Benchmark Characterization for Experimental System Evaluation

Thomas M. Conte

Wen-mei W. Hwu

Center for Reliable and High-Performance Computing
University of Illinois
conte@csg.uiuc.edu

Workload Characterization: Motivation, Goals and Methodology

Lizy Kurian John, Purnima Vasudevan and Jyotsna Sabarinathan*
Electrical and Computer Engineering Department
The University of Texas at Austin
{ljohn,purnima,sabarina}@ece.utexas.edu

Why Workload/Benchmark Characterization?

1. Interpret simulation results effectively
2. Design machines to match workload features
3. Validate representativeness of sampled traces
4. Benchmark Subsetting
5. Synthetic Benchmark Validation
6. Abstract program behavior model which in conjunction with a system model can be used for quick performance evaluation of systems.

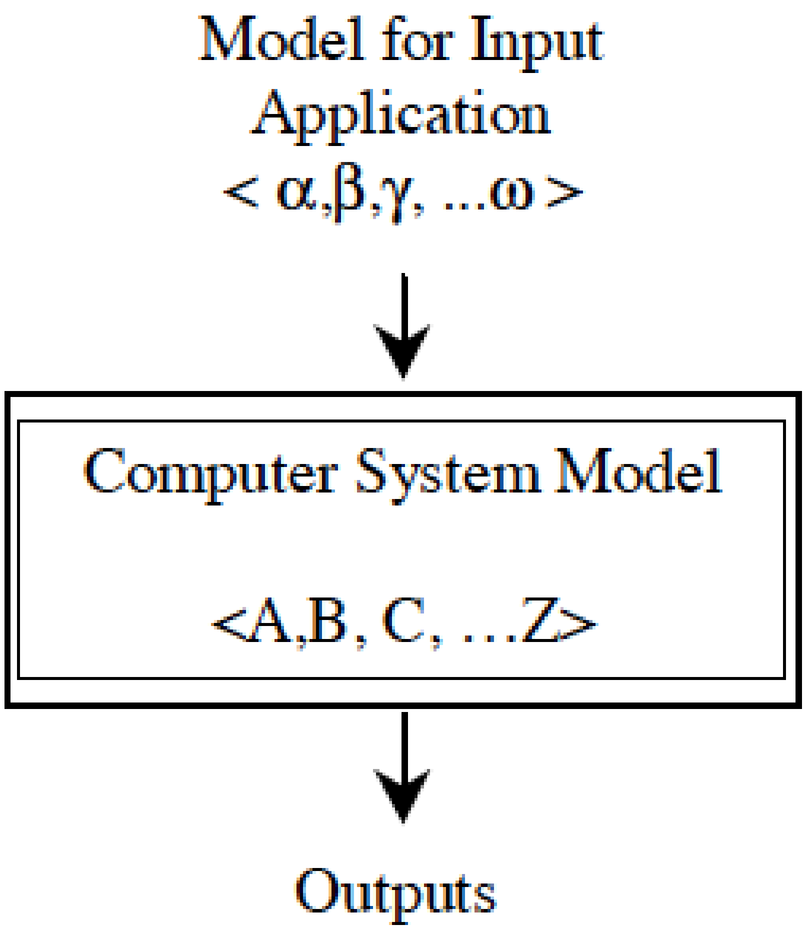


Figure 1. System Model

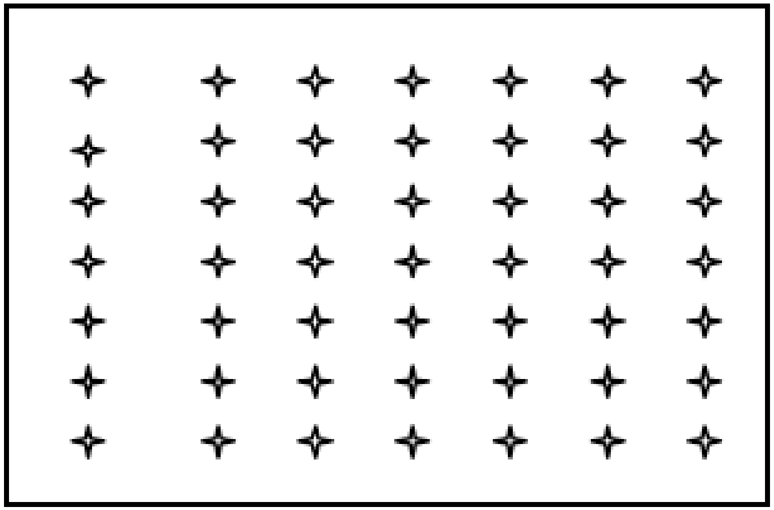


Figure 3. Potential application span

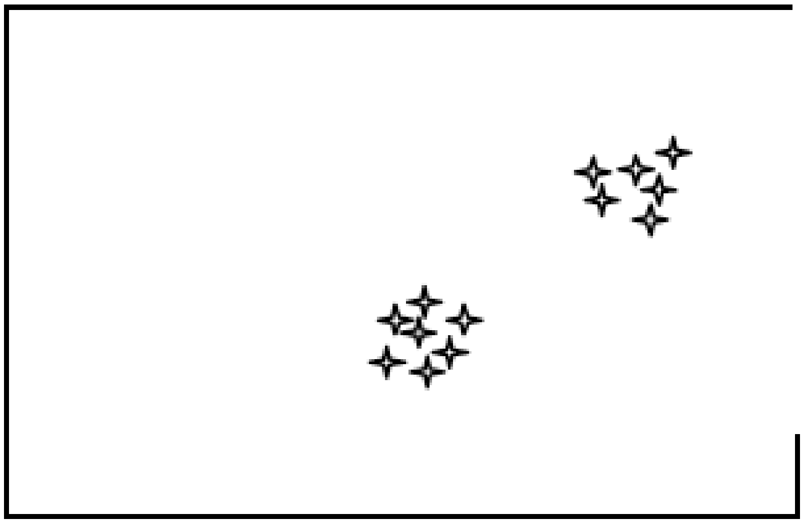


Figure 2 Potential span of existing benchmark suites in the potential workload space

If we need to find good coverage of the workload space, we need to understand the performance domain

A trace

Is a time sequence w(t) = ri,

Where ri an element of set R

Eg: R {2000,2004,2008,2012}

w(t) = 2000,2004,2008,2012,2004,2008,2012

Ri’s can be addresses of instructions or data

DEFINITION 2.1: Define next$(w(t)) = k$, if k is the smallest integer such that $w(t) = w(t+k)$. ∎

2000,2004,2008,2012,2000,2004,2008,2012,

next(2000) = 4

DEFINITION 2.2: The number of unique references between $w(t)$ and next$(w(t))$, is defined as, $u(w(t)) = \|\{ w(t+k) \mid i \leq k < \text{next}(w(t)) \}\|$. ∎

u(2000)=3

DEFINITION 2.3: Define $f^T(x)$, the *interreference temporal density function*, $f^T(x)$,, to be the probability of there being x unique references between successive references to the same item,

$$f^T(x) = \sum_t P\Big[u(w(t)) = x\Big].$$

2000,2004,2008,2012,(2004,2008,2012^9),

ft(inf)=1/31; ft(1)=0 ft(2)=30/31

2000,2000,2000,2000,2000

ft(0)=1

DEFINITION 2.3: Define $f^T(x)$, the *interreference temporal density function*, $f^T(x)$,, to be the probability of there being x unique references between successive references to the same item,

$$f^T(x) = \sum_t P\Big[u(w(t)) = x\Big].$$

■

3000,3002,3004,3000,3004,3000,3002

u(3000-1)=2

u(3002)=2

u(3004)=1

u(3000-2)=1

ft(0)=0; ft(1)=0.5; ft(2)=0.5

Why temporal density function?

1. Hit rate of LRU managed buffers
2. Mattson's stack distance
3. Abstract cache model

The performance of buffers managed under stacking replacement policies (e.g., LRU) depends directly on this measure of temporal locality. The hit ratio for a fully associative buffer of size N is $h(N) = \sum_{y \leq N} f^T(y)$ (see [17]).

Definition 2.4: The *interreference spatial density function*, $f^S(x)$, is defined as,

$$f^S(x) = \sum_t \sum_{k=1}^{\mathrm{next}(w(t))} P\Big[\big| w(t) - w(t+k) \big| = x \Big].$$

∎

(2004,2008,2012)^10

fs(4)=0.5

fs(8) =0.5

2000,2000,2000,2000,2000

fs(0)=1

DEFINITION 2.4: The *interreference spatial density function*, $f^S(x)$, is defined as,

$$f^S(x) = \sum_t \sum_{k=1}^{\mathrm{next}(w(t))} P\Big[\big|w(t) - w(t+k)\big| = x\Big].$$

∎

3000,3004,3008,300C,3004,3000

3000-3000 -> 4,8,12,4
3004-3004 -> 4,8

fs(4)=3/6=0.5;   fs(8) = 2/6 = 0.3333
fs(12)=1/6 = 0.167

Conte paper

```
Calc_loc_measures(r_i):
 begin
     if  not first time r_i encountered then
     begin
         d ← depth(r_i)
         remove r_i from the stack
         for  all r_j with depth(r_j) < d
         begin
             dist ← |α(r_j) − α(r_i)|
             f̂^S(dist) ← f̂^S(dist) + 1
         end
         f̂^T(d) ← f̂^T(d) + 1
     end
     push(r_i)
 end
```

Figure 1: The algorithm for calculating the locality distributions.

Mattson's stack distance [1970 paper from IBM]

For LRU stack, C_t is the position of X_t in the stack S_{t-1}, so that $x_t = S_{t-1}(C_t)$

- This position is called *stack distance* Δ_t: $\Delta_t = C_t$

Time	1	2	3	4	5	6	7	8	9	10
Trace	a	b	b	c	b	a	d	c	a	a
Δ_t	∞	∞	1	∞	2	3	∞	4	3	1
LRU stack	a	b	b	c	b	a	d	c	a	a
		a	a	b	c	b	a	d	c	c
				a	a	c	b	a	d	d
							c	b	b	b

- Distance counters

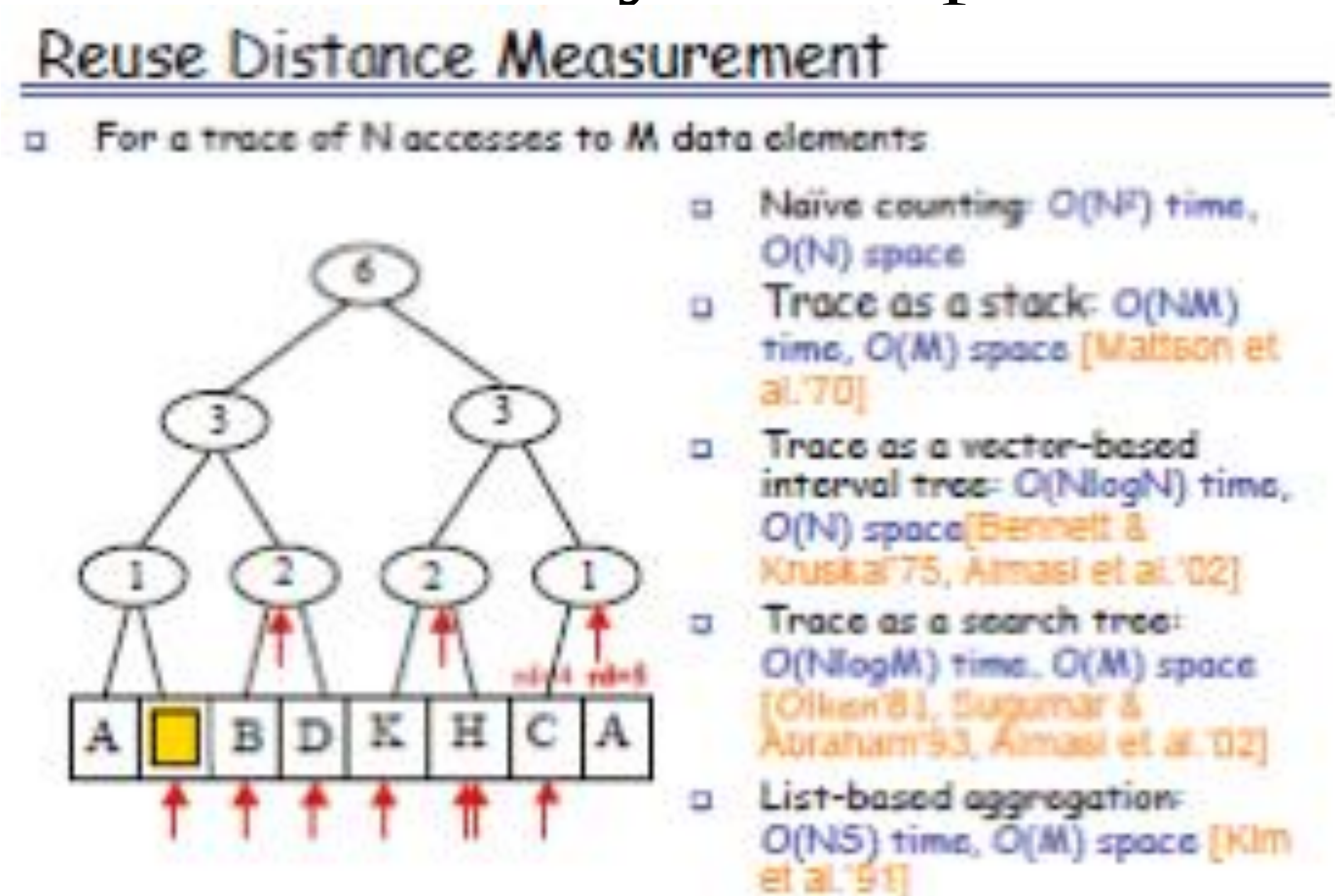
1	2	3	4	∞
2	1	2	1	4

[3] Mattson, R.L.; Gecsei, J.; Slutz, D.R.; Traiger, IL., "Evaluation techniques for storage hierarchies," *IBM Systems Journal* , vol.9, no.2, pp.78,117, 1970

Stack Distance is also called Data Reuse distance

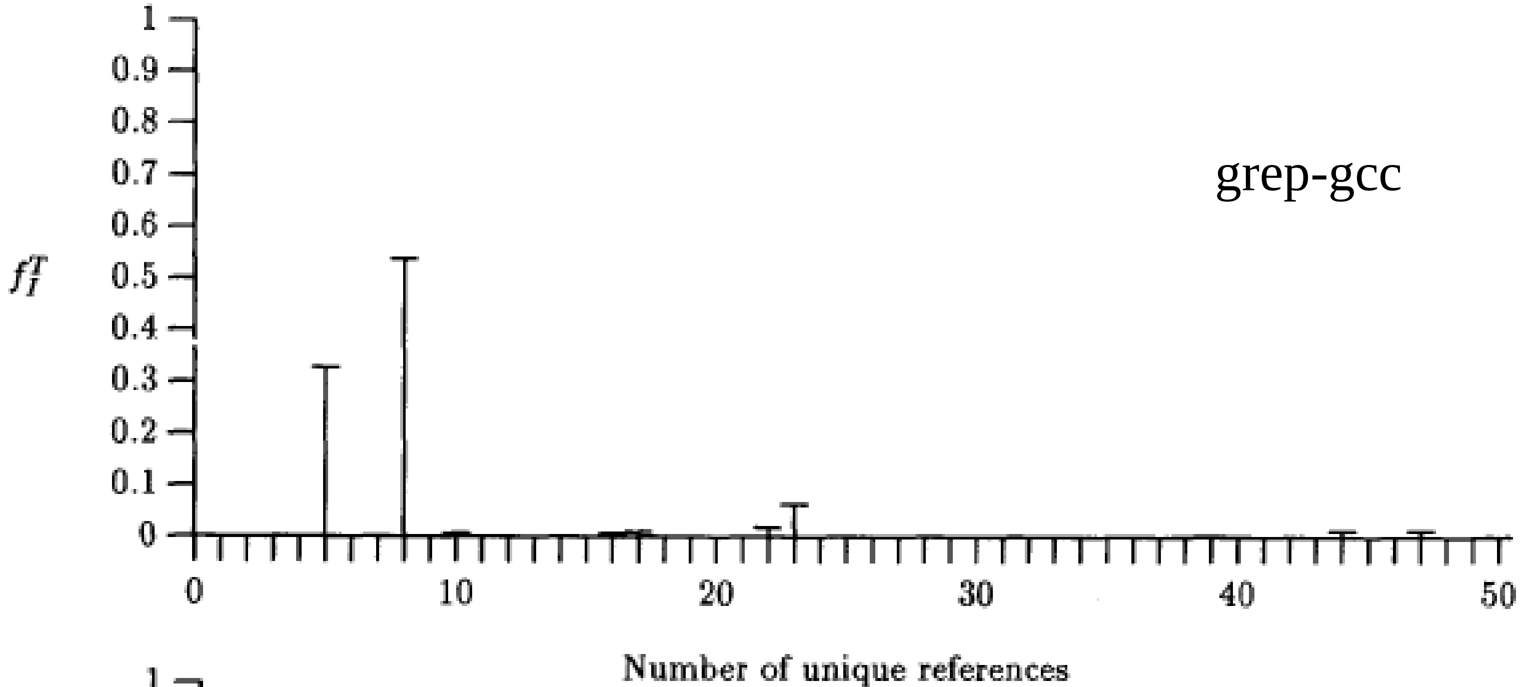
Chen Ding and Yutao Zhong. 2003. Predicting whole-program locality through reuse distance analysis. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation* (PLDI '03).

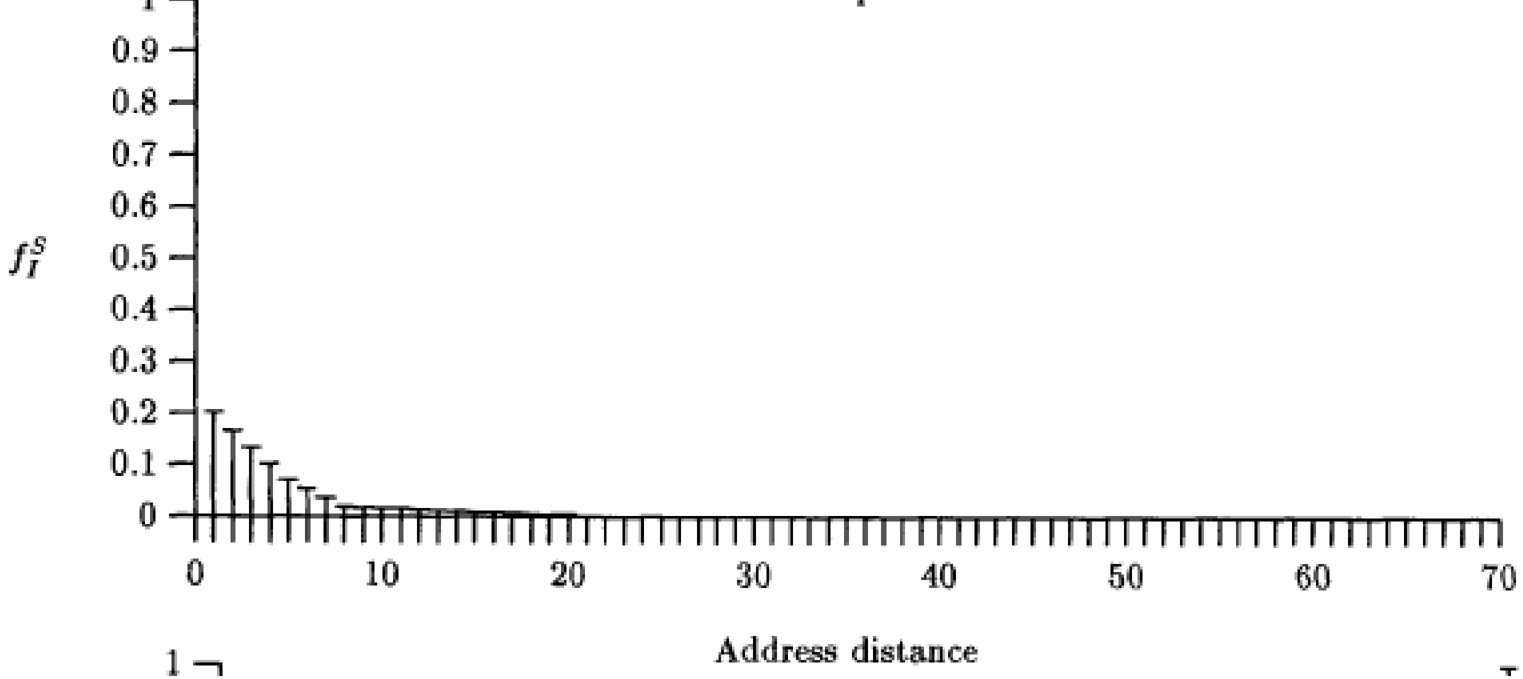
Complexity varies depending on how you keep data

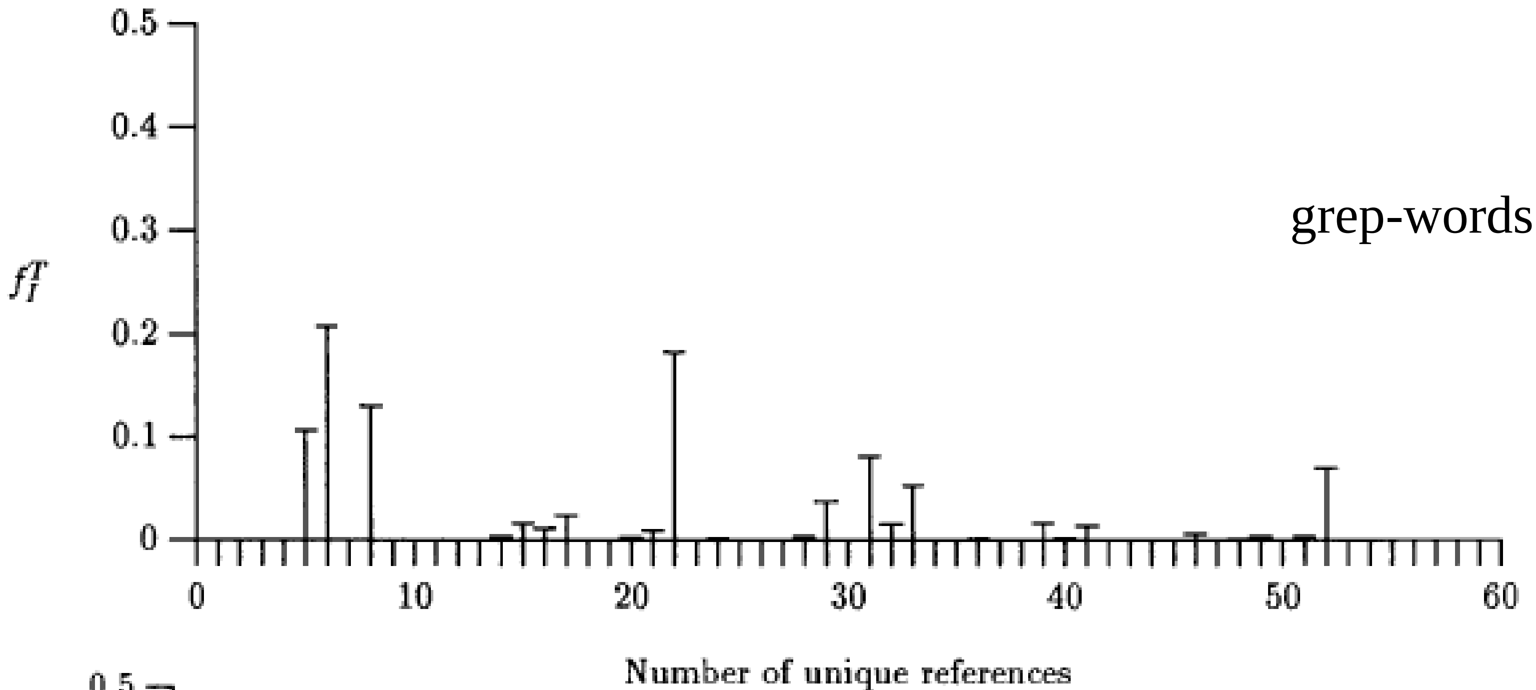


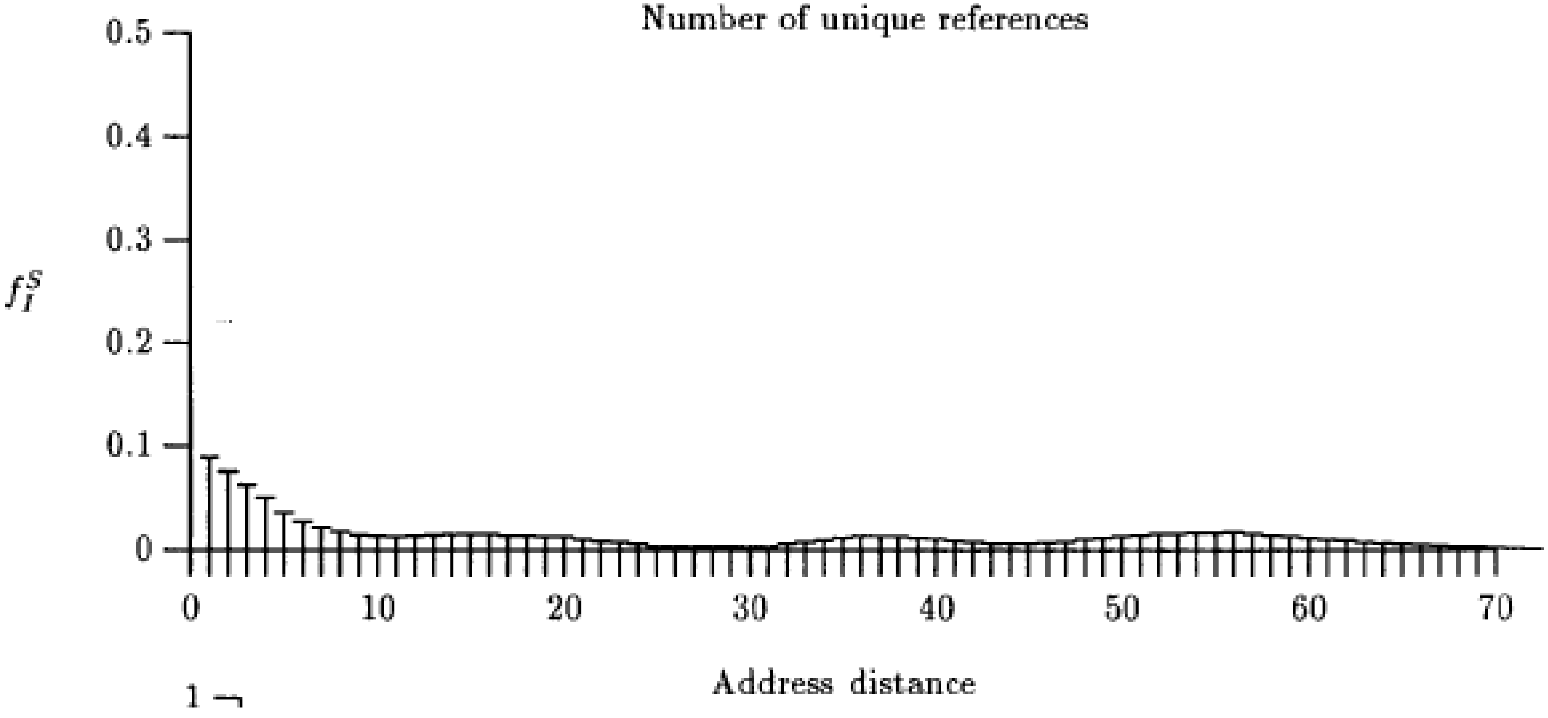
Stack method takes O(NM) time.

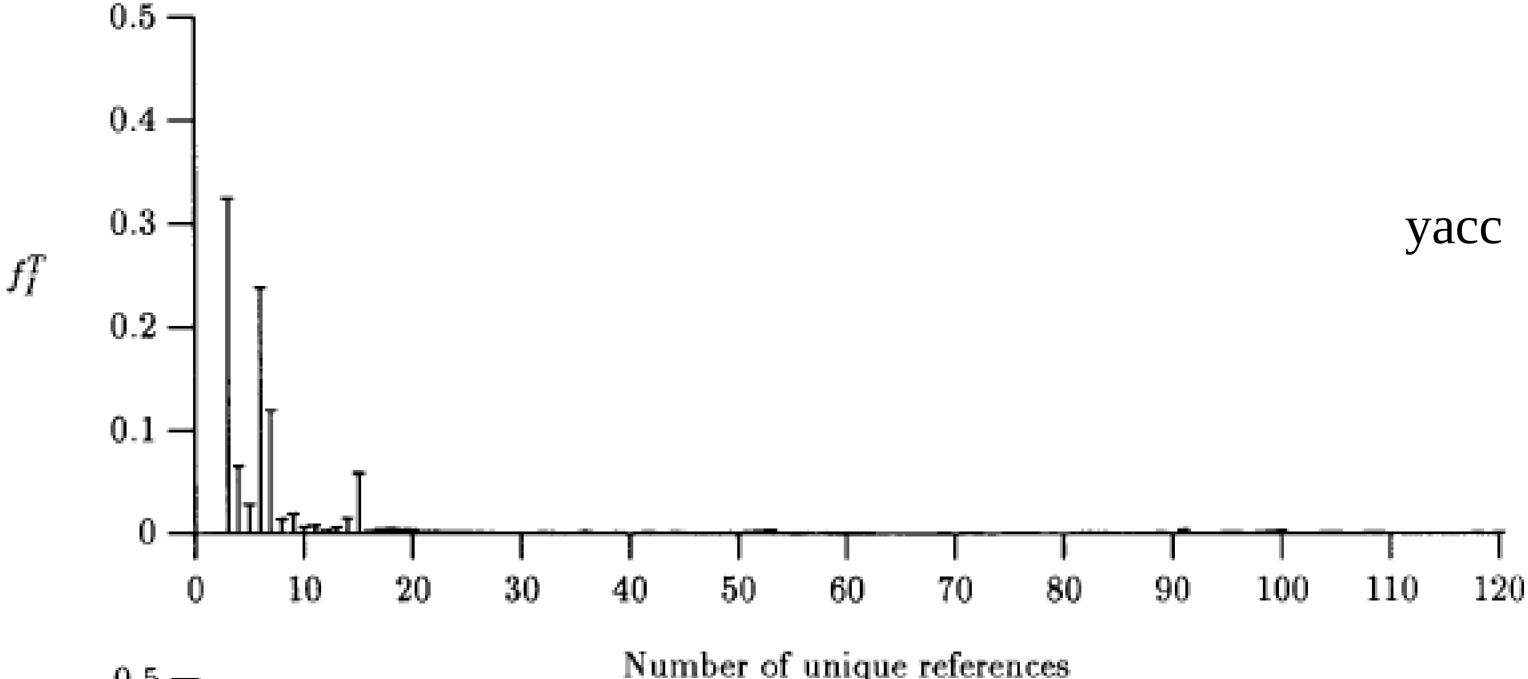
Storing data as a tree reducees complexity from O(NM) to O(NlogM)

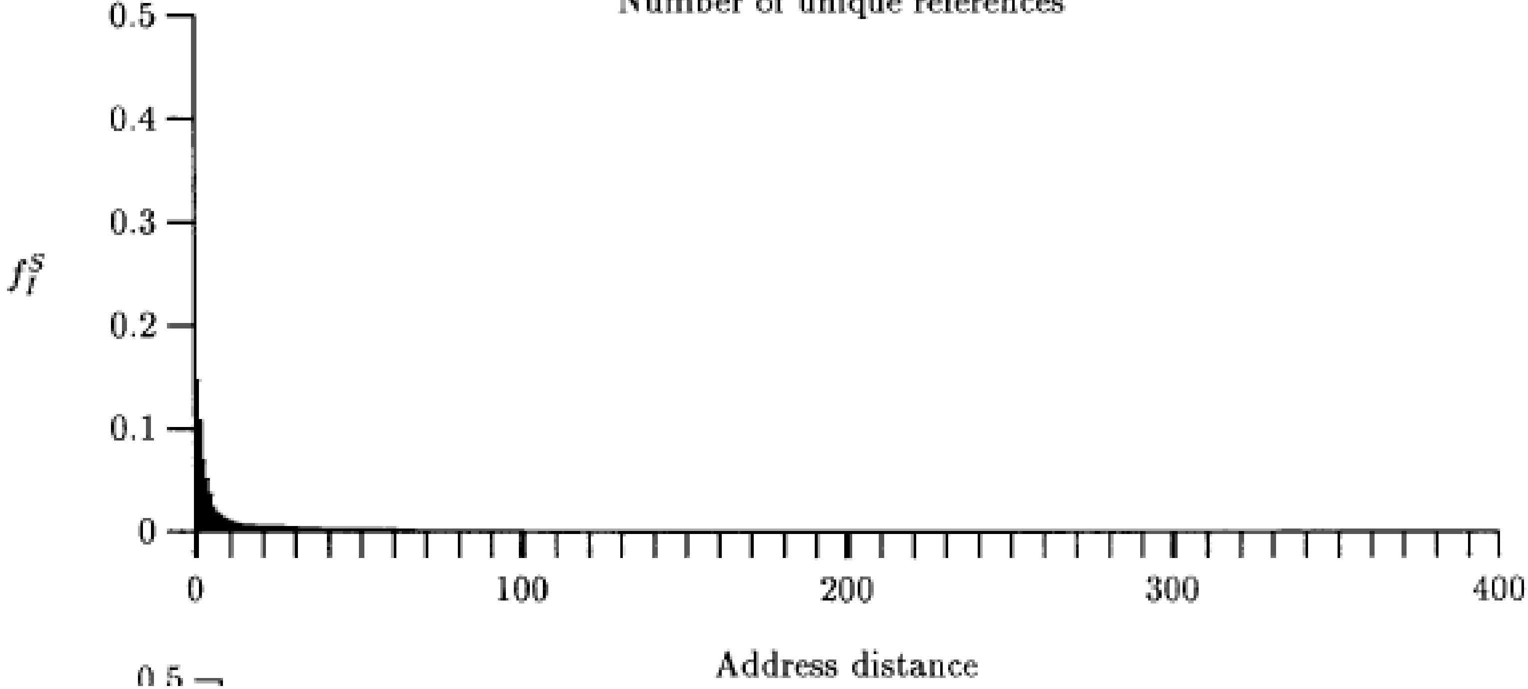

grep-gcc
f_I^T
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
Number of unique references


f_I^S
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
Address distance


grep-words
f_I^T
0.5
0.4
0.3
0.2
0.1
0
0 10 20 30 40 50 60
Number of unique references


f_I^S
0.5
0.4
0.3
0.2
0.1
0
0 10 20 30 40 50 60 70
Address distance

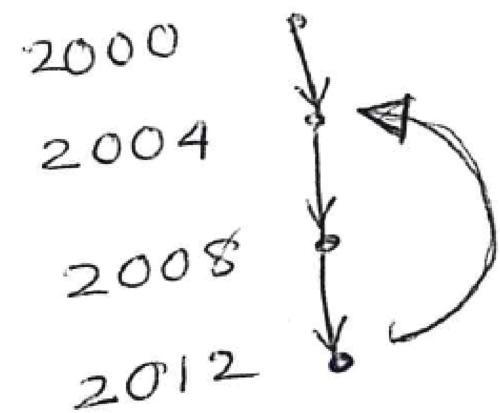

yacc
f_l^T
0.5
0.4
0.3
0.2
0.1
0
0 10 20 30 40 50 60 70 80 90 100 110 120
Number of unique references


f_l^S
0.5
0.4
0.3
0.2
0.1
0
0 100 200 300 400
Address distance

DEFINITION 2.5: The (directed) reference graph, $G = (V, E)$, of a reference stream is defined as $V = R$ and,

$$E = \{ (r_i, r_j) \mid w(t) = r_i \text{ and } w(t+1) = r_j \}.$$

2000,(2004,2008,2012)^10



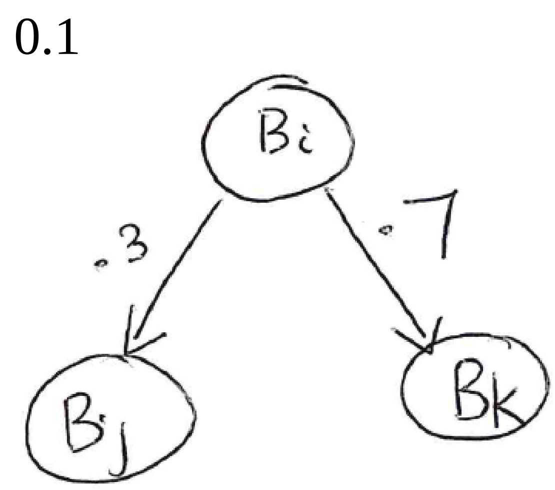
V={2000,2004,2008,2012}

E={(2000,2004), (2004,2008), (2008,2012), (2012,2004)}

Definition 2.6: Let $n_i(r_i)$ be the number of occurrences $w(t) = r_i$, for $0 \leq t \leq T$. Furthermore, let $n_{ij}(r_i, r_j)$ be the number of occurrences of $w(t+1) = r_j$, if $w(t) = r_i$. Then, the weighted reference graph, $G' = (V, E)$, is defined such that each node, $r_i \in V$, is weighted with $P[r_i] = n_i/T$, and each edge, $(r_i, r_j) \in E$ is weighted with $P[r_j|r_i] = n_{ij}/n_i$. ∎

Weighted Reference Graph

10% probability to be in Bi
Once you are in Bi, 70% probability
To go to Bk.



Based on graph definitions, groups of items referenced together in graph can be defined

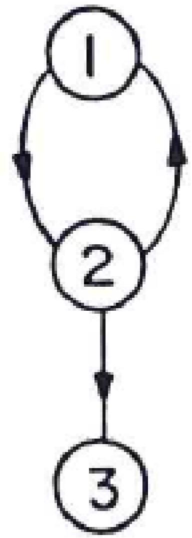
The strongly connected components of the graph are called phases.

Definition 2.7: The set of phases for a reference stream is defined as $\Phi = \{\phi_1, \phi_2, \ldots \phi_i \ldots \phi_p\}$, where

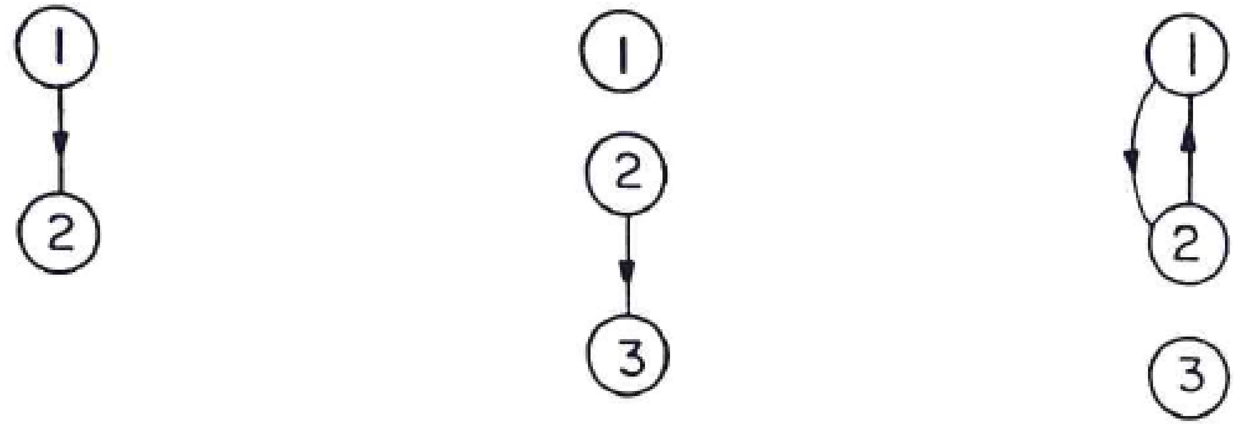
$$\phi_i = \{\, r_i \;\; | \;\; \{(r_i, r_{i+1}), (r_{i+1}, r_{i+2}), \ldots, (r_{k-1}, r_k), (r_k, r_i)\} \subseteq E \,\},$$

and, $\phi_1 \cap \phi_2 \cap \cdots \cap \phi_p = \emptyset$.

■

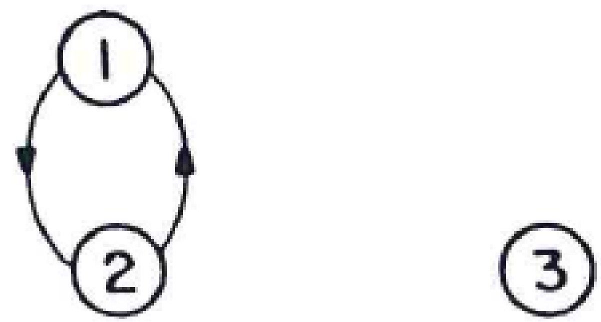


G_3



(ii) (iii) (iv)

(b) Some of the subgraphs of G_3



Strongly connected components of G_3.

In a phase, any node can be reached from any other node through a sequence of edge traversals.

During execution, the items in a newly encountered phase are guaranteed to not have been referenced before.

Intrinisic cold start buffer behavior can be predicted using phase transitions

Interphase density function, a new metric can be defined for capturing phase behavior

DEFINITION 2.8: The interphase density function, $f^{\phi}(x)$, is the probability that a phase of size x is encountered in the reference stream,

$$f^{\phi}(x) = \sum_{\|\phi\|=x} \sum_{r_i \in \phi} P[r_i], \quad \text{for all } \phi \in \Phi.$$

CONTROL FLOW BEHAVIOR

Basic blocks

Basic Block Weighted Reference Graph

$G_{bb} = (V_{bb}, E_{bb})$

When the program is mapped into linear memory space of a computer, the graph nature of the program is preserved using branch instructions.

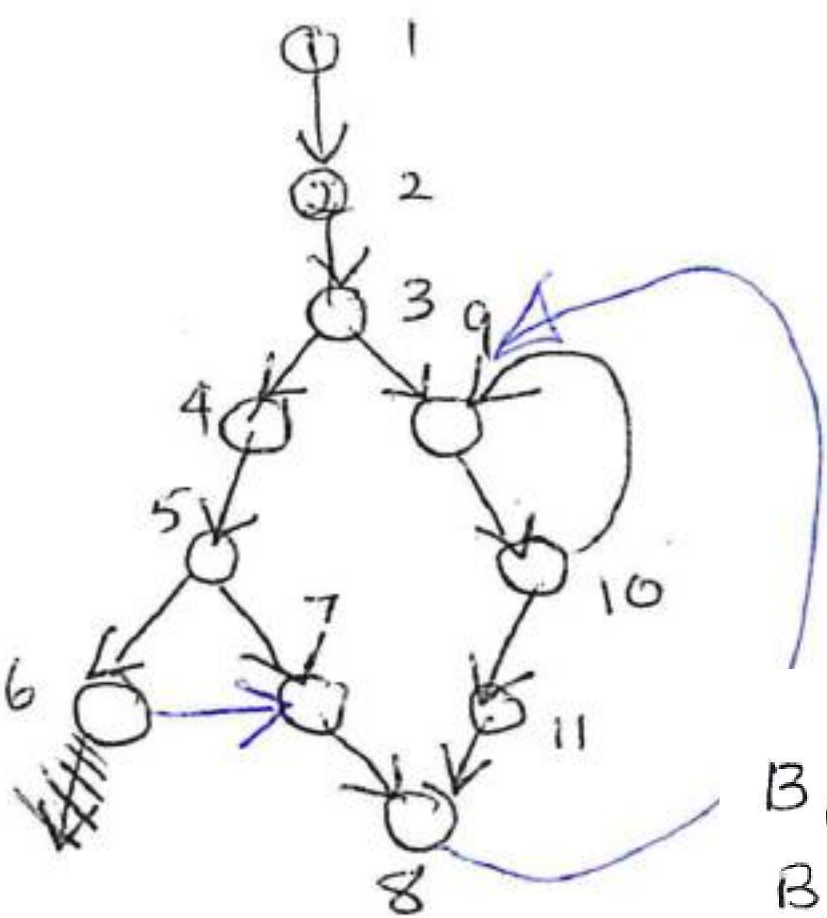
CONTROL FLOW BEHAVIOR

Basic blocks

Basic Block Weighted Reference Graph

$G_{bb} = (V_{bb}, E_{bb})$

When the program is mapped into linear memory space of a computer, the graph nature of the program is preserved using branch instructions.



$B_1 = \{1, 2, 3\}$.

$B_2 = \{4, 5\}$

$B_3 = \{6\}$

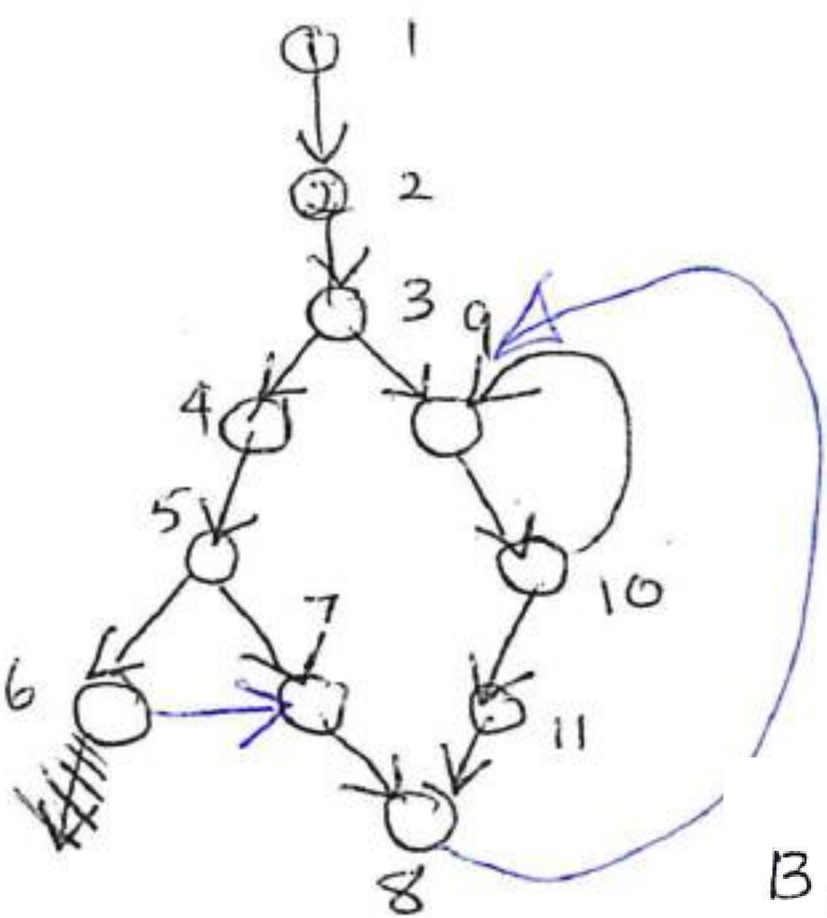
$B_4 = \{7\}$

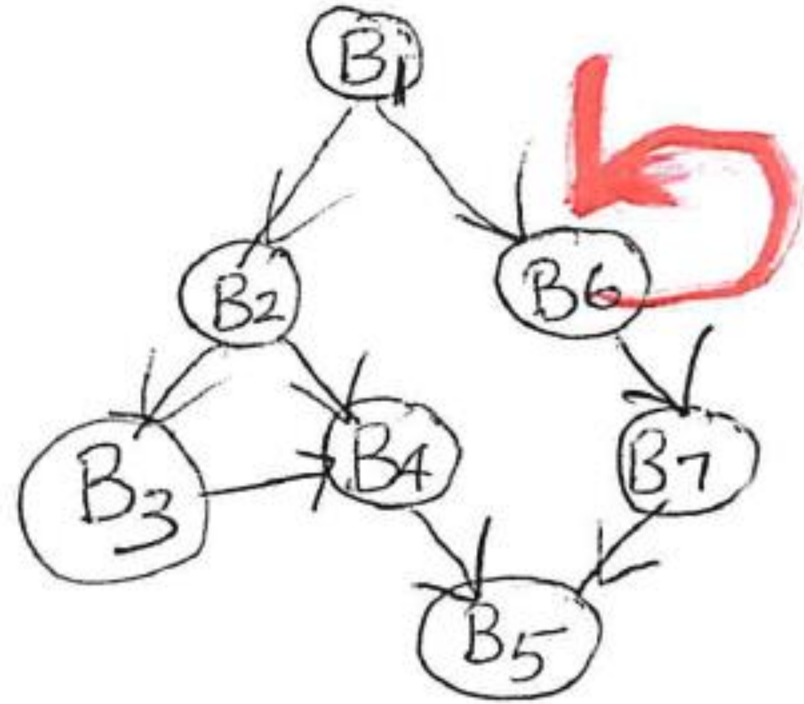
$B_5 = \{8\}$

$B_6 = \{9, 10\}$

$B_7 = \{11\}$

F





$B_1 = \{1, 2, 3\}$.

$B_2 = \{4, 5\}$

$B_3 = \{6\}$

$B_4 = \{7\}$

$B_5 = \{8\}$

$B_6 = \{9, 10\}$

$B_7 = \{11\}$

DEFINITION 2.9: The prediction probability of B_i, $P_p(B_i)$ is defined as,

$$P_p(B_i) = \max\{\, P[\, B_j | B_i \,] \mid (B_i, B_j) \in E_{BB} \,\}.$$

■

DEFINITION 2.10: The branch prediction accuracy, A, is defined as,

$$A = \sum_{i=1}^{N} P(B_i) P_p(B_i).$$

■

P(Bi) is the prob of occurrrence of the branch Bi

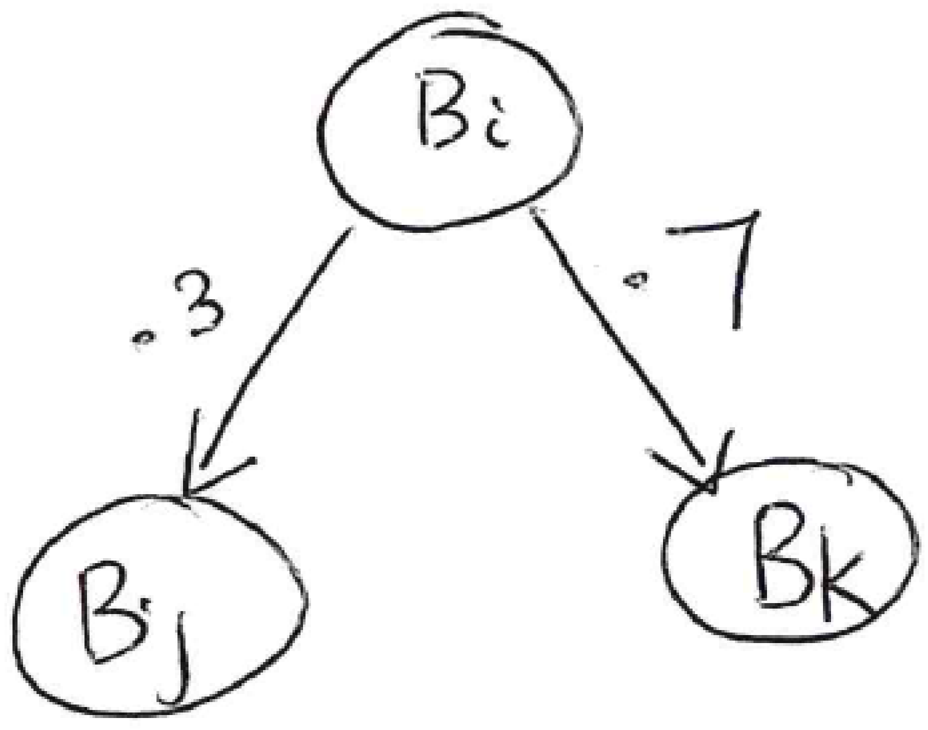

Bi
.3
.7
Bj
Bk

Table 1: Control flow GRIPs

GRIP	Benchmark characteristic measured
A	Predictability of branches
F_{CB}	Fraction of conditional branches
$f_I^T(x)$	Instruction stream temporal locality
$f_I^S(x)$	Instruction stream spatial locality
$f_I^{\phi}(x)$	Instruction stream phase behavior

Another metric

$\bar{L}_{BB}$ = average length of B.B.

DATA FLOW GRIPs

Important features of data items/variables

Lifetime of variables

Locality of Variables

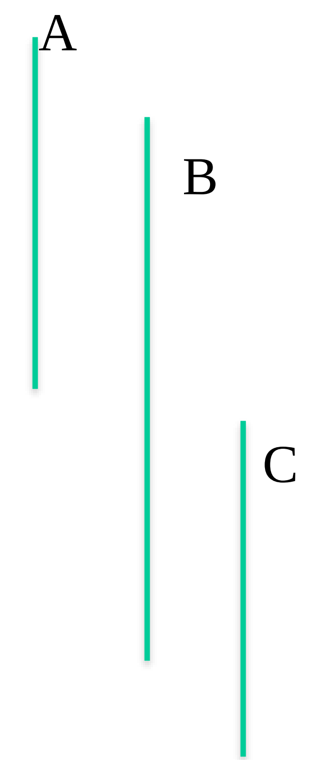
Data dependence between variables

Life Cycle of Variables - Variables go through a life cycle in which they are created, used, and then discarded.

Register allocation is performed using the technique of graph coloring

A register is assigned to two different variables if the two variables are not live (active) at the same time.

No of variables estimated by variable life density function



DEFINITION 2.11: Define the variable life density function, $f^{VL}(n_V)$, as the probability that n_V variables are live at any time during execution of the benchmark program.

■

If there are 'm' registers available, then register utilization will be $\sum_{i \leq m} f^{VL}(i)$

Amount of spill code $= \sum_{i > m} f^{VL}(i)$

Table 2: Data flow GRIPs

GRIP	Benchmark characteristic measured
$f^{VL}(n_v)$	Live variables/register use
$f_D^T(x)$	Data stream temporal locality
$f_D^S(x)$	Data stream spatial locality
$f_D^\phi(x)$	Data stream phase behavior
$p_{i,j}^{DD}$	Data dependence schedulability

Data Dependence Behavior

DEFINITION 2.12: If $\mathcal{R}(i_j)$ is the set of variables read by instruction $w(t_1) = i_j$, and $\mathcal{W}(i_k)$ is the set of variables written by instruction $w(t_2) = i_k$, for $i_j, i_k \in I$, and $t_1 < t_2$, then, the instruction dependence graph is a graph, $G_{ID} = (V_I, E_{ID})$, such that $V_I = I$ and

$$E_{ID} = \{ (i_k, i_j) \mid \mathcal{W}(i_j) \cap \mathcal{R}(i_k) \neq \emptyset \}$$

t1 > t2

Dynamic Scheduling (OOO) by Tomasulo Alg is dictated by the data dependence graph