Program Behavior Sequence Prediction

Bo Wu, Yunlian Jiang, Xipeng Shen
(The College of William & Mary)
Yaoqing Gao, Raul Silvera, Graham Yiu
(IBM Toronto)

Outline

- Motivation
- Our perspectives
- Behavior sequence prediction framework
- Some results of loop trip count prediction
- Possible uses
- Summary

Motivation

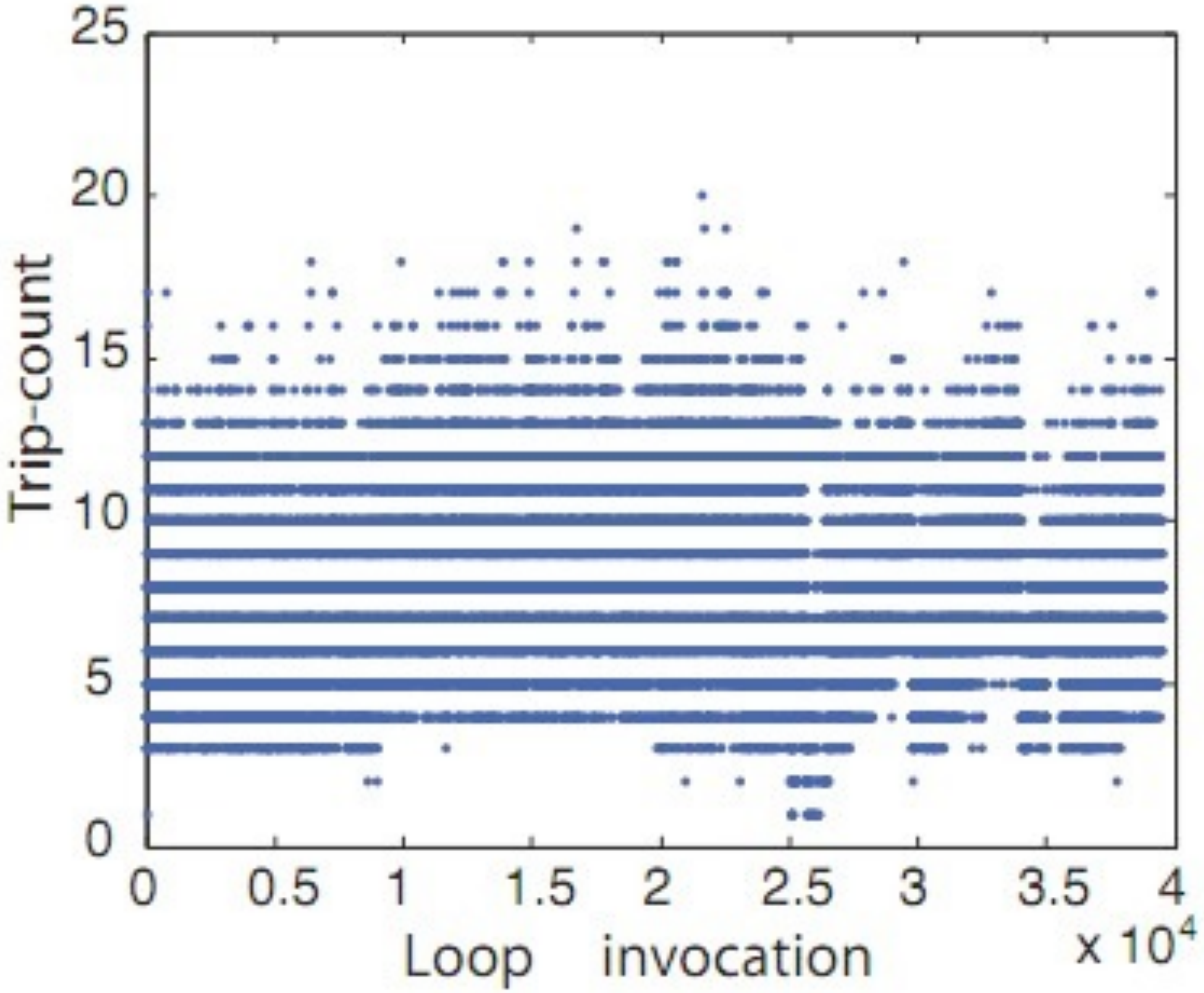
- Accurate and proactive prediction of program behaviors is essential for many optimizations
  - Loop trip counts for loop unrolling
  - Function hotness for function optimization level in JIT
  - profitability for speculative parallelization
  - Cache miss rates for prefetching aggressiveness
  - Loop coldness for outlining
  - ......

Motivation

- The usefulness is not limited to program optimizations
  - OS level
    - Provision in cloud computing
    - Scheduling to reduce resource contention
  - Architecture level
    - Voltage scaling

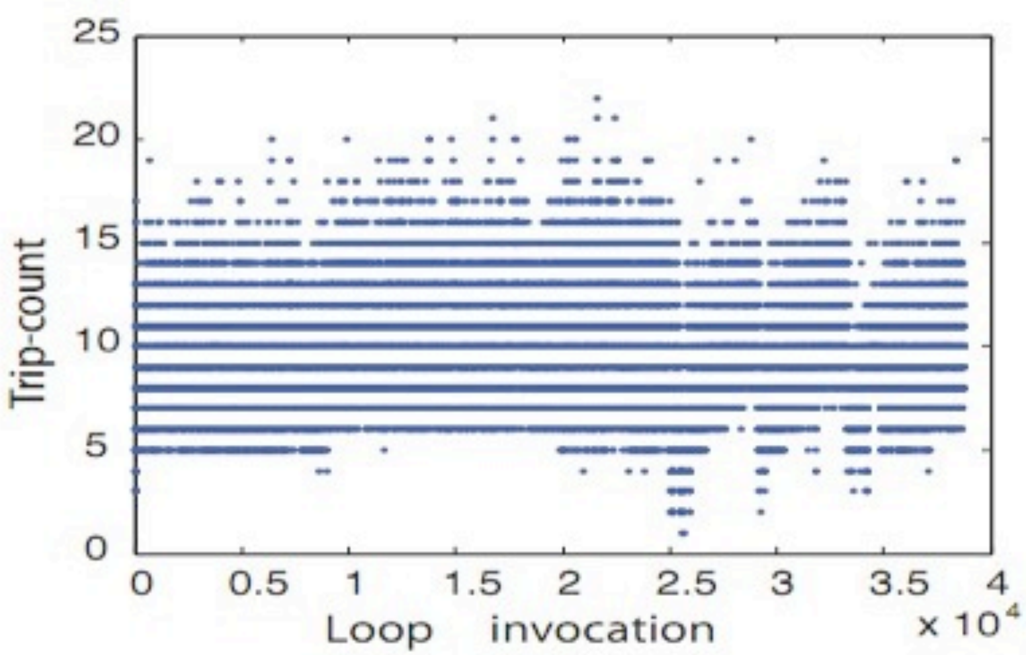
Motivation

- However, the prediction of program behaviors is challenging

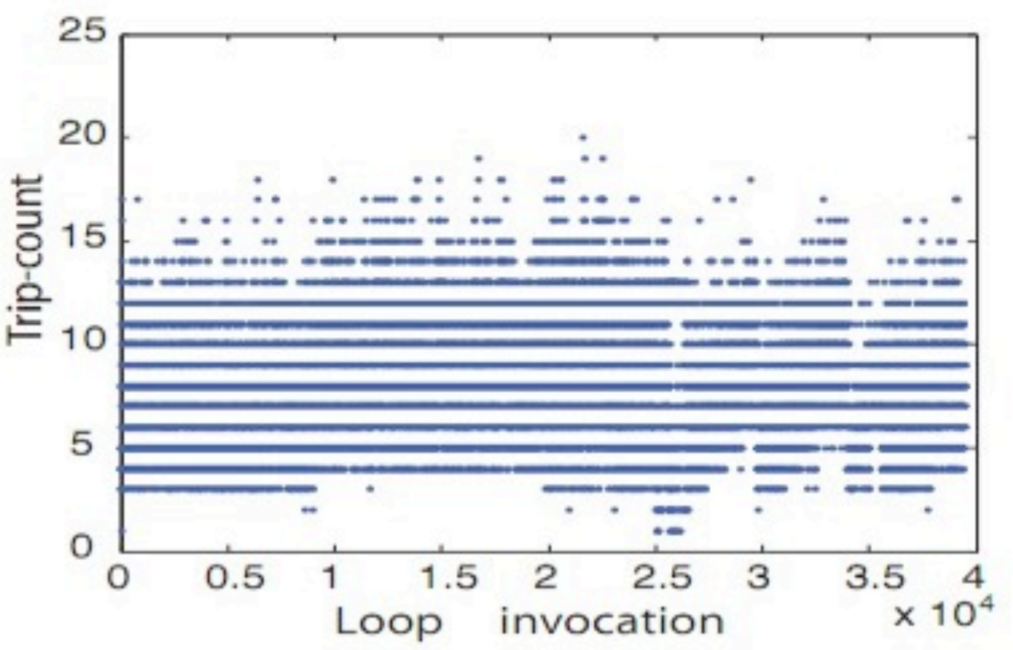


Motivation

- Opportunities do exist



(a) loop 1 trip-count sequence



(b) loop 2 trip-count sequence

```
         int contains_underbar(char * s) {
            /*  finding an underscore */
Loop 1:     while(*s != '\0') {
               if (*s == '_') return TRUE;
               s++;
            }}
```

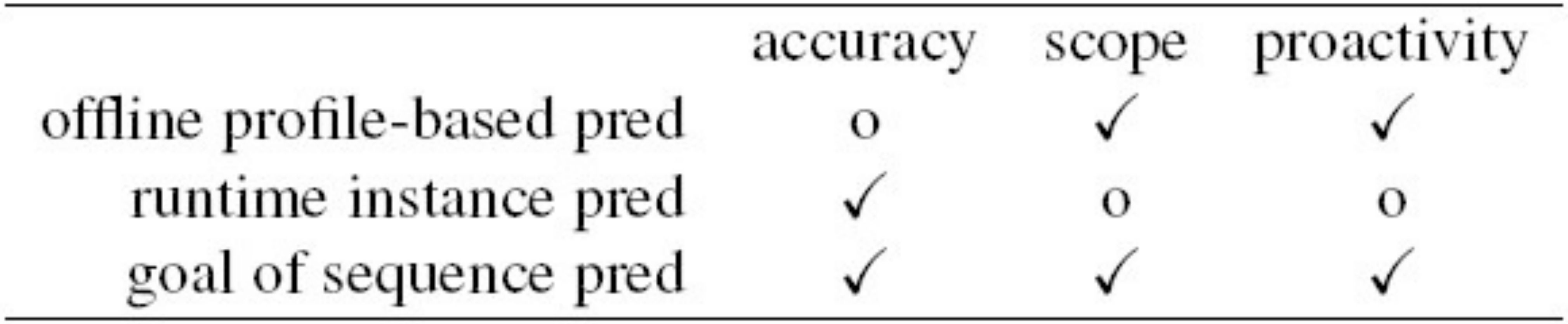
```
         int numberfy(char * s){
            /* finding a decimal point */
Loop 2:     for (; (*s != '\0') && (*s != '.'); s++)
             ...
         }
```

Our Perspectives

- Difference between instance prediction and sequence prediction
  - Instance prediction: the next one or several instances
  - Sequence Prediction: the whole sequence of the considered behavior
- Statistical correlation among different behaviors
  - Trip counts of two different loops
  - Loop trip counts and function hotness
- Context awareness
  - Loop stack and call stack
  - Correlated behaviors happened before

Our Perspectives

- Three requirements for behavior prediction
  - Accuracy
  - Proactivity
  - Scope



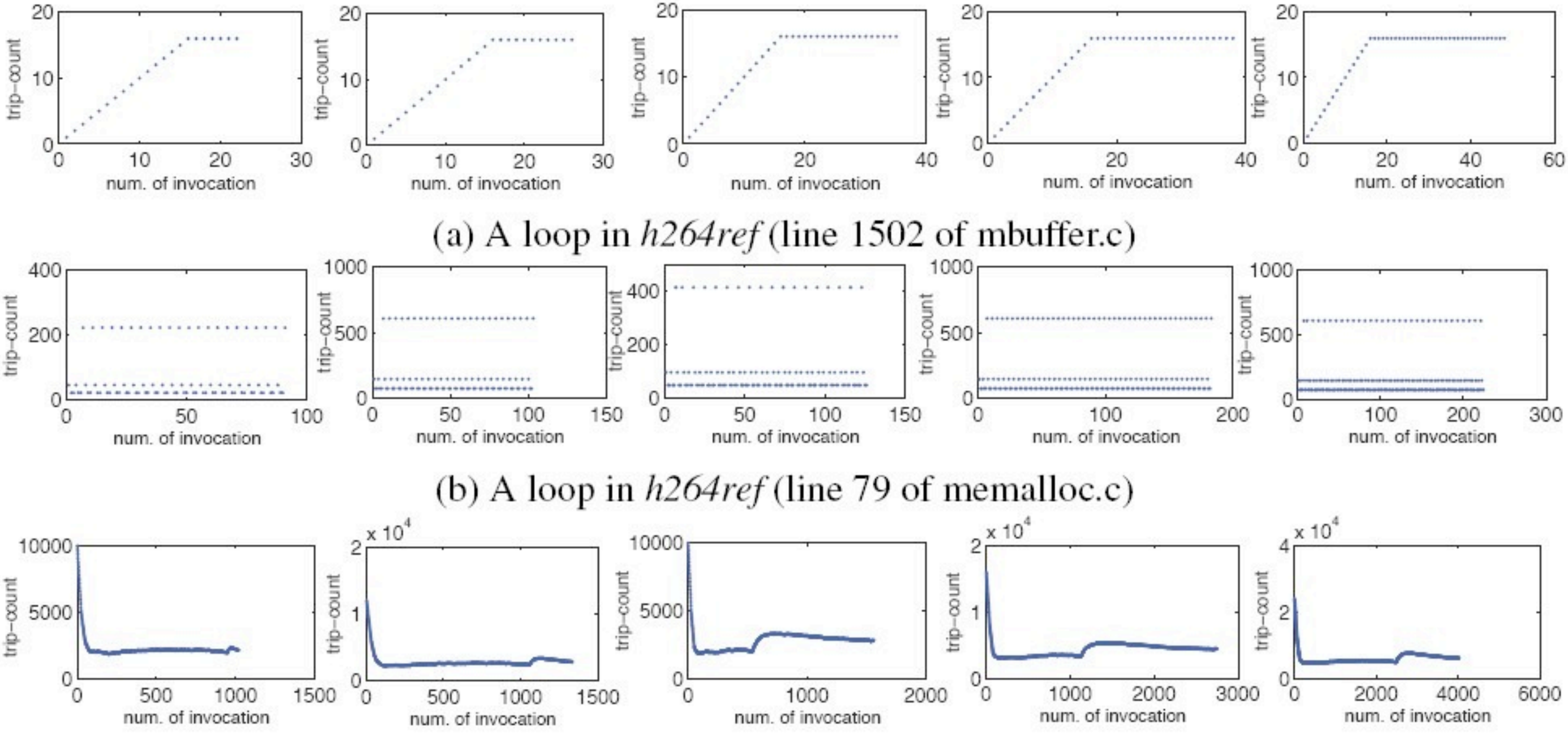
	accuracy	scope	proactivity
offline profile-based pred	o	✓	✓
runtime instance pred	✓	o	o
goal of sequence pred	✓	✓	✓

Sequence Prediction Framework

- The initial study is on loop trip counts prediction
    - Loops are dominant parts
    - Resource requirements
    - inlining
    - Computation granularity
    - ...

Sequence Prediction Framework

- Loop trip count sequences follow patterns



(a) A loop in *h264ref* (line 1502 of mbuffer.c)

(b) A loop in *h264ref* (line 79 of memalloc.c)

(c) A loop in *mcf* (line 52 of pstart.c)

Sequence Prediction Framework

- Three steps
  - Simplification

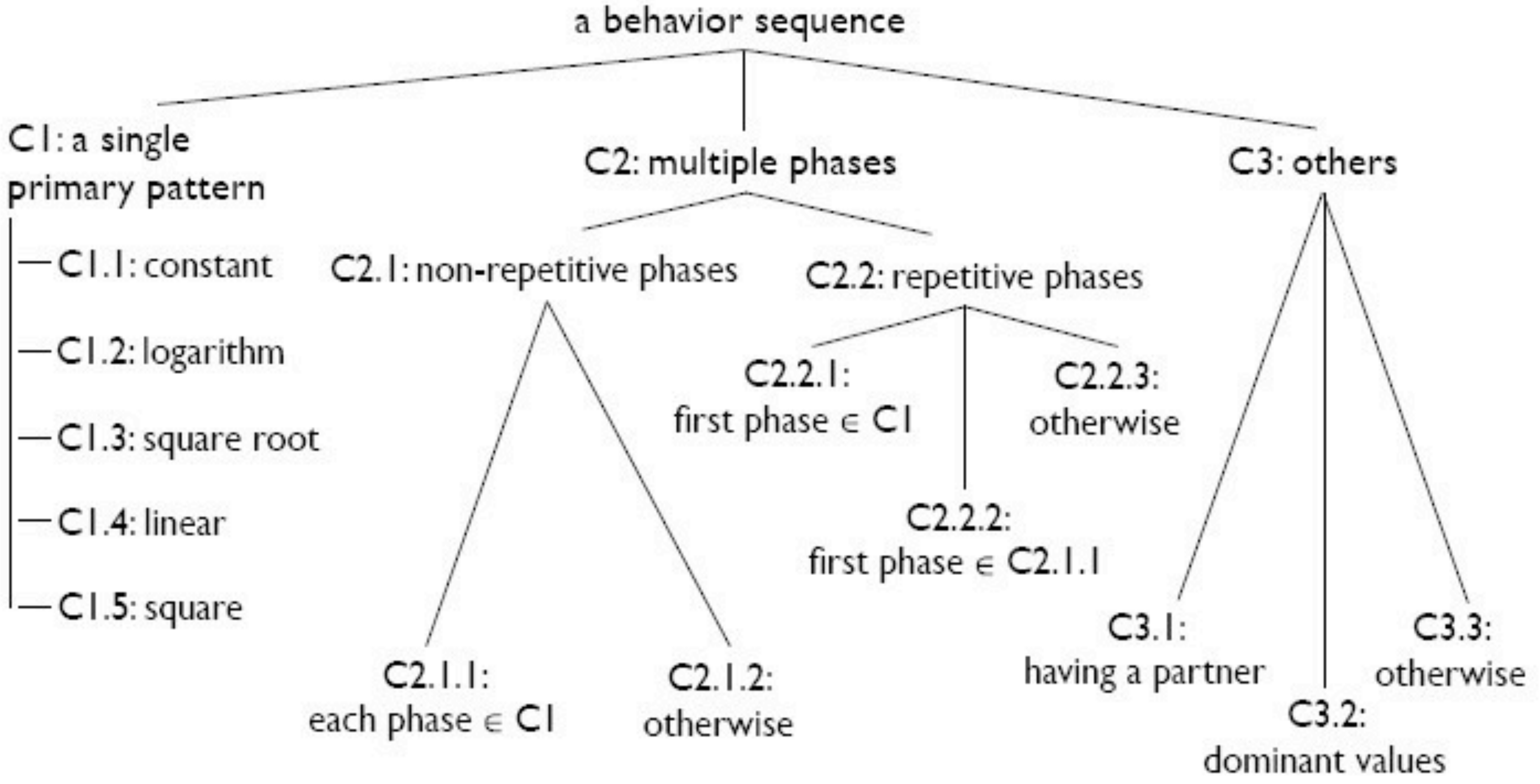
    Recognize the pattern of a sequence and use several features to represent it
  - Prediction

    Predict the sequence features through correlation
  - Generation

    Reconstruct sequences from the predicted features

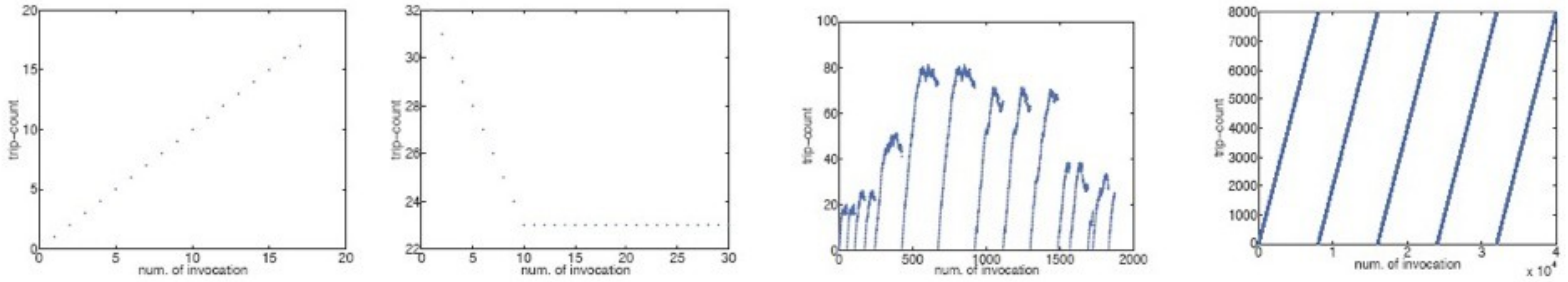
Sequence Prediction Framework

- Pattern Recognition



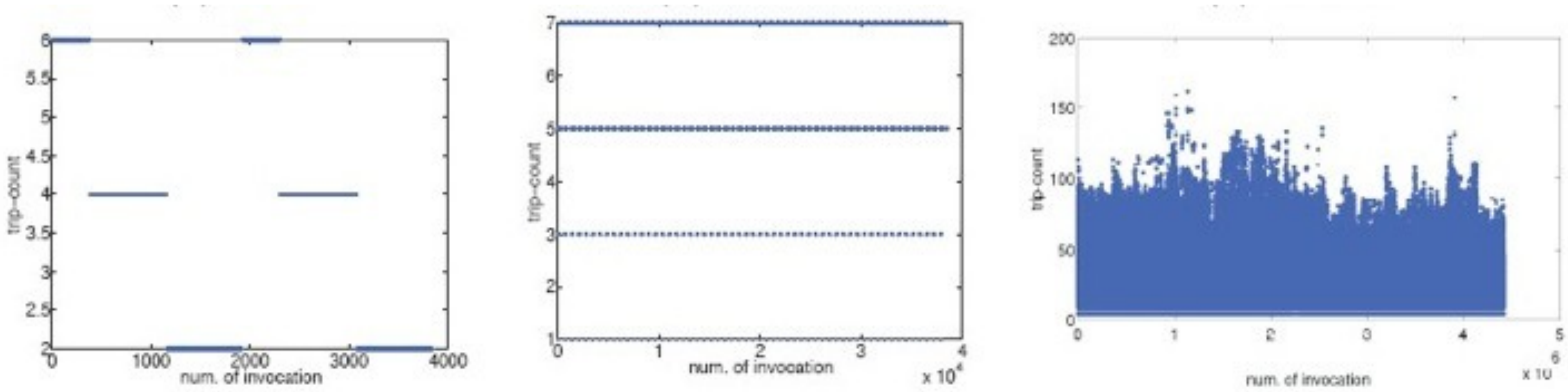
Sequence Prediction Framework

- Pattern Recognition



(a) C1.4 (b) C2.1.1 (c) C2.1.2 (d) C2.2.1

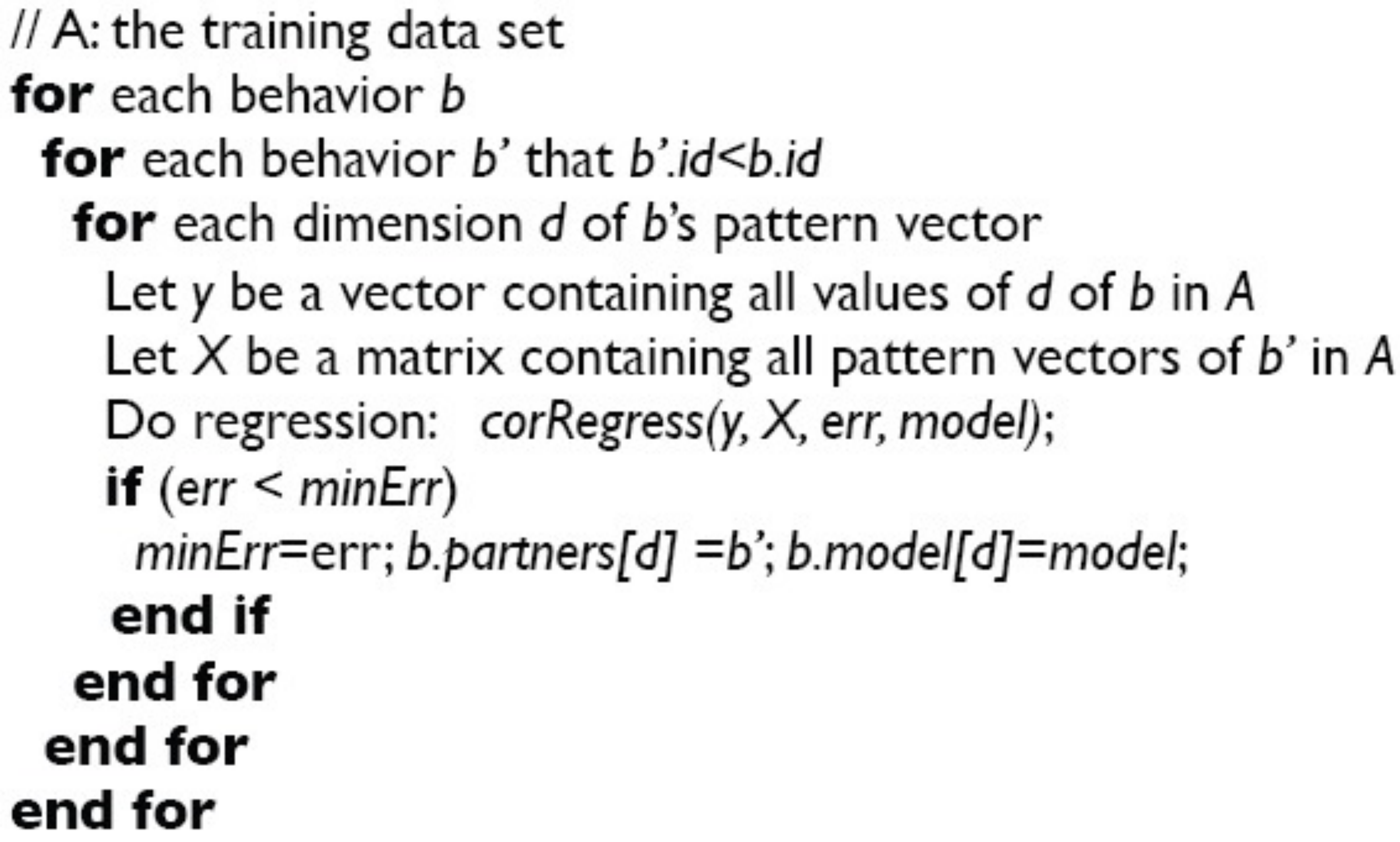
<c1.4,1,1,17> <c2.1.1,c1.4,10,31,-1,c1.1,18,23>



(e) C2.2.3 (f) C3.2 (g) C3.3

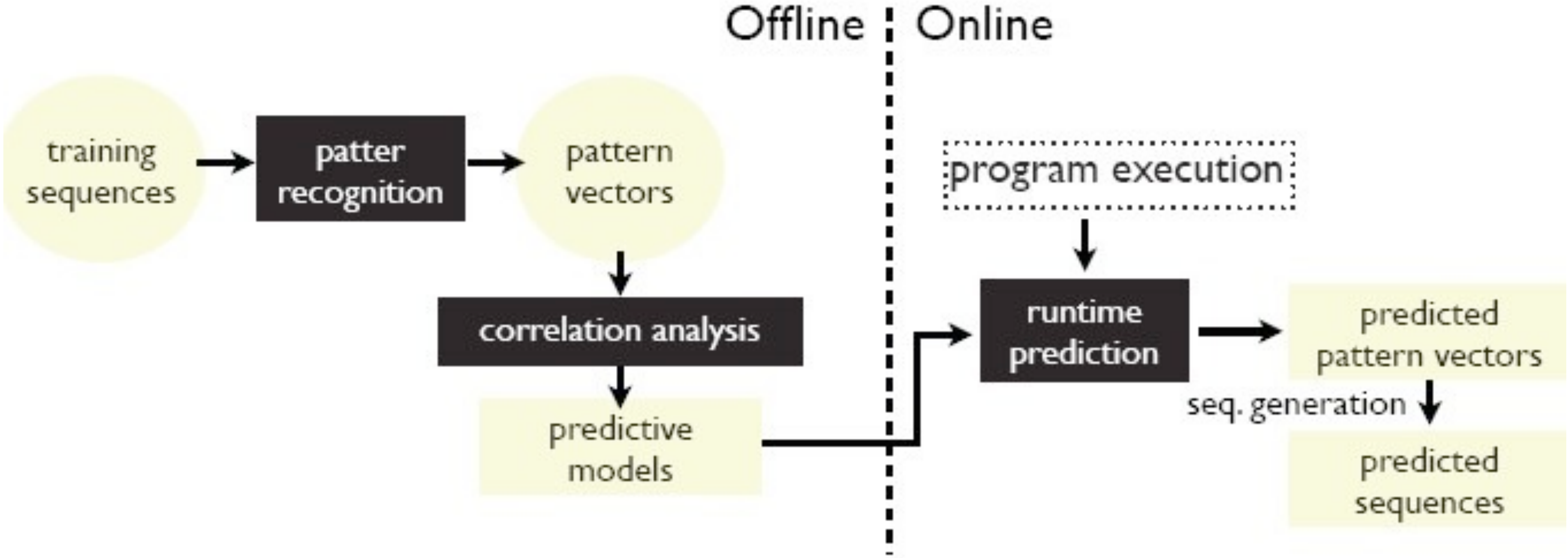
Sequence Prediction Framework

- Correlation Prediction

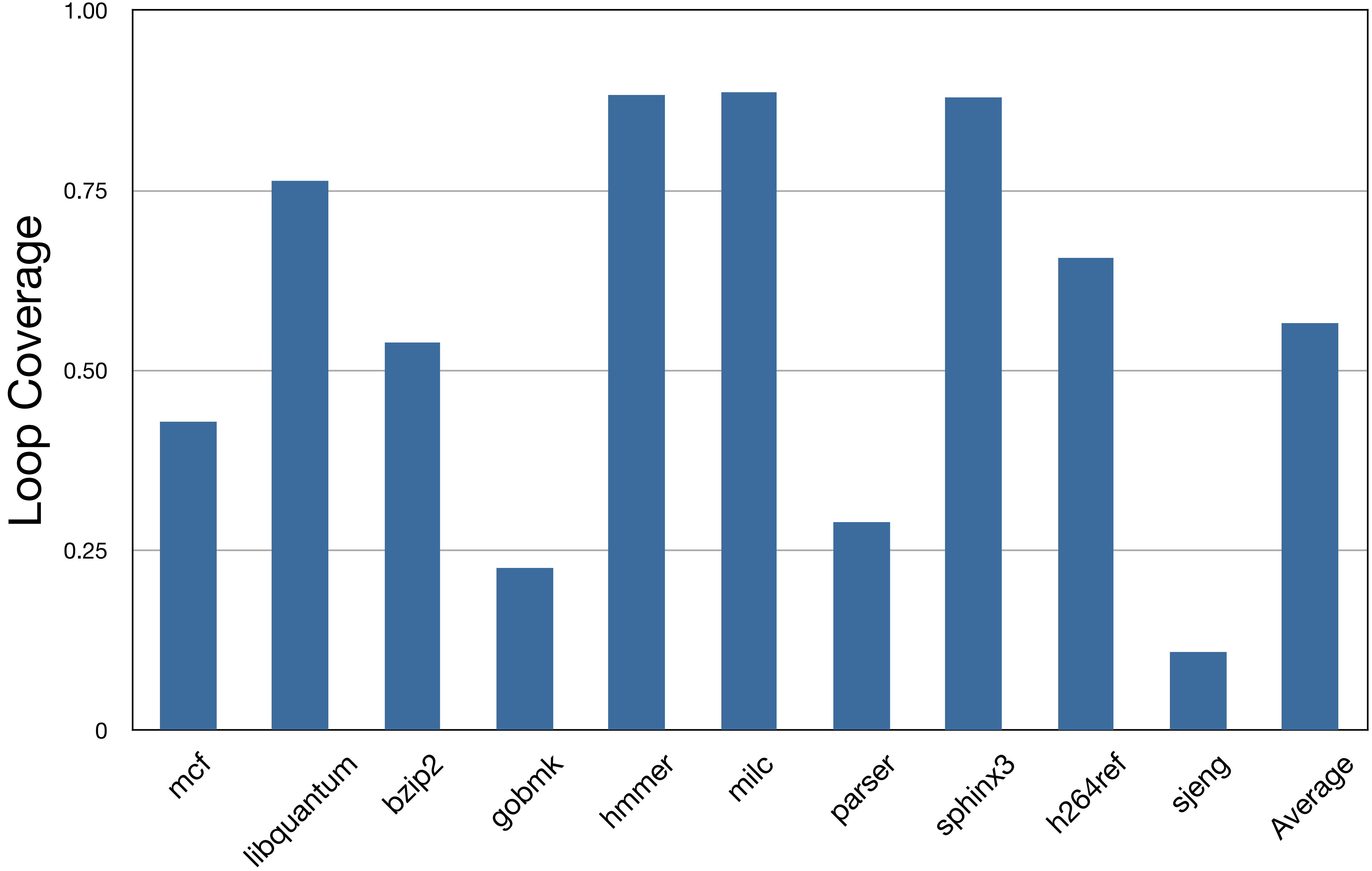


```
// A: the training data set
for each behavior b
  for each behavior b' that b'.id<b.id
    for each dimension d of b's pattern vector
      Let y be a vector containing all values of d of b in A
      Let X be a matrix containing all pattern vectors of b' in A
      Do regression:   corRegress(y, X, err, model);
      if (err < minErr)
        minErr=err; b.partners[d] =b'; b.model[d]=model;
      end if
    end for
  end for
end for
```

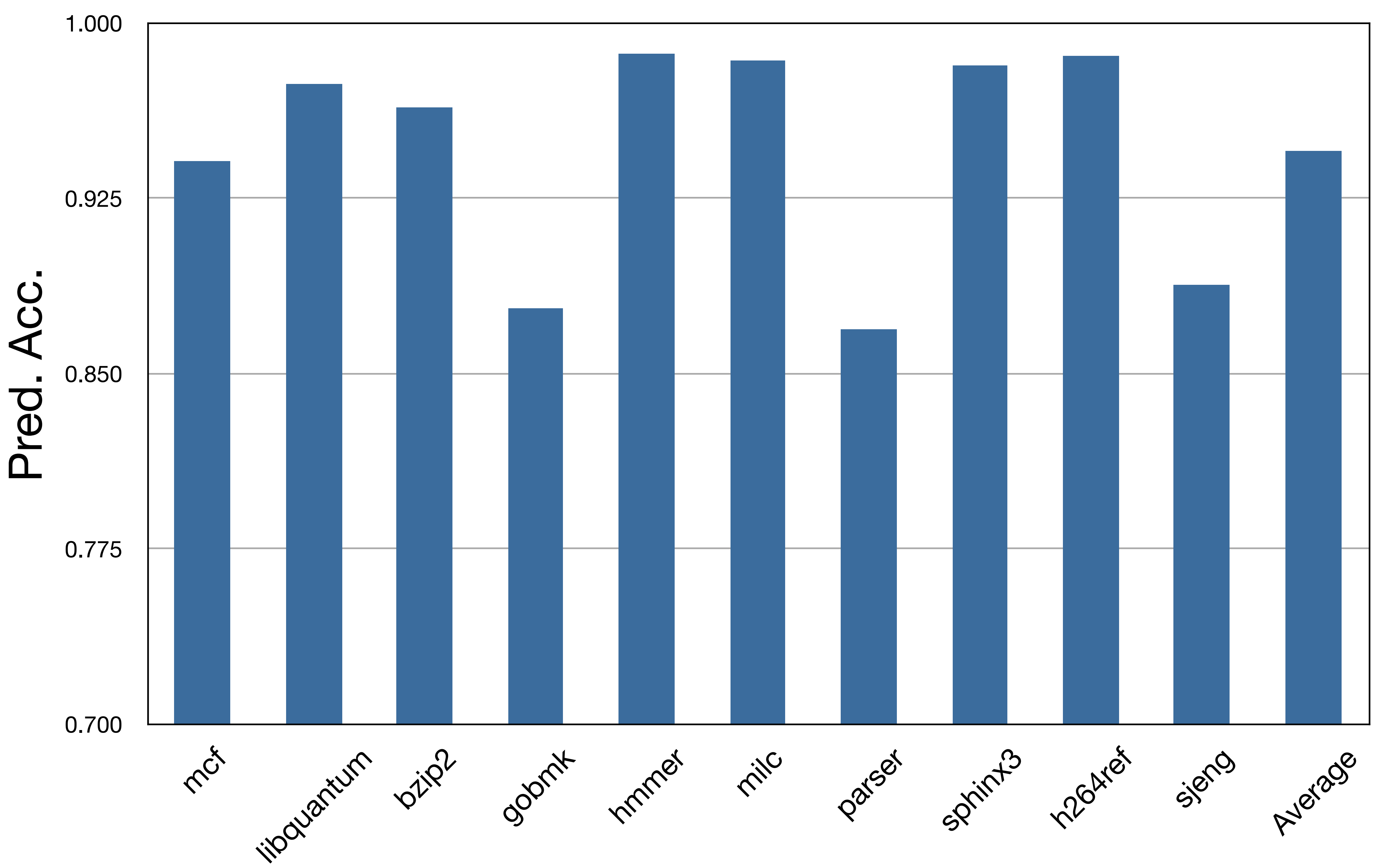
Sequence Prediction Framework



Results


Loop Coverage
1.00
0.75
0.50
0.25
0
mcf
libquantum
bzip2
gobmk
hmmer
milc
parser
sphinx3
h264ref
sjeng
Average

Results


Pred. Acc.
1.000
0.925
0.850
0.775
0.700
mcf
libquantum
bzip2
gobmk
hmmer
milc
parser
sphinx3
h264ref
sjeng
Average

Possible Uses

- Aggressive Optimizations
  - Loop unrolling for non-countable loops

```
While(!p) {
    if(satisfySomeCondition(p)) {
         result = p;
         break;
     }
     else
          p = p->next;
 }
```

  - Need runtime check and recovery support

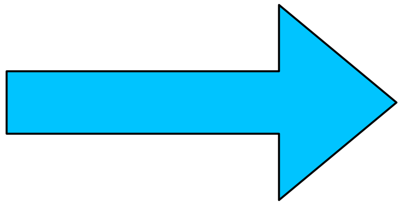
Possible Uses

- Loop parallelization

```
loop1 {

    loop2 {
         ...;
    }

}
```



loop interchange?

```
loop2 {

    loop1 {
         ...;
    }

}
```

Possible Uses

- From loop trip counts to other behaviors
  - Function hotness
  - Prefetching aggressiveness
  - Software pipelining
  - Trace selection in trace JIT

Summary

- Program behavior prediction is useful for many compiler optimizations, and even for OS and architecture level
- Behavior Sequences show extreme complexity, but correlation provides an opportunity to predict them
- Three requirements for useful predictions
- High prediction accuracy is possible for many loops

Thanks!

Bo Wu @ William & Mary