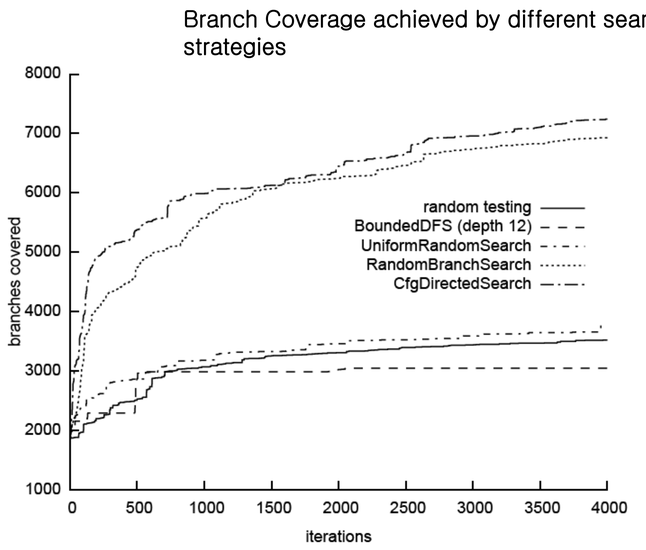
HUMAN GENERATE TEST INPUT-HUGETI

Youil Kim, Byeongjun Yu, Saransh Srivastava, Ludovic Patey
Seoul National University Programming Lab

1. problem

To employ a better and more efficient way than the Automatic Test Input Generator for Software Testing,keeping in mind budget,time and quality to do so.

Automated Test Generation Tool :
Crest



(c) Vim 5.7, with 20 characters as input. Contains 39166 branches, an estimated 23400 of which are reachable given our testing. 2-3 sec/iteration.

2. IDEA

Present Automatic Test Input Generators generates test inputs which can cover various paths.Unfortunately,they skip some paths in the program which reduces efficiency.So,this is our attempt to cover those paths using Human Computation.

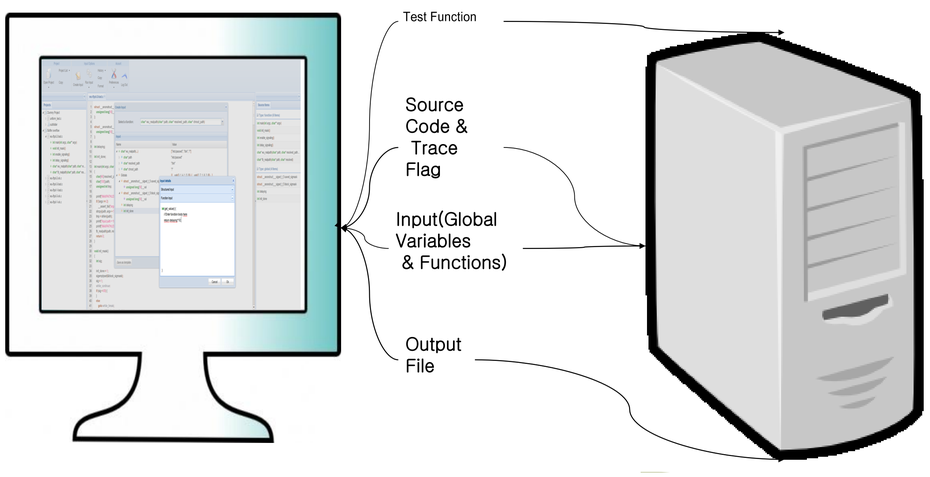
Human Computation :
The idea of using human effort to perform tasks that computers cannot perform efficiently, usually in disguise and in an enjoyable manner.

Scenario:

- webpage shows a software source.
- people submit test inputs for the software.
- if an input covers a different execution path from existing inputs, the one who submit the input gets some whatever incentives.
- the incentive grows as the coverage nears to 100%

3. SOLUTION

Overview



Select a particular C file(or a function file) from Projects.
Select 'Create Input' and enter random values for the function and global variables.
This information is transferred to the server.

Instrumentation

```
#include<stdio.h>
#include<stdlib.h>

int func_1(int x)
{

    int i;
    if(x > 100){
        i = -1;
        printf("%d\n",i);
    }else{
        i = x ;
        printf("%d\n",i);
    }
    return 0;
}
```

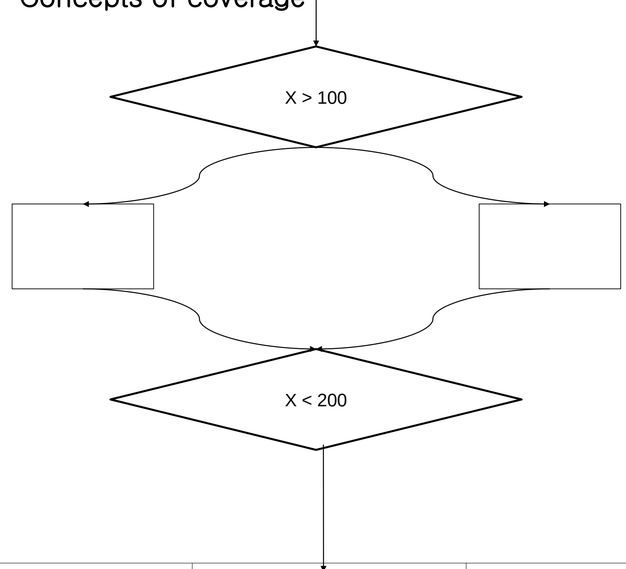
```
void __HugetiPathTracer(int id ) ;
#line 11 "func_1.c"
extern int ( /* missing proto */  printf)() ;
#line 5 "func_1.c"
int func_1(int x )
{ int i ;
  int __retres3 ;

  {
  __HugetiPathTracer(1);
#line 9
  if (x > 100) {
    __HugetiPathTracer(2);
#line 10
    i = -1;
#line 11
    printf("%d\n", i);
  } else {
    __HugetiPathTracer(3);
#line 13
    i = x;
#line 14
    printf("%d\n", i);
  }
  __HugetiPathTracer(4);
#line 16
  __retres3 = 0;
  {
  __HugetiPathTracer(5);
#line 5
  return (__retres3);
  }
}
}
```

To calculate branch coverage, we have to know which branches are covered.
So we use CIL(C Intermediate Language) to instrument some codes which can store which branches are covered.

Using this technique, we can also make instrumented code to get paths covered by specific input.

Concepts of coverage



	Branch Coverage	Path Coverage
1st input x == 150	50 %	25%
2nd input x == 300	75%	50%
3rd input x == 50	100%	75%

4.CONCLUSION

UI – Supports primitive data types (also some data structures) and functions as input.

Server – Instrument target code to calculate branch coverage.

The current HUGETI client is merely a web-site where users generate almost random inputs. Such a naive way of input generation possibly makes users boring. There should be some opportunities where human can solve the problems ‘easily’ and ‘joyfully’. Our goal is to let the users enjoy creating test inputs.

Also , we plan to implement data structure support on the server side.