Input-Centric Program Behavior Analysis & Optimizations

Xipeng Shen

Computer Science Department
The College of William and Mary, Williamsburg, VA

What is the common, fundamental prerequisite for program optimizations?

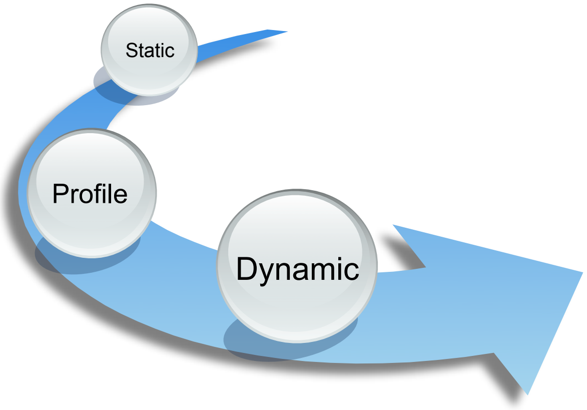
Prediction of how the program would behave.

Program Behaviors

(calling freq, locality, loop tripcount, ...)

Program Behavior Analysis

- Goal is to uncover patterns for prediction
- Current approaches



Our Goal

A new paradigm:

input-centric program behavior analysis.

Include program inputs into the focus.

Outline

- Why input-centric?
- How to exploit inputs for program optimizations?

What are inputs?

- All the data that are not generated but accessed by the program
  - command arguments
  - input files
  - ... ...

Why input-centric?

Strong and predictive correlations between inputs and behaviors.

Better behavior analysis.

Better prediction.

Better optimizations.

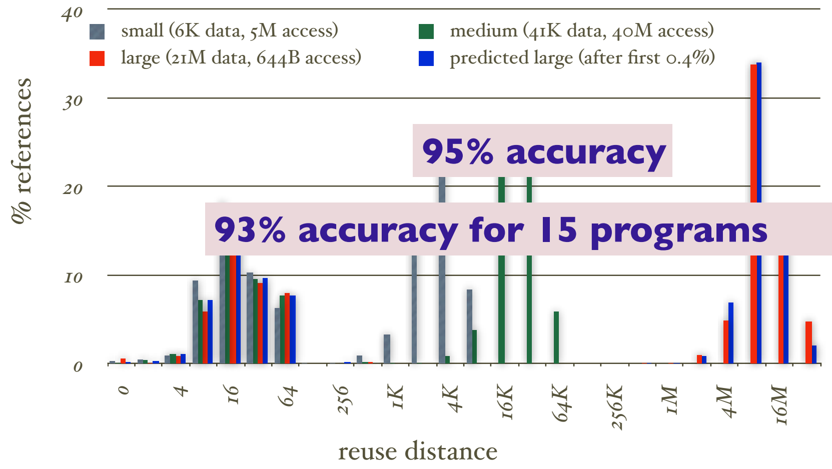
Qualitative View

Prog Beh = Code + Inputs + Running Environments

Input is the only deciding factor for a given program in a given environment.

Quantitative Evidence

- Reuse distance histograms of *lucas* [Zhong+:TOPLAS'09]



95% accuracy

93% accuracy for 15 programs

Quantitative Evidence (cont.)

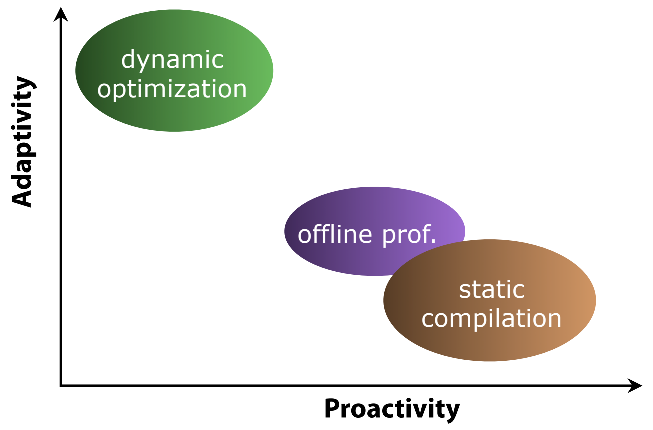
- JIT optimization levels [Mao+:CGO'09]
- Profitability of speculation [Jiang+:ICPADS'09]
- Minimum required heap size [Mao+:VEE'09]
- Optimization parameters for GPU [Liu+:IPDPS'09]
- Cache contention on CMP [Jiang+:EuroPar'08]

Current Treatments to Inputs

- Static compilation: code only.
- Offline profiling: not adapt to input changes.
- Runtime sampling: no explicit treatment to inputs, hence loses proactivity in prediction and optimizations.

predicting behaviors before or early in a run.

Adaptivity-Proactivity Dilemma

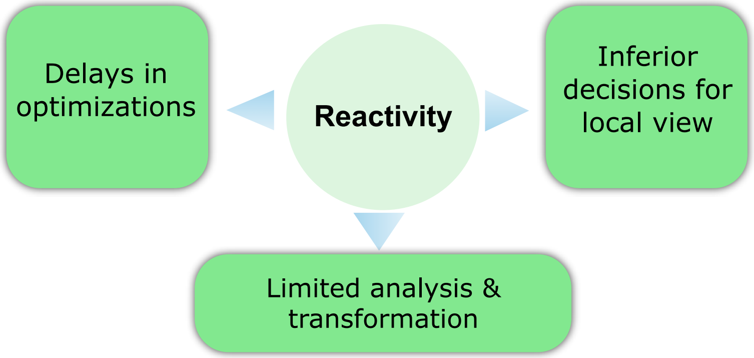


dynamic optimization

offline prof.

static compilation

Drawbacks of Reactivity


Delays in optimizations
Reactivity
Inferior decisions for local view
Limited analysis & transformation

Potential

21%

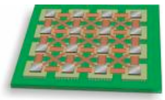
JVM [CGO'09]

50%

Parallelization [ICPADS'09]

12%

GC[VEE'09]


14%

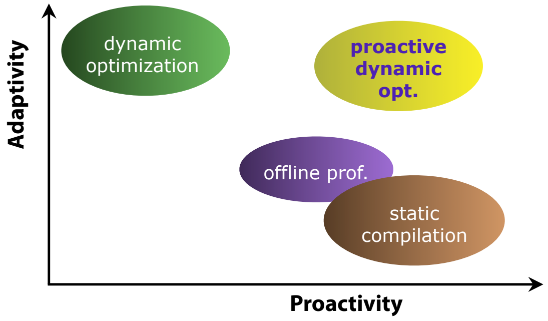
CMP Sched. [PACT'08]

2.8X

GPU [IPDPS'09]

Opportunities from Inputs

- Inputs come early
- Strong predictive input-behavior correlations lead to proactive prediction.
- The prediction is meanwhile cross-input adaptive.

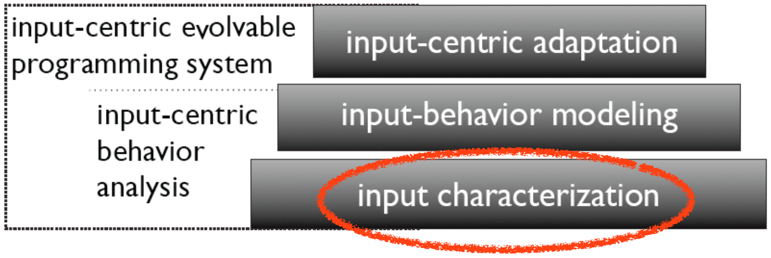


Outline

- Why inputs?

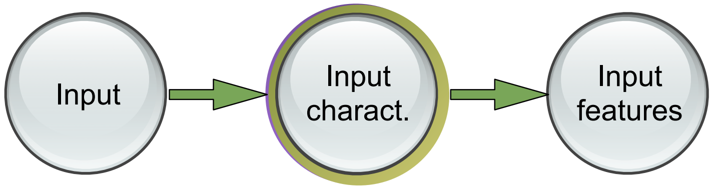
- How to exploit inputs for program optimizations?

Overview


input-centric evolvable programming system
input-centric behavior analysis
input-centric adaptation
input-behavior modeling
input characterization

Input Characterization

To extract important features from raw inputs



Challenges

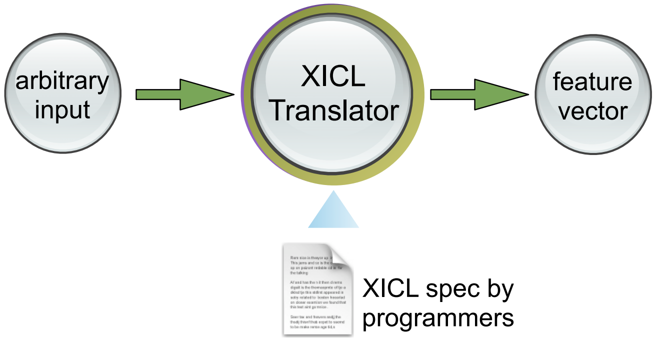
- Input attributes rather than values matter
  - e.g., data distribution
- Complex input syntax & semantics
  - e.g., a graph or a tree or a signal
- Interplay among input components
  - overshadow, equivalence, default values, etc.

Domain knowledge needed;
automatic solutions are difficult.

Specification-Based Solution

[CGO'09]

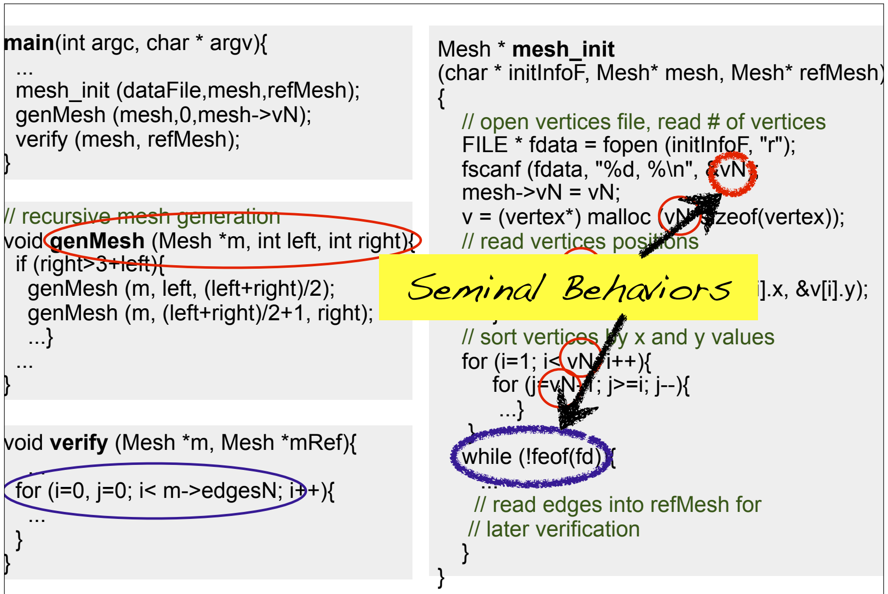
- e**X**tensible **I**nput **C**haracterization **L**anguage



Automatic Solution

Seminal-Behavior Analysis

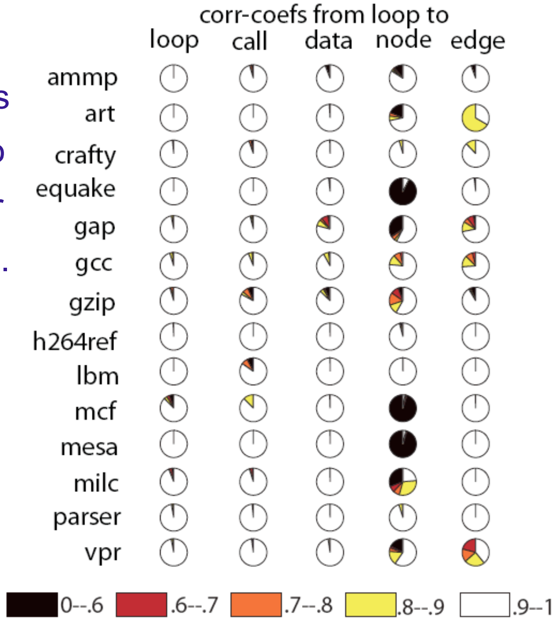
- Key observation: correlations in a program.



```
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
                              ].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
  }
}
```

Strong correlations exist from loops to loops and to other types of behaviors.



Intuition

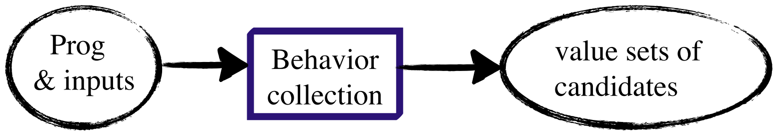
Prog Beh = Code + Inputs + Running Environments

Input is the only deciding factor for a given program in a given environment.

Seminal Behaviors

- Definition (informal)
  - Behaviors that can lead to accurate prediction of all behaviors of interest, and appear early in a run.
- Reflection of critical program input features.
- Implication
  - Enable proactive & adaptive optimizations.
  - Remove the needs for explicit input characterizations.

Recognition of Sem Beh


Prog & inputs

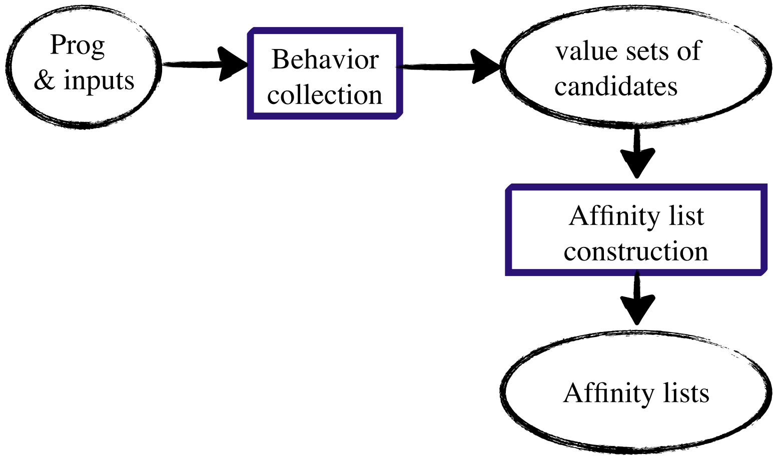
Behavior collection

value sets of candidates

Candidate Seminal Behaviors

- Loop trip-counts
- Interface behaviors
  - values directly obtained from program inputs.
  - ignore massive file content
    - include corresponding loop trip-counts

Recognition of Sem Beh


Prog & inputs

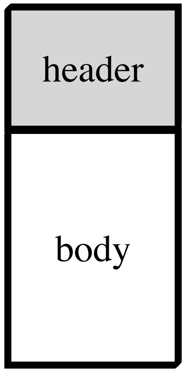
Behavior collection

value sets of candidates

Affinity list construction

Affinity lists

Behavior Affinity List



Header can predict body accurately.

Affinity List of *mcf*

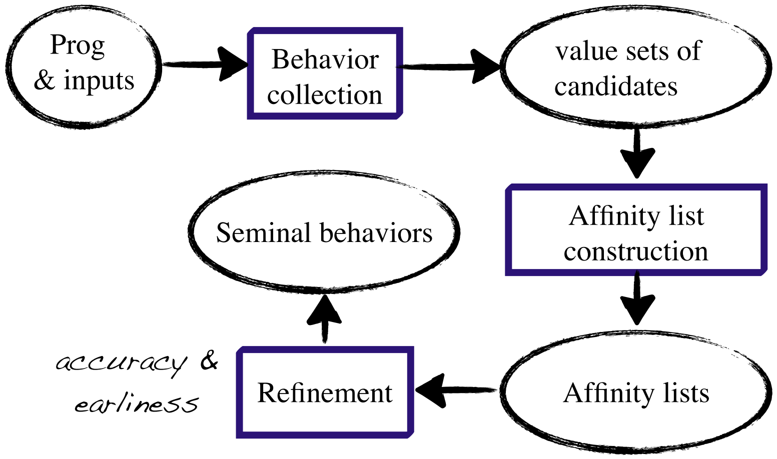
list 0	list 1	list 2	list 3		list 9	list 10
Interface: filesize net.m flow_cost new_arcs net.n_trips net.iterations i	LoopID8	LoopID17	LoopID66	. . .	LoopID14	LoopID9
LoopID1 LoopID3 LoopID20 LoopID11 .... LoopID13 LoopID38 LoopID16	LoopID2 LoopID5 LoopID12 .... LoopID23 LoopID37 LoopID15	LoopID62 .... LoopID43 LoopID31 LoopID25	LoopID32 .... LoopID63			

- Incremental construction
  - start with interface behaviors
  - iteratively find headers from the rest based on their predictive capability.

Predictive Capability

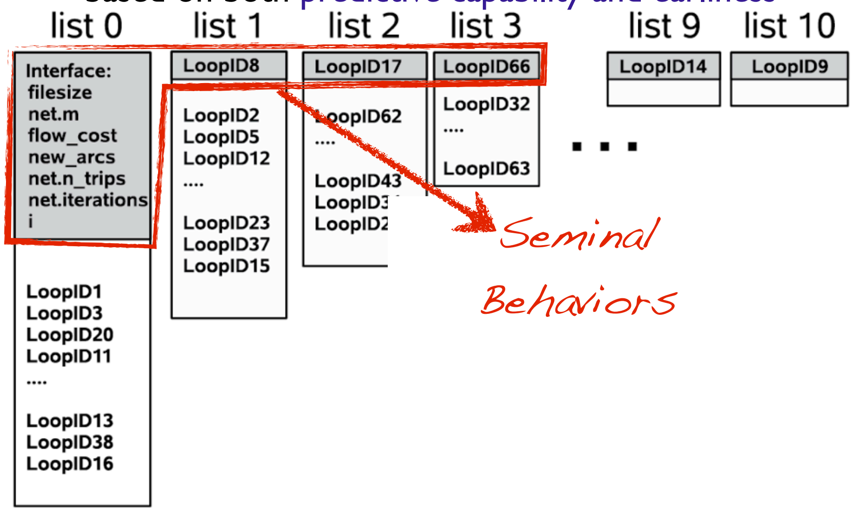
- Regression models
  - LMS (Least Mean Square)
  - Regression Trees
- 10-fold cross-validation

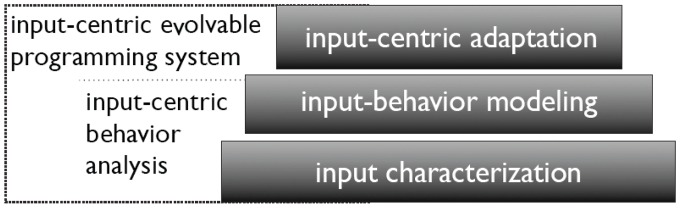
Recognition of Sem Beh



Refinement

based on both predictive capability and earliness




input-centric evolvable programming system
input-centric adaptation
input-centric behavior analysis
input-behavior modeling
input characterization

Modeling and Adaptation

- Modeling --- construct predictive models

  Target Behaviors = f (Seminal Behaviors)

  - Machine learning problem
  - Classification (e.g., for optimization levels)
  - Regression (e.g., for calling frequencies)
- Adaptation
  - Runtime version selection
  - JIT
  - Dynamic speculation
  - ... ...

Evaluation

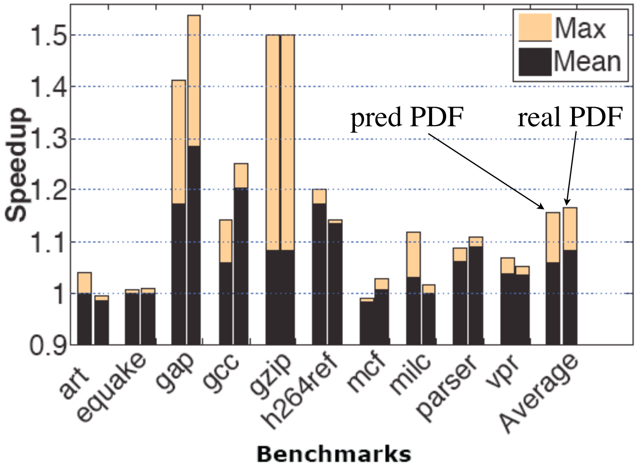
- Predictive capability of seminal behaviors
- Potential for program optimizations

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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
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

100+ options, 130 files, 484930 lines of code, 7615 loops

Speedup by PDF Compilation on Pred & Real Profiles

(IBM Power5 with XL v11.1)



Conclusions

- Inputs strongly correlate with program behaviors and are beneficial to exploit.
- Input-centric behavior analysis is a promising solution.

Acknowledgment

- Students
  - Yunlian Jiang, Eddy Z Zhang, Kai Tian, Yixun Liu, Feng Mao, Malcom Geathers
- C. Ding at Univ of Rochester offered the behavior equation and the initial insights on inputs.
- Compiler support from the TPO group at IBM Toronto Lab.
- Some program inputs from J. Amaral's group at Univ of Alberta.
- Equipments from IBM, DoE Jefferson Lab, NVIDIA.
- NSF & IBM provided funding support.

Thanks!
Questions?