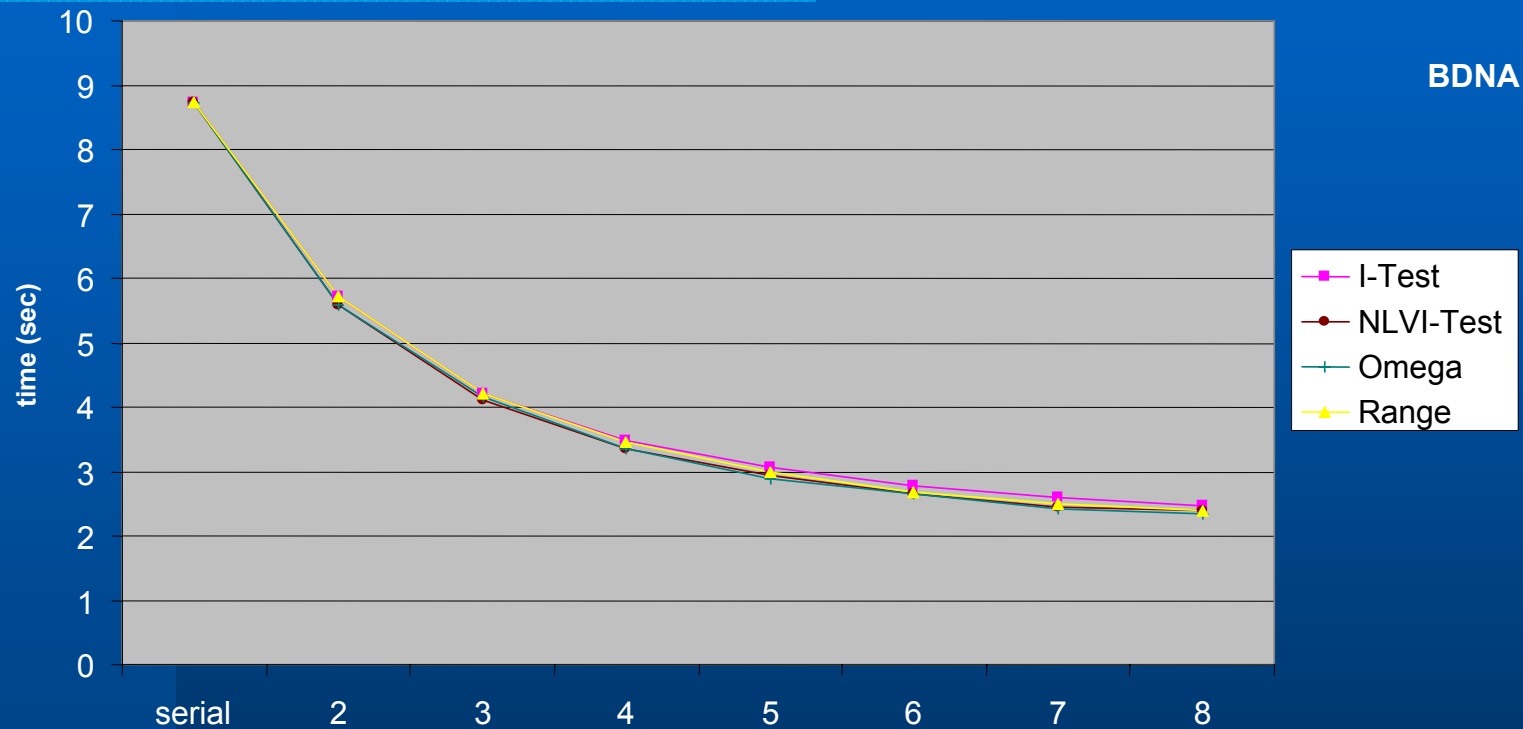
	Test	Total Loops	Parallel		Test	Total Loops	Parallel
Perfect	I-Test	4118 (100%)	2253 (55%)	SPEC	I-Test	1729 (100%)	1040 (60%)
	NLVI-Test	4118 (100%)	2369 (58%)		NLVI-Test	1729 (100%)	1163 (67%)
	Omega	4118 (100%)	2295 (56%)		Omega	1729 (100%)	1069 (62%)
	Range	4118 (100%)	2335 (57%)		Range	1729 (100%)	1094 (63%)

Execution Performance in MDG of the Perfect Benchmarks



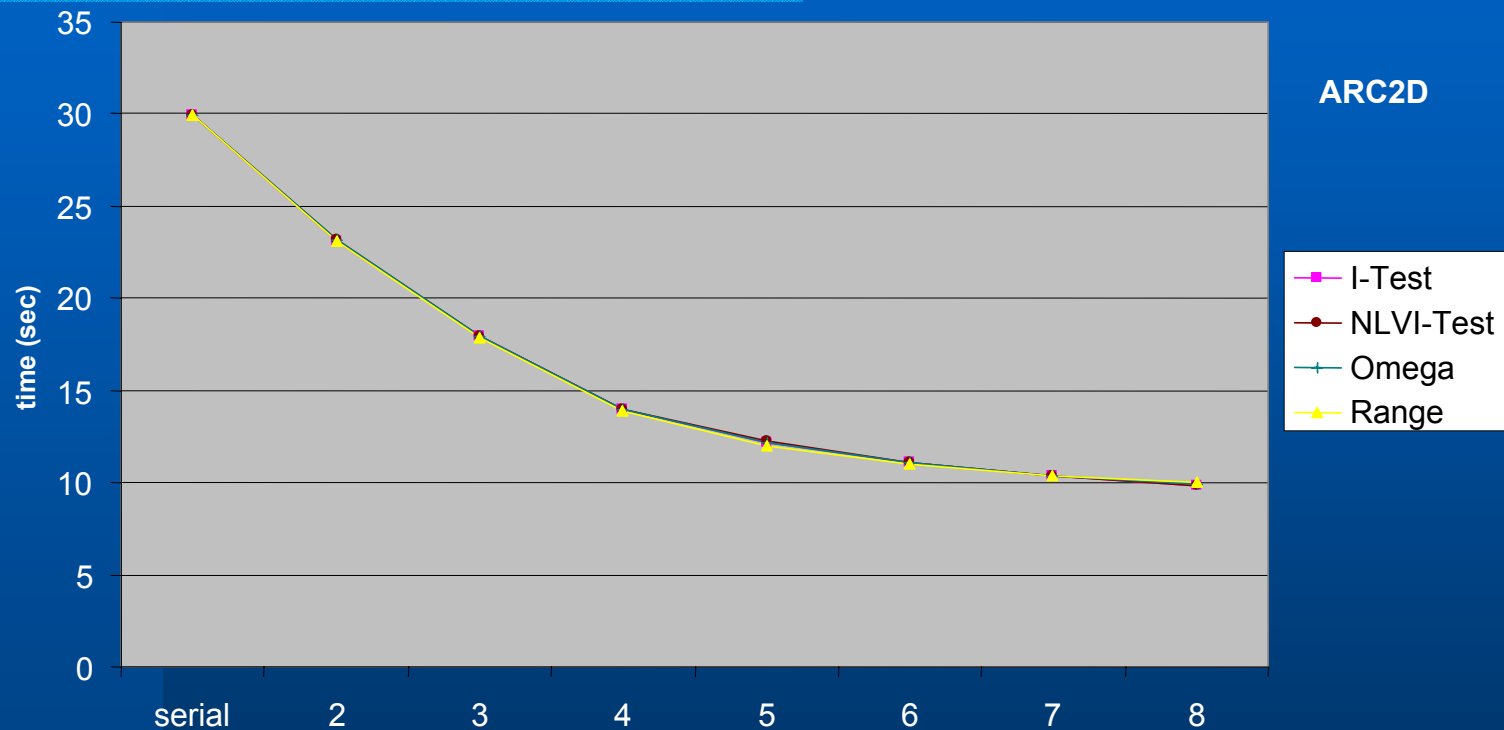
Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
MDG	I-Test	51.33	28.76	1.8	19.48	2.6	14.90	3.4	12.08	4.2	10.23	5.0	9.26	5.5	7.96	6.4
	NLVI-Test	51.33	28.97	1.8	19.63	2.6	15.01	3.4	12.16	4.2	10.31	5.0	9.35	5.5	8.00	6.4
	Omega	51.33	28.69	1.8	19.43	2.6	14.88	3.4	12.09	4.2	10.22	5.0	9.27	5.5	7.97	6.4
	Range	51.33	29.03	1.8	19.65	2.6	15.00	3.4	12.16	4.2	10.31	5.0	9.33	5.5	8.00	6.4

Execution Performance in BDNA of the Perfect Benchmarks



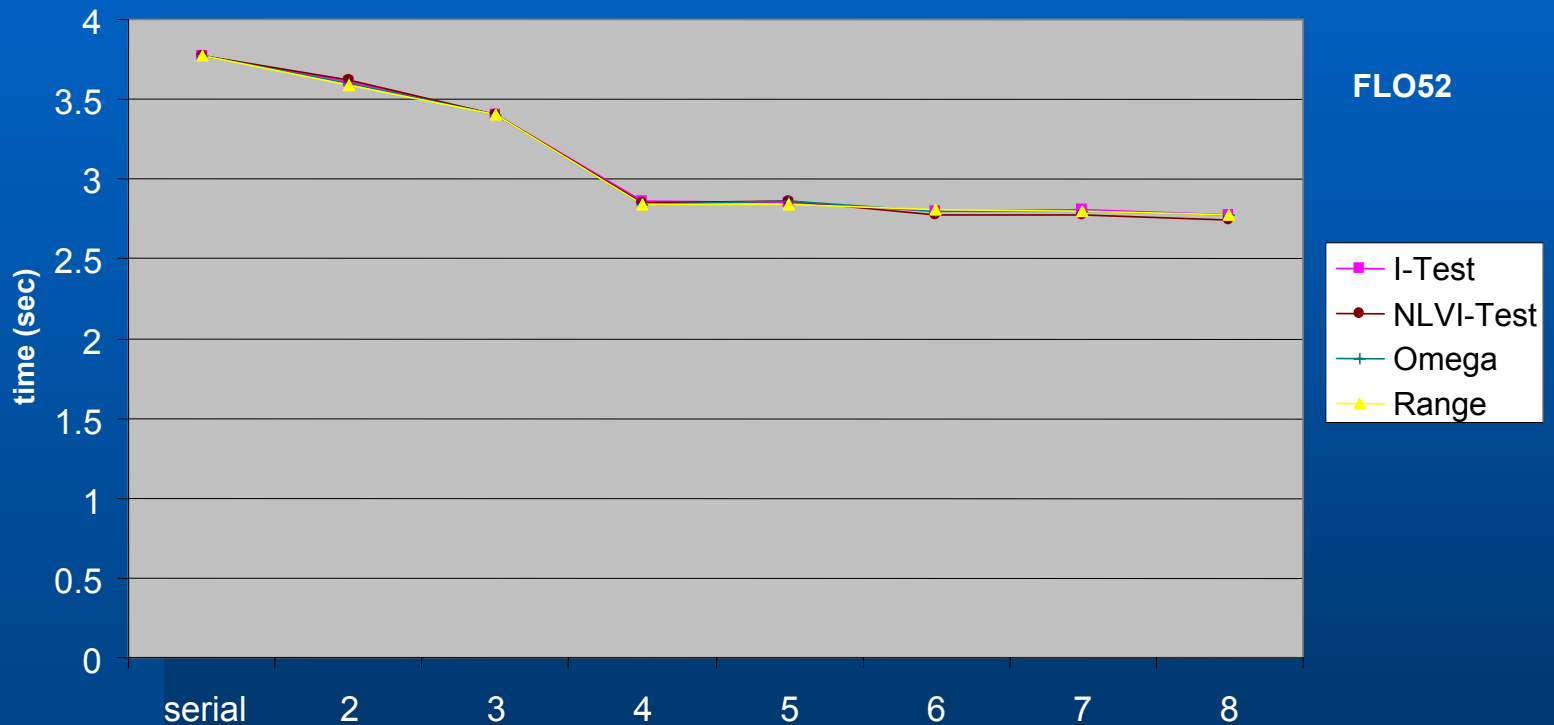
Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	8.73	5.71	1.5	4.22	2.1	3.49	2.5	3.07	2.8	2.79	3.1	2.61	3.3	2.47	3.5
	NLVI-Test	8.73	5.58	1.6	4.11	2.1	3.35	2.6	2.93	3.0	2.64	3.3	2.45	3.6	2.39	3.7
BDNA	Omega	8.73	5.59	1.6	4.15	2.1	3.34	2.6	2.89	3.0	2.66	3.3	2.42	3.6	2.35	3.7
	Range	8.73	5.72	1.5	4.21	2.1	3.46	2.5	2.98	2.9	2.68	3.3	2.49	3.5	2.38	3.7

Execution Performance in ARC2D 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	29.93	23.10	1.3	17.97	1.7	13.94	2.1	12.19	2.5	11.12	2.7	10.35	2.9	9.84	3.0
	NLVI-Test	29.93	23.15	1.3	17.93	1.7	13.96	2.1	12.30	2.4	11.08	2.7	10.40	2.9	9.86	3.0
ARC2D	Omega	29.93	23.18	1.3	17.92	1.7	13.96	2.1	12.22	2.4	11.13	2.7	10.36	2.9	9.91	3.0
	Range	29.93	23.10	1.3	17.88	1.7	13.85	2.2	12.02	2.5	10.98	2.7	10.36	2.9	10.00	3.0

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	2.81	1.3	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	Omega	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
	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	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	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



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	172.99	109.27	1.6	78.72	2.2	62.86	2.8	53.39	3.2	47.74	3.6	43.65	4.0	40.70	4.3
	NLVI-Test	172.99	104.11	1.7	74.11	2.3	58.30	3.0	49.01	3.5	43.21	4.0	39.31	4.4	36.29	4.8
HYDRO2D	Omega	172.99	109.67	1.6	78.58	2.2	62.70	2.8	53.26	3.2	47.86	3.6	43.69	4.0	40.70	4.3
	Range	172.99	104.34	1.7	74.15	2.3	58.54	3.0	48.93	3.5	43.37	4.0	39.28	4.4	36.18	4.8

Execution Performance in MGRID of the SPEC Benchmarks



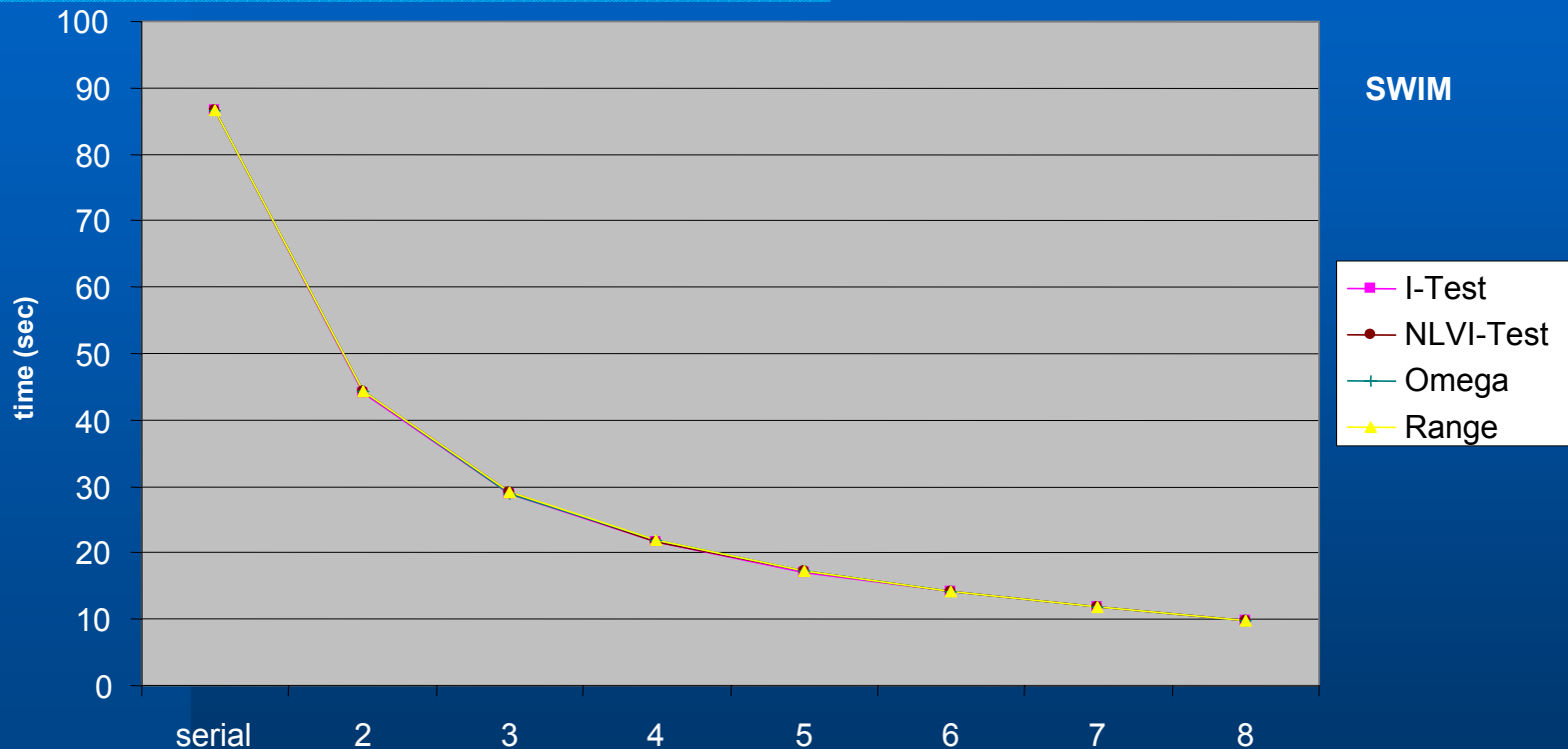
Program	Test	serial	2-proc	3-proc	4-proc	5-proc	6-proc	7-proc	8-proc
	I-Test	119.20	66.39	58.44	47.67	42.95	40.26	38.78	37.28
	NLVI-Test	119.20	62.84	54.07	43.05	37.83	34.95	32.63	30.82
MGRID	Omega	119.20	66.61	58.34	47.64	42.90	40.35	38.68	37.43
	Range	119.20	65.89	57.80	47.08	42.61	39.99	38.79	37.15

Execution Performance in SU2COR of the SPEC Benchmarks



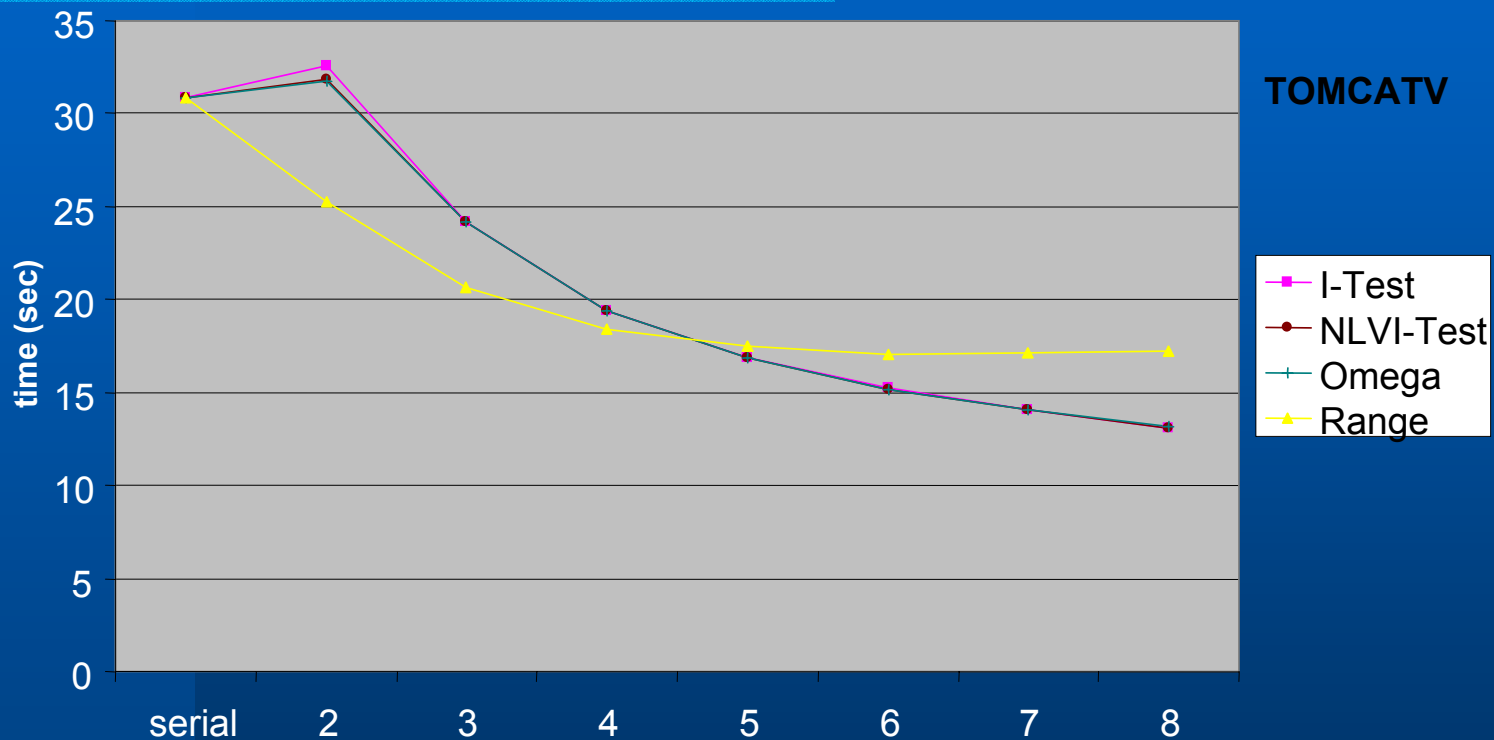
Program	Test	serial	2-proc	sp	3-proc	sp	4-proc	sp	5-proc	sp	6-proc	sp	7-proc	sp	8-proc	sp
	I-Test	119.33	69.51	1.7	51.54	2.3	41.85	2.9	37.82	3.2	35.07	3.4	34.34	3.5	32.86	3.6
	NLVI-Test	119.33	69.79	1.7	51.35	2.3	41.80	2.9	37.87	3.2	35.14	3.4	34.19	3.5	33.05	3.6
SU2COR	Omega	119.33	69.72	1.7	51.62	2.3	41.83	2.9	37.76	3.2	35.07	3.4	34.17	3.5	32.94	3.6
	Range	119.33	70.15	1.7	51.60	2.3	42.21	2.8	38.14	3.1	35.43	3.4	34.27	3.5	33.35	3.6

Execution Performance in SWIM of the SPEC 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	86.69	44.12	28.98	21.72	17.11	14.18	11.86	9.84
	NLVI-Test	86.69	44.44	29.13	21.74	17.16	14.27	11.88	9.88
SWIM	Omega	86.69	44.24	28.84	21.88	17.15	14.22	11.92	9.81
	Range	86.69	44.39	29.17	21.78	17.16	14.21	11.82	9.89

Execution Performance in TOMCATV of the SPEC 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	30.86	32.59	0.9	24.22	1.3	19.43	1.6	16.89	1.8	15.22	2.0	14.06	2.2	13.10	2.4
	NLVI-Test	30.86	31.85	1.0	24.14	1.3	19.40	1.6	16.87	1.8	15.19	2.0	14.06	2.2	13.11	2.4
TOMCATV	Omega	30.86	31.77	1.0	24.19	1.3	19.42	1.6	16.87	1.8	15.18	2.0	14.04	2.2	13.13	2.4
	Range	30.86	25.29	1.2	20.69	1.5	18.36	1.7	17.49	1.8	17.06	1.8	17.17	1.8	17.25	1.8

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