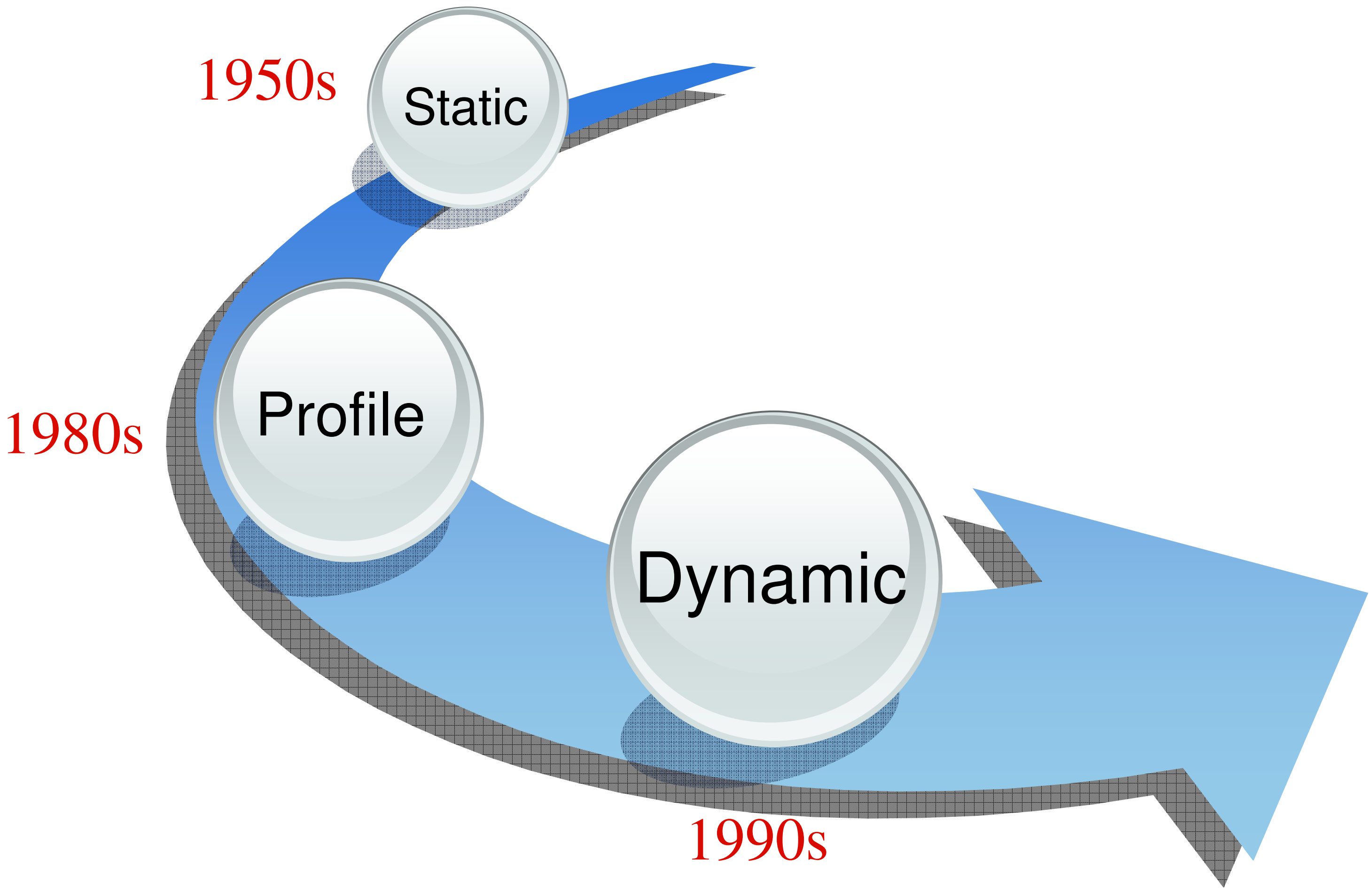
Exploiting Statistical Correlations for Proactive Prediction of Program Behaviors

Yunlian Jiang, Eddy Zhang, Kai Tian,
Feng Mao, Malcom Gethers, Xipeng Shen
CAPS Research Group, College of William and Mary, USA

Yaoqing Gao
IBM Toronto Lab, Canada

Program Optimizations



Prerequisite for Optimizations

Accurate prediction of <u>how programs would behave</u>.

Program Behaviors

↓

(procedure calling freq,
locality, loop trip counts...)

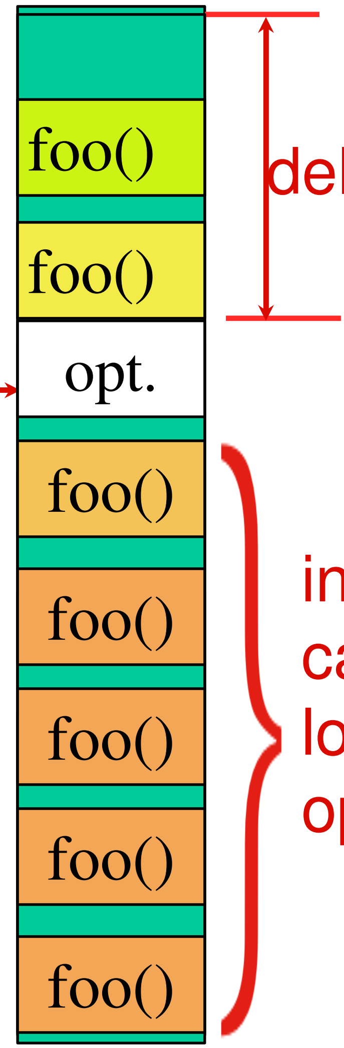
Program Behavior Predictions

Property / Opt	Accuracy	Scope	Timing (Proactivity)
Static Compilation	↗	↗	↗
Profile Feedback	↗	↗	↗
Runtime Adaptive Optimization	↗	↗	↗

Importance of Proactivity

```
while (...){
   foo ();
}
```

overhead



delay

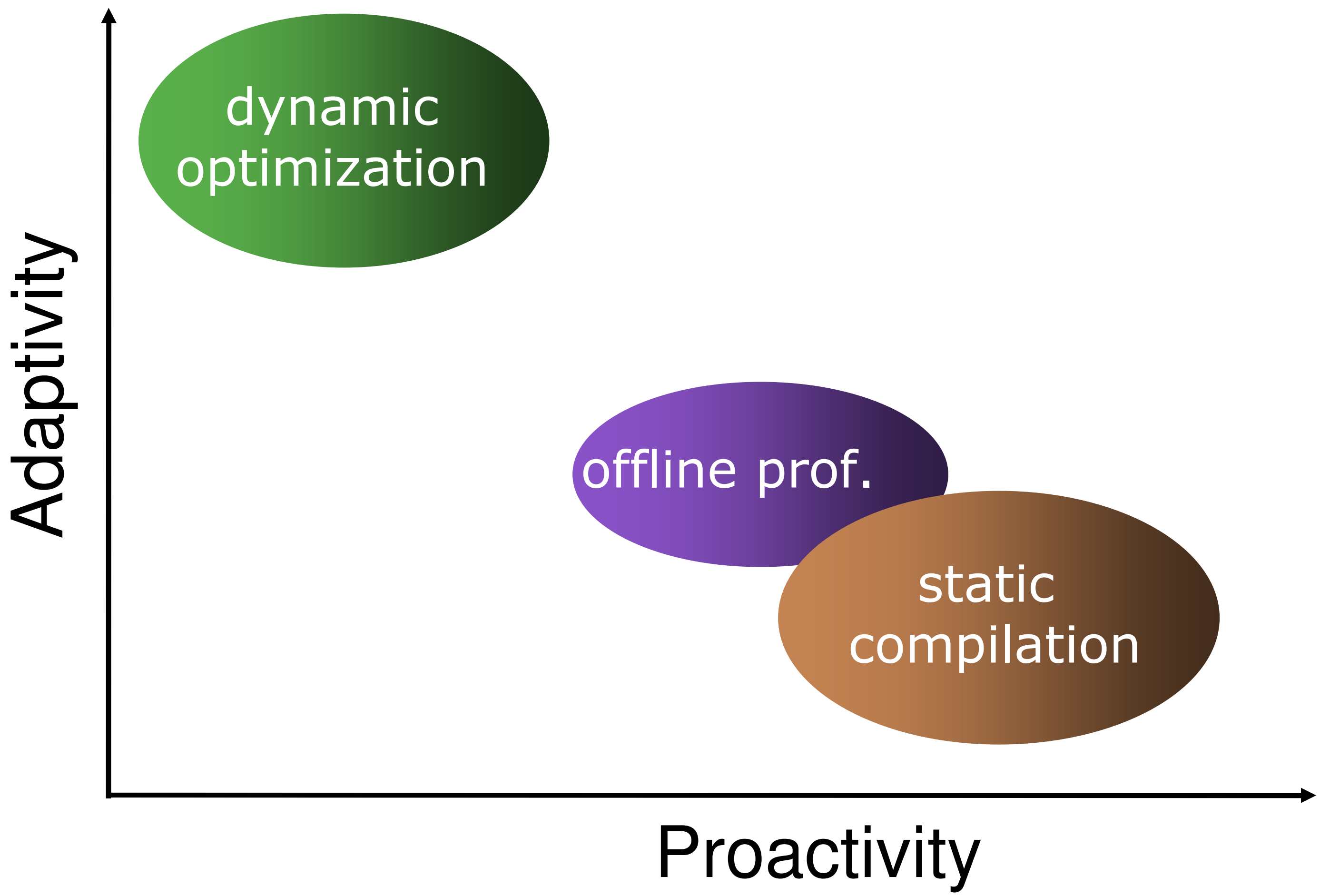
Reactive approach

inferior performance caused by local view-based optimizations

47% on J9 [Arnold+' 05]

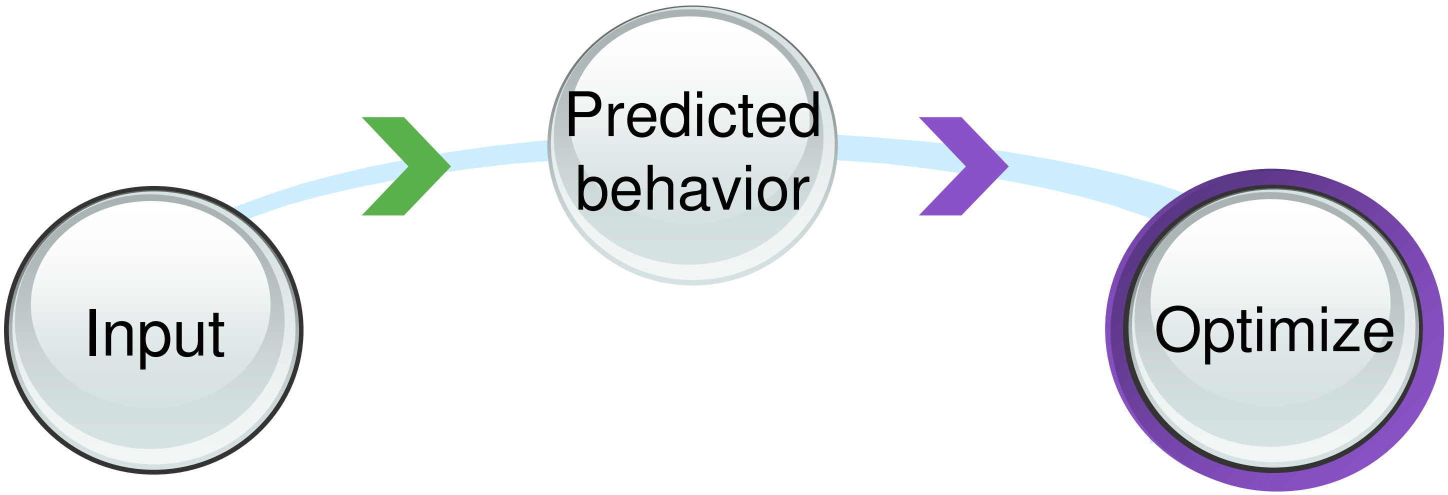
21% on JikesRVM [Mao+' 09]

Adaptivity-Proactivity Dilemma



Prior Solution: Input-Based Prediction

[Mao+:CGO'09]



Idea: Predicting behavior from inputs as program starts

Problem: Requiring manual characterization of inputs

Our Solution

Exploit correlations among program components for proactive runtime prediction and optimization

```
main(int argc, char * argv){
  ...
  mesh_init (dataFile,mesh,refMesh);
  genMesh (mesh,0,mesh->vN);
  verify (mesh, refMesh);
}
```

```
// recursive mesh generation
void genMesh (Mesh *m, int left, int right){
  if (right>3+left){
    genMesh (m, left, (left+right)/2);
    genMesh (m, (left+right)/2+1, right);
    ...}
  ...
}
```

```
void verify (Mesh *m, Mesh *mRef){
    ...
  for (i=0, j=0; i< m->edgesN; i++){
    ...
  }
}
```

```
Mesh * mesh_init
(char * initInfoF, Mesh* mesh, Mesh* refMesh)
{
    // open vertices file, read # of vertices
    FILE * fdata = fopen (initInfoF, "r");
    fscanf (fdata, "%d, %\n", &vN);
    mesh->vN = vN;
    v = (vertex*) malloc (vN*sizeof(vertex));
    // read vertices positions
                                    [i].x, &v[i].y);
      ...}
    // sort vertices by x and y values
    for (i=1; i< vN; i++){
         for (j=vN-1; j>=i; j--){
          ...}
     }
    while (!feof(fd)){
      ...
     // read edges into refMesh for
    // later verification
    }
}
```

Seminal Behaviors

Questions to Answer

- Do such correlations exist commonly?
- How can they be automatically identified?
- Are they useful for program optimizations?

Outline

- A systematic measurement of correlations
- A framework for identification and modeling
- A demonstration of uses for optimizations
- Related work and conclusion

Behaviors under Study

- Loop trip-counts (by a modified GCC)
- Procedure calling frequencies (by GNU gpof v2.19)
- Block access freq. (data profiles) (by IBM XL C 10.1)
- Edge profiles and node profiles (by IBM XL C 10.1)

Correlations to Measure

- Among same types of behaviors of different components
  - E.g. trip-counts of two loops
- Among different types of behaviors of different components
  - E.g. trip-counts *vs* procedure calling freq.

Benchmarks [spec 2000 & 2006]

Program				Factor of changes
name	lines	inputs	loops	caused by inputs
ammp	13263	20	425	9.9×10^1
art	1270	108	101	4.0×10^4
crafty	19478	14	425	4.6×10^8
equake	1513	100	106	1.0×10^2
gap	59482	12	1887	1.1×10^8
gcc	484930	72	7615	1.1×10^6
gzip	7760	100	223	4.3×10^7
h264ref	46152	20	2074	2.1×10^9
lbm	875	120	27	6.0×10^6
mcf	1909	64	76	1.4×10^5
mesa	50230	20	995	2.0×10^1
milc	12837	10	473	2.1×10^9
parser	10924	20	1350	2.1×10^6
vpr	16976	20	435	3.9×10^6

[Thanks to Amaral's group for extra inputs]

Calculation of Correlations

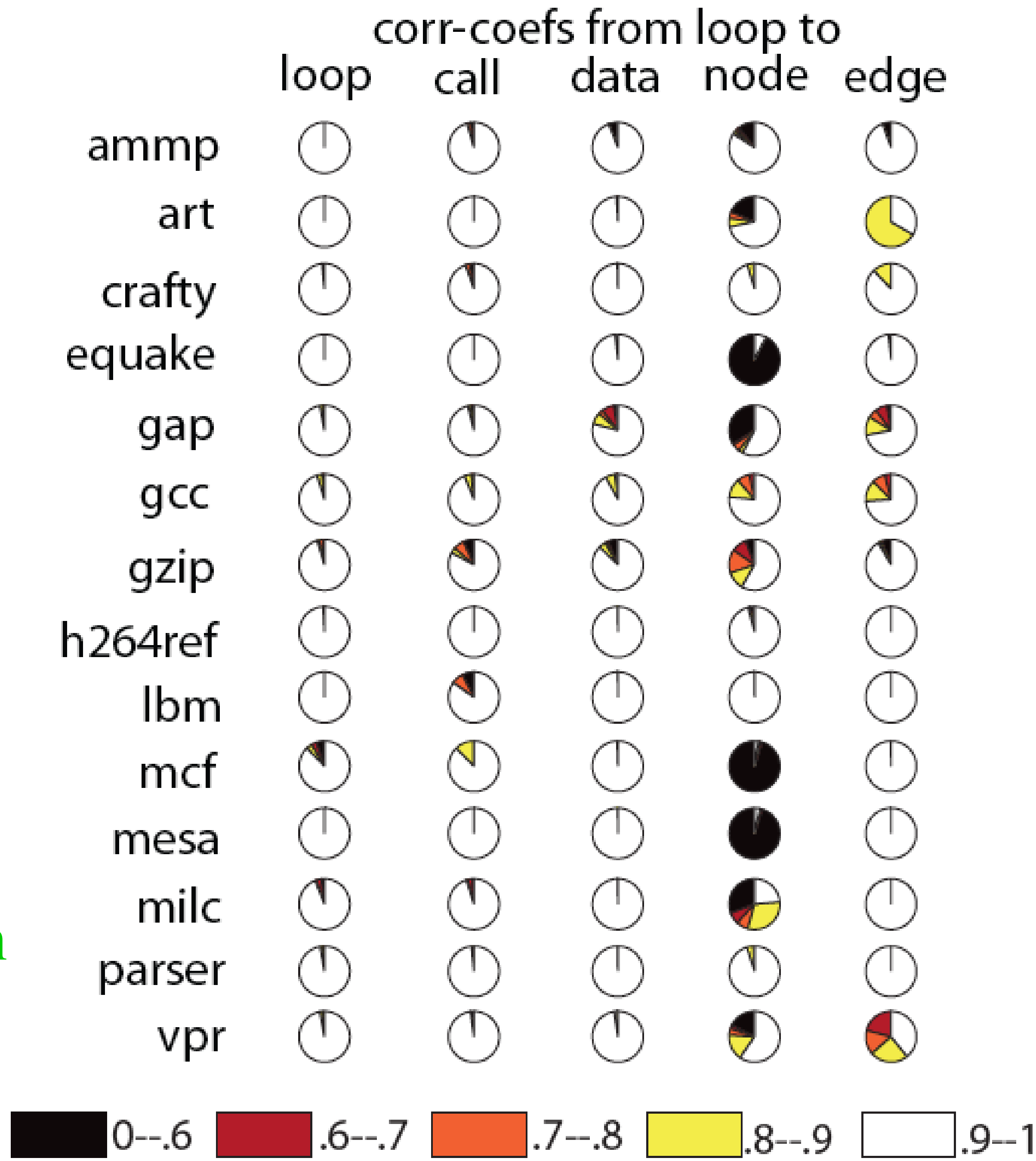
$$r_{XY} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_X s_Y}$$

Sample standard deviation

The higher r is, the easier to predict one from the other.

Strong correlations from loops to loops and to other behaviors

Uses for runtime behavior prediction



Outline

- A systematic measurement of correlations
- A framework for identification and modeling
- A demonstration of uses for optimizations
- Related work and conclusion

Two Goals

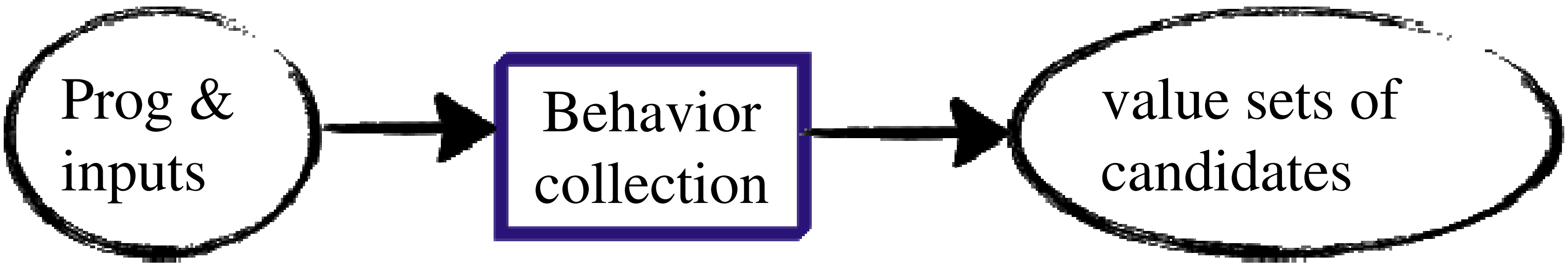
- Identify Seminal Behaviors
- Build predictive models

Target behavior value = f (values of seminal behaviors)

Seminal Behaviors

- A small set of program behaviors
  - Predictive capability
    - Strongly correlate with target behaviors
  - Earliness
    - Values become known early in an execution

Identification of Sem Beh

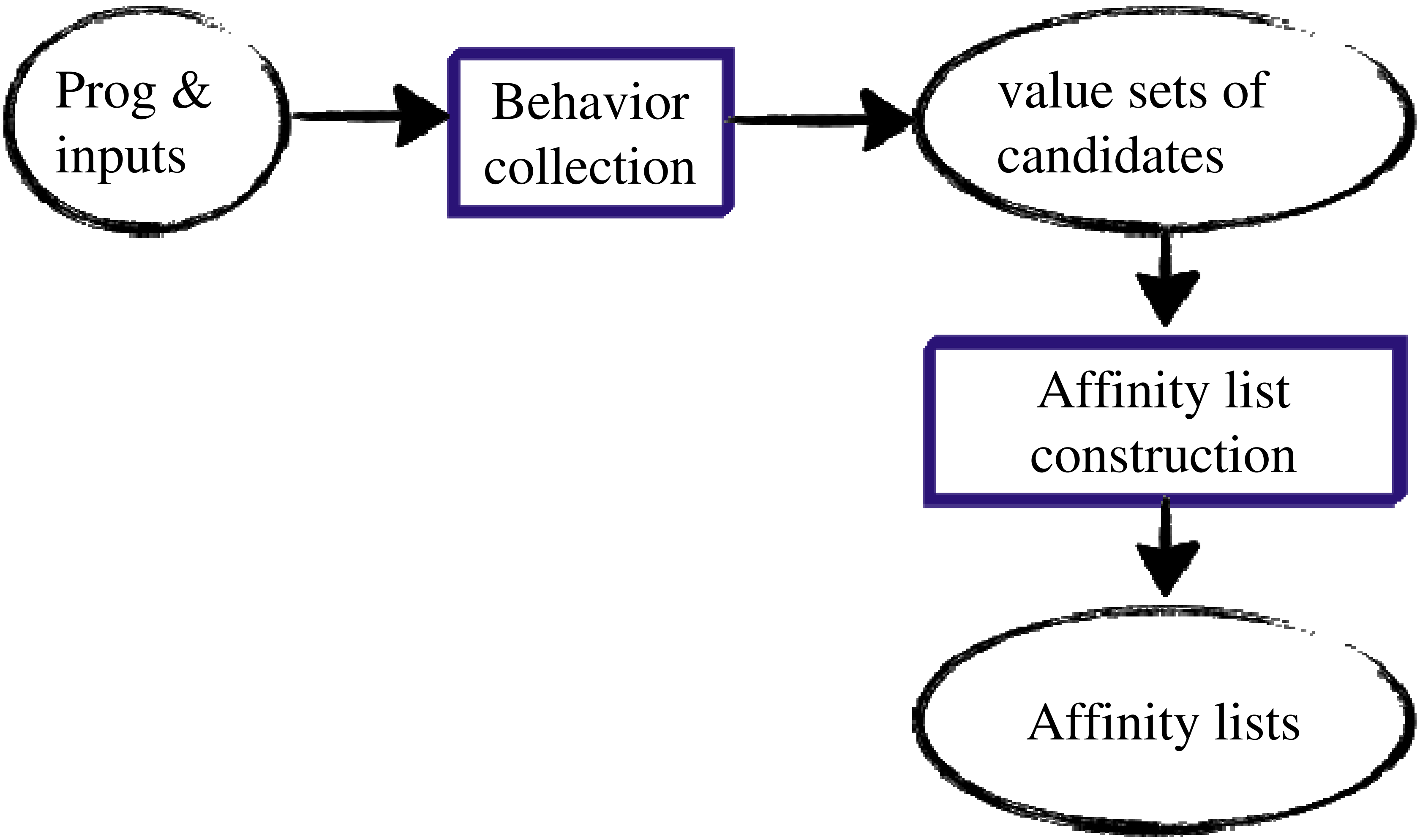


Candidate Seminal Behaviors

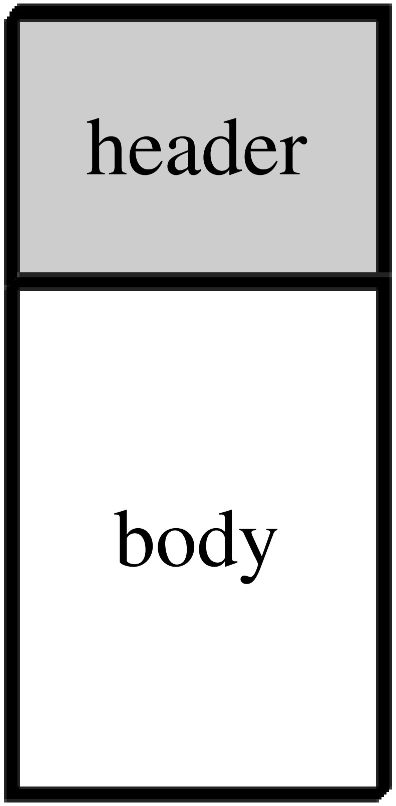
- Interface behaviors
  - Values directly obtained from program inputs
  - Ignore massive file content
    - Include corresponding loop trip-counts

- Loop trip-counts
  - Importance in programs and strong correlations with other behaviors

Recognition of Sem Beh

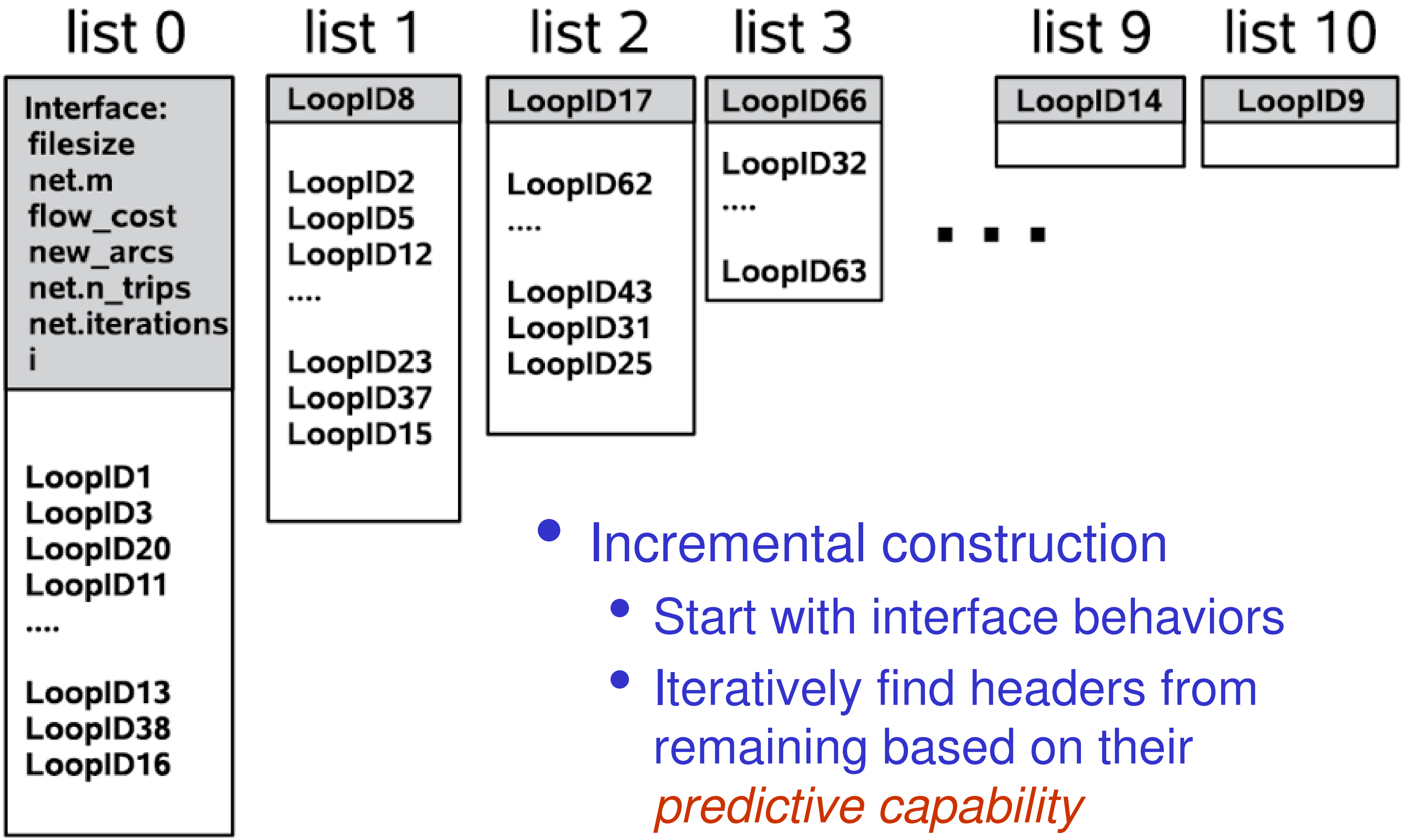


Behavior Affinity List



Header can predict body accurately.

Affinity List of *mcf*

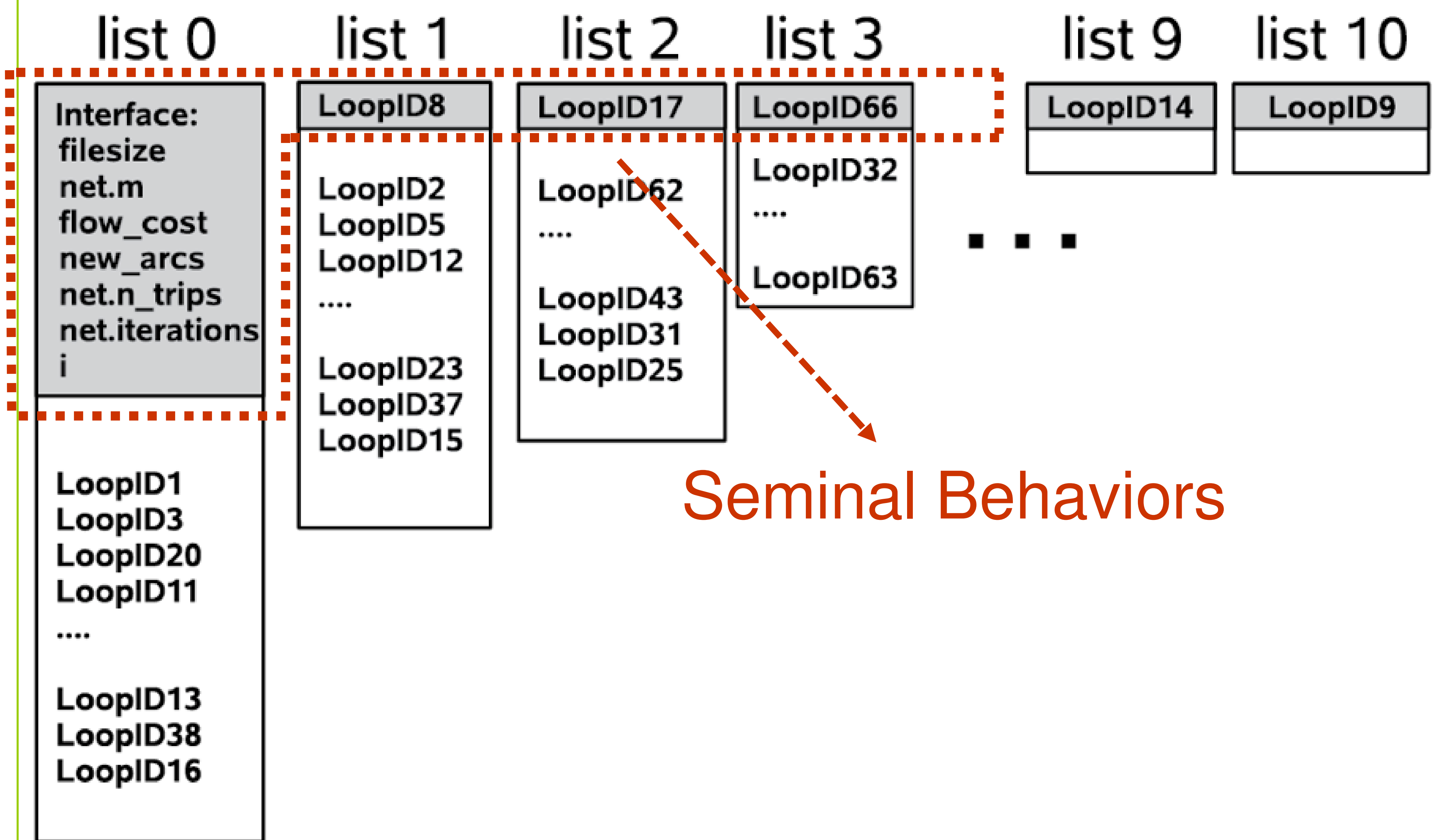


- Incremental construction
  - Start with interface behaviors
  - Iteratively find headers from remaining based on their *predictive capability*

Predictive Capability

- Predictive models
  - LMS (Least Mean Square)
  - Regression Trees
- Compute predictive capability
  - 10-fold cross-validation

Refine by *predictive capability* & *earliness*



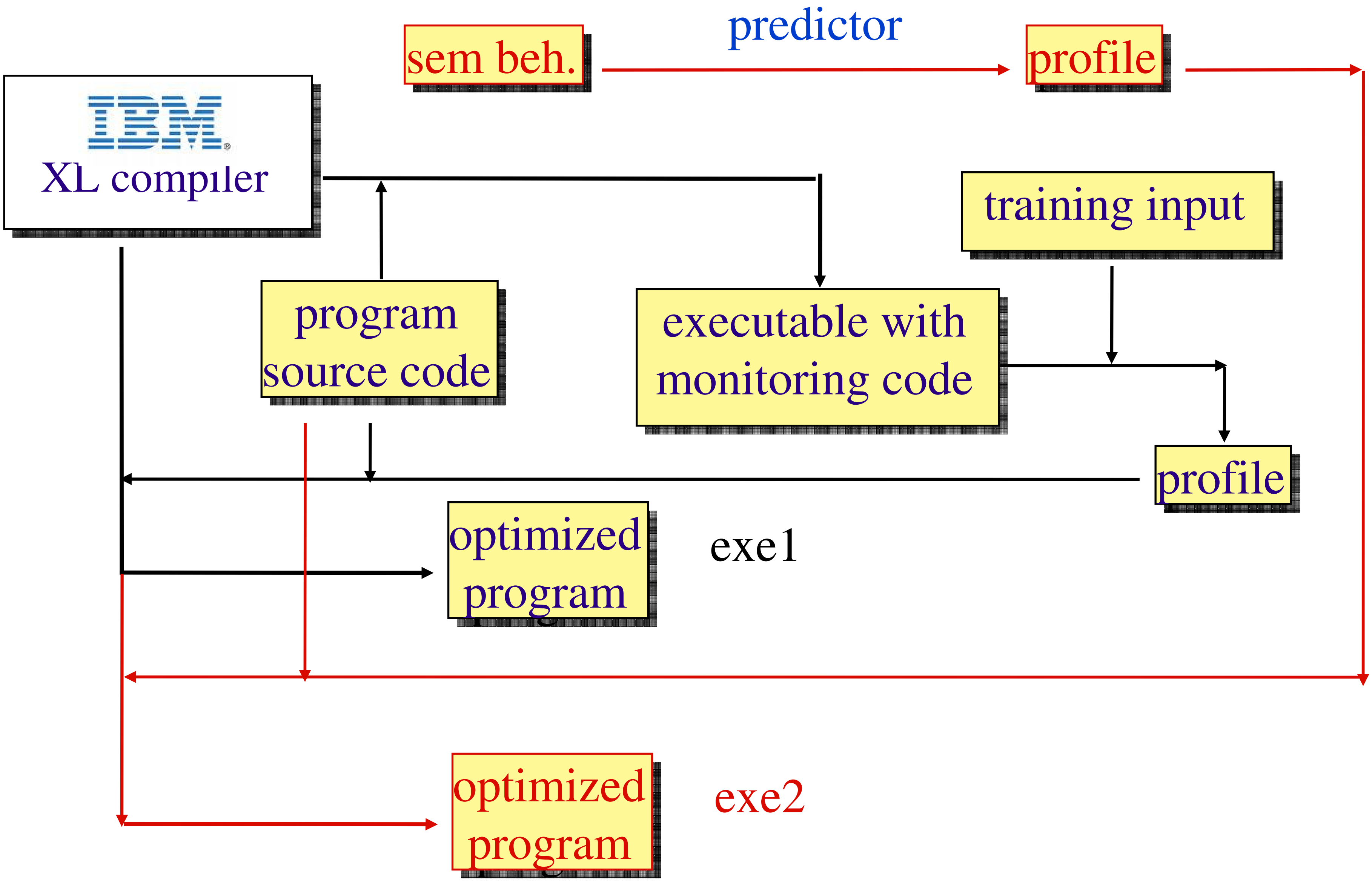
Seminal Behavior Based Predict

Num of seminal behaviors and prediction accuracy

Prog	interface values						earliness ≥ 90%					
	num	accuracy					num	accuracy				
		loop	call	edge	node	data		loop	call	edge	node	data
ammp	1	99.5	96.7	100	91.1	99.7	1	99.5	96.7	100	91.1	99.7
art	4	91.0	96.8	100	82.0	96.8	4	91.1	96.8	100	80.0	96.1
crafty	1	89.9	58.9	88.2	35.5	76.0	2	91.1	63.0	90.8	44.5	79.3
equake	1	98.0	100	100	96.3	99.3	1	98.0	100	100	96.3	99.3
gap	2	97.5	44.9	11.9	44.2	76.6	7	99.5	78.7	56.3	69.7	88.5
gcc	4	82.9	38.9	56.2	61.0	78.5	54	97.0	86.1	93.6	95.4	95.6
gzip	3	92.2	87.0	84.1	67.5	94.5	6	91.6	87.6	83.5	69.0	94.5
h264ref	3	99.8	99.8	98.7	98.8	99.8	4	99.8	99.7	97.0	97.8	99.7
lbm	3	99.8	90.1	100	100	100	3	99.8	90.1	100	100	100
mcf	5	87.3	87.7	100	92.2	97.8	10	92.2	91.0	100	89.5	97.5
mesa	1	100	100	99.5	12.2	100	1	100	100	99.5	12.2	100
milc	2	79.2	72.1	37.1	27.4	93.9	18	83.0	72.8	100	52.0	99.7
parser	1	90.2	85.4	73.8	75.9	87.6	2	91.8	88.0	79.2	78.0	90.8
vpr	3	93.3	95.1	60.4	81.9	94.6	9	95.2	95.5	64.0	82.2	95.8
Average	2.4	92.9	82.4	79.3	69.0	92.5	8.7	95.0	89.0	90.3	75.5	95.5

Outline

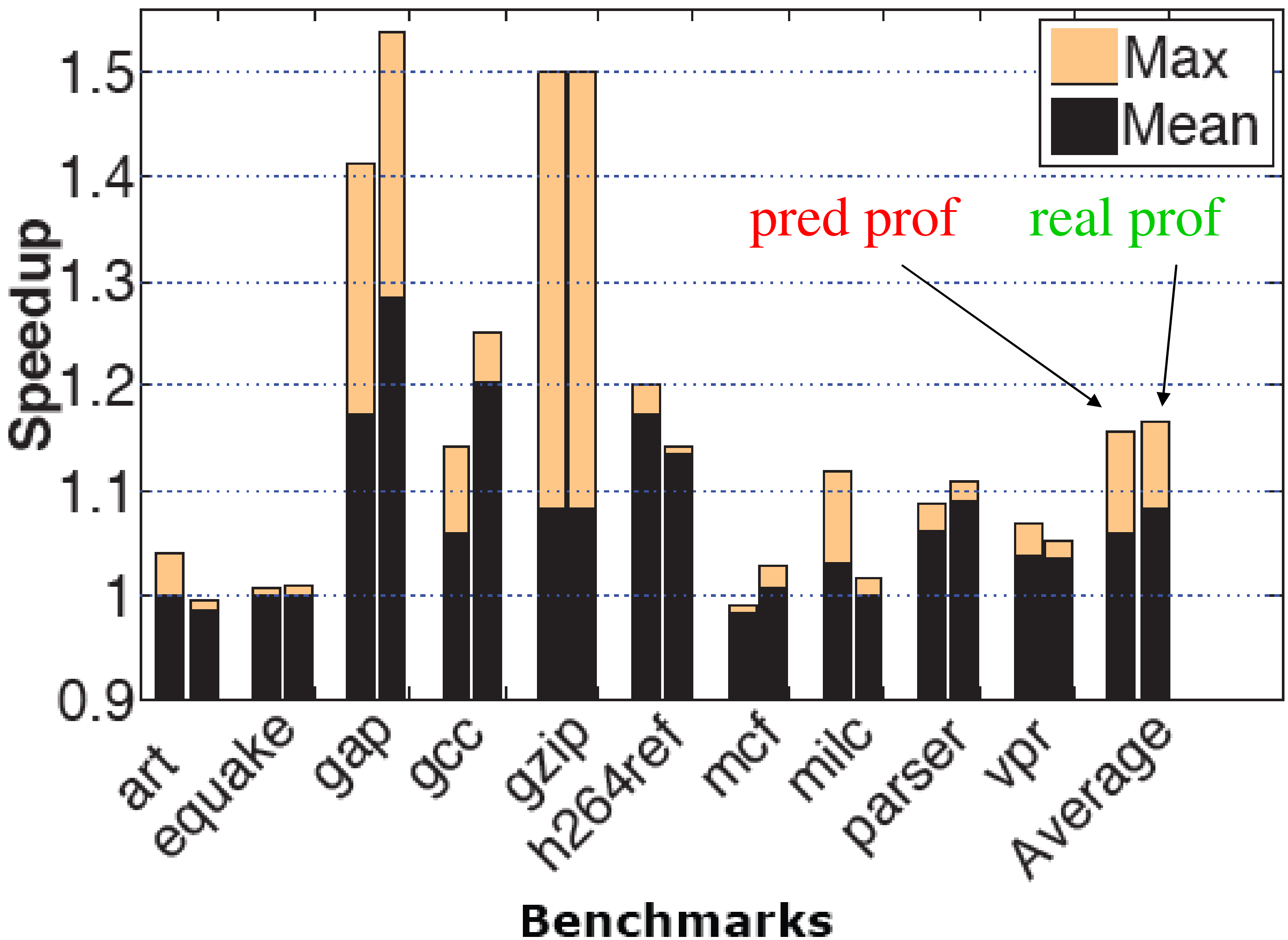
- A systematic measurement of correlations
- A framework for identification and modeling
- A demonstration of uses for optimizations
- Related work and conclusion



predictor
sem beh.
profile
IBM
XL compiler
training input
program source code
executable with monitoring code
profile
optimized program
exe1
optimized program
exe2

Performance Improvement

(Baseline: highest static opt)



IBM Power5

XL 11.1

More Potential Uses

- Help JIT compilers make better decisions in managed environment
  - i.e. JVMs
- Boost performance through dynamic code version selection
  - for imperative languages such as C

Outline

- A systematic measurement of correlations
- A framework for identification and modeling
- A demonstration of uses for optimizations
- Related work and conclusion

Related Work

- Correlations between control flow signatures and hardware performance
  - [Sherwood+:ASPLOS'02, Annavaram+:Micro 04, etc.]
- Adaptive dynamic optimization
  - [Arnold+:OOPSLA'00, Chen+:PLDI'06,Lau+:PLDI'06, etc.]
- Exploiting inputs for optimization
  - [Wang+:PLDI'04, Mao+:CGO'09, Chen+:PLDI'10]

Conclusion

- Strong correlations exist among behaviors.
- Seminal behavior-based technique is promising.
- Significant potential for program optimizations.

Thanks!

Questions?

Kai Tian
ktian@cs.wm.edu