Using Execution Paths to Evolve Software Patches

ThanhVu Nguyen*, Westley Weimer, Claires Le Gouges**, Stephanie Forrest***

* University of New Mexico
** University of Virginia

Tuesday, March 31, 2009

Introduction

- Software is THE problem

- Software Repair using Genetic Programming (SRGP)
  - Start with the original (buggy) program
  - Focus on execution path through AST
  - Restrict mutation and crossover to execution path
  - Don’t invent any new code

  - Results: successfully repair 13 **real** programs in over **140,000** lines of code in **feasible** time

SRGP Algorithm Outline

```
Preprocessing
Repeat
    Fitness Evaluation
        If found an individual C with accepted fitness, Return C
    Exploitation
        Select mating pool M consisting of high fit individuals
    Exploration
        Perform recombination operator on M to get a new generation N
        Apply mutation operator on N
Until termination criteria are met
```

Example: Zunebug

- Millions of Microsoft Zune media players mysteriously froze up on December 31st, 2008

- The bug: a date related function in Zune enters an *infinite loop* when the input is the last day of a leap year

Zunebug

```
void zunebug(int days) {
  int year = 1980;
  while (days > 365)  {
    if (isLeapYear(year)){
      if (days > 366)  {
        days -= 366;
        year += 1;
      }
      else{
      }
    }
    else {
      days -= 365;
      year += 1;
    }
  }
  printf(“year is %d\n”, year);
}
```

Zunebug

```
1.  void zunebug(int days) {
2.    int year = 1980;
3.    while (days > 365)  {
4.      if (isLeapYear(year)){
5.        if (days > 366)  {
6.          days -= 366;
7.          year += 1;
8.        }
9.        else{
10.       }
11.     }
12.     else {
13.       days -= 365;
14.       year += 1;
15.     }
16.   }
17.   printf("year is %d\n", year);
18. }
```

Input: 1000

Positive Exec Path

1 – 8, 11-18

Zunebug

```
1.  void zunebug(int days) {
2.    int year = 1980;
3.    while (days > 365)  {
4.      if (isLeapYear(year)){
5.        if (days > 366)  {
6.          days -= 366;
7.          year += 1;
8.        }
9.        else{
10.       }
11.     }
12.     else {
13.       days -= 365;
14.       year += 1;
15.     }
16.   }
17.   printf("year is %d\n", year);
18. }
```

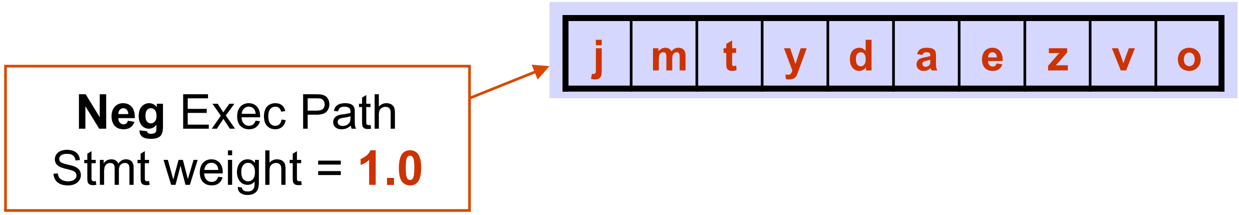
Input: 10593

Negative Exec Path

1 - 16

(3,4,8,11 infinitely repeating)

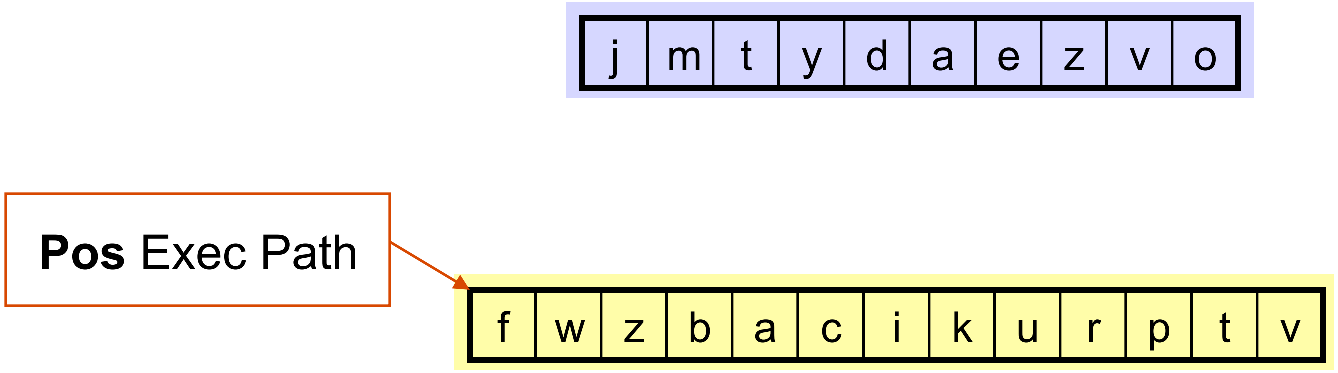
Weighted Execution Path



Neg Exec Path
Stmt weight = **1.0**

j	m	t	y	d	a	e	z	v	o

Weighted Execution Path



j	m	t	y	d	a	e	z	v	o

Pos Exec Path

f	w	z	b	a	c	i	k	u	r	p	t	v

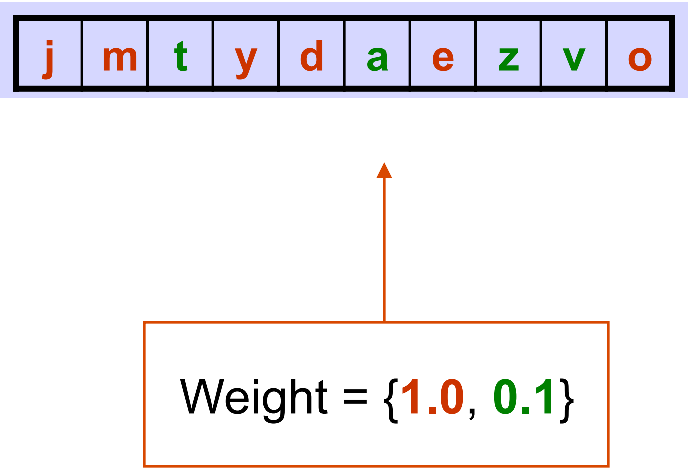
Weighted Execution Path

j	m	**t**	y	d	**a**	e	**z**	**v**	o

f	w	**z**	b	**a**	c	i	k	u	r	p	**t**	**v**

Stmts in both **Neg** Exec Path and **Pos** Exec Path

Weighted Execution Path



j	m	t	y	d	a	e	z	v	o

Weight = {1.0, 0.1}

Fitness Evaluation

- Take in a program source **P** to be evaluated

- Compile **P** to an executable program **P'**
  - If cannot compile, assign fitness **0**.

- Fitness score of **P'** : weighted sum of test cases that the **P'** passes

  Fitness(P') = # pos pass * W_pos + # neg pass * W_neg

  - 5 positive test cases (weight = 1), 1 or 2 negative test cases (weight = 10)
  - If **P'** passes all test cases, then **P** is a solution candidate
  - Note: the original (buggy) program passes all positive test cases

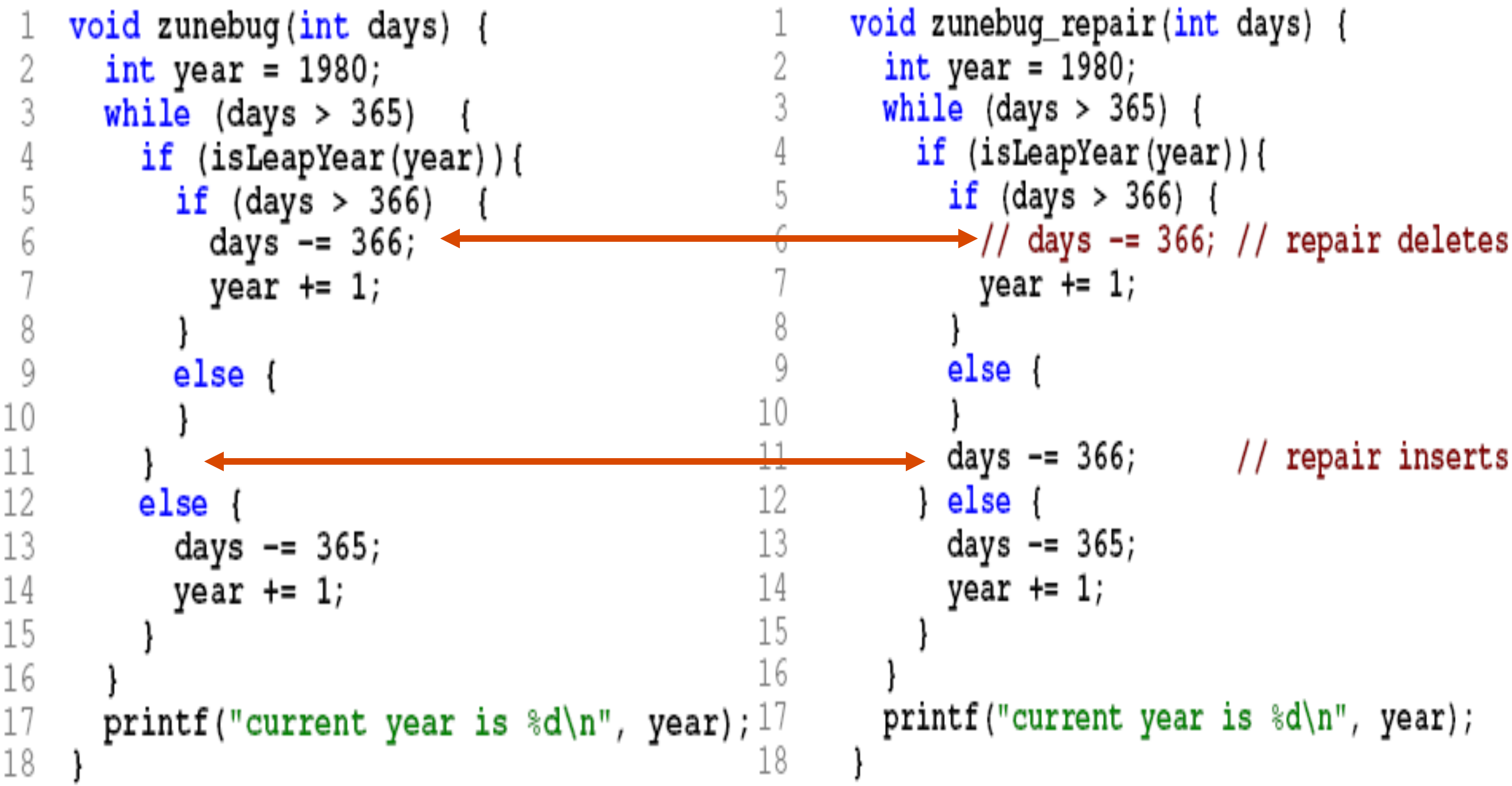
Genetic Operators

- **Recombination** (crossback)
  - Cross the input individuals back with the *original* program (instead of crossing over each other)

- **Mutation**
  - *delete(s).* s = {};
  - *insert(s,y).* s = {s; y;};
  - *swap(s,y).* t = s; s = {y}; y = {t};

Original Program

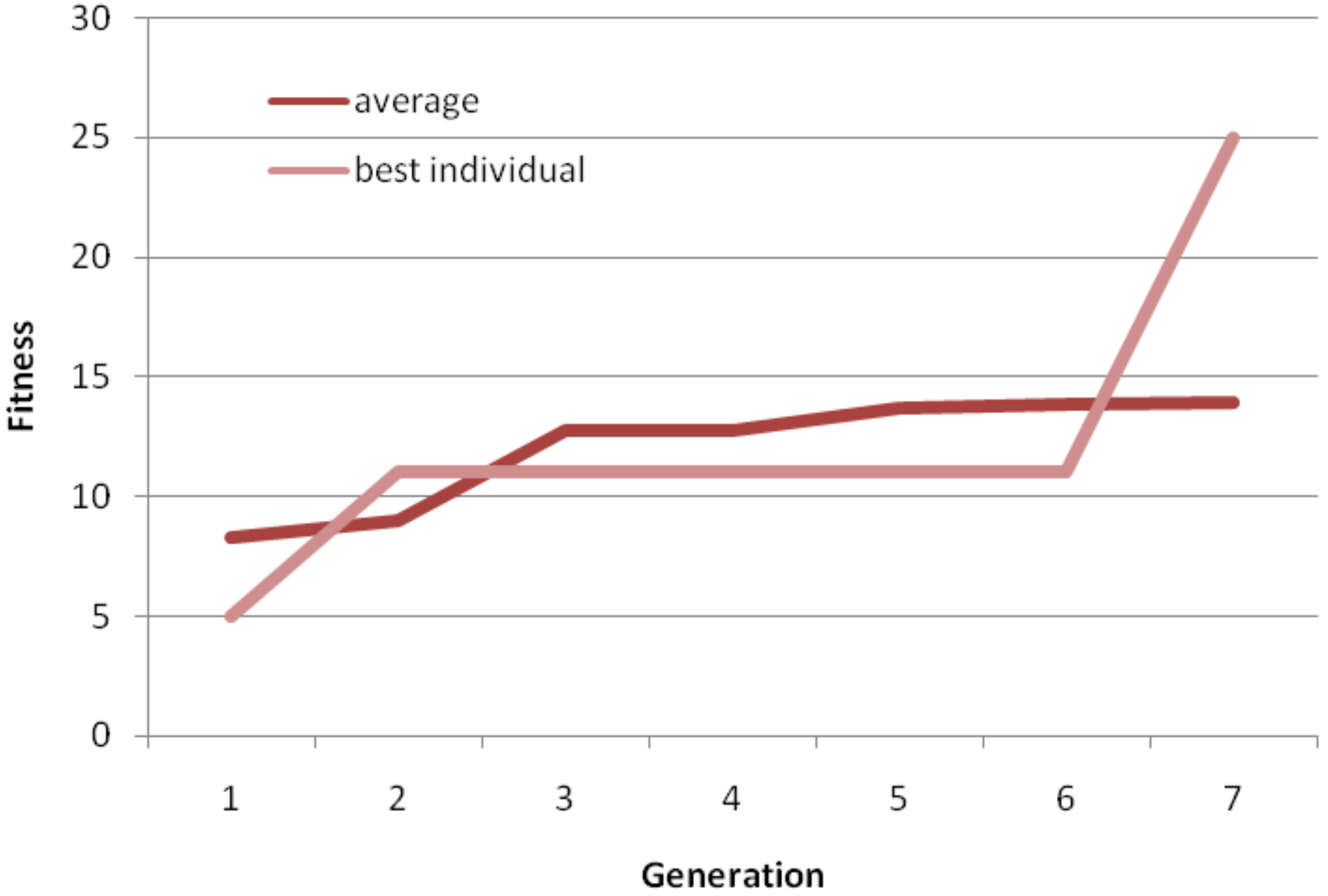
Final Repair



```c
void zunebug(int days) {
  int year = 1980;
  while (days > 365)  {
    if (isLeapYear(year)){
      if (days > 366)  {
        days -= 366;
        year += 1;
      }
      else {
      }
    }
    else {
      days -= 365;
      year += 1;
    }
  }
  printf("current year is %d\n", year);
}
```

```c
void zunebug_repair(int days) {
  int year = 1980;
  while (days > 365) {
    if (isLeapYear(year)){
      if (days > 366) {
        // days -= 366; // repair deletes
        year += 1;
      }
      else {
      }
      days -= 366;      // repair inserts
    } else {
      days -= 365;
      year += 1;
    }
  }
  printf("current year is %d\n", year);
}
```

Evolution of Zunebug



THE UNIVERSITY *of*
NEW MEXICO

Results

Program	Version	LoC	Stmts	Path Len	Program Description	Fault
gcd	_	22	10	1.3	Handcrafted example	Infinite loop
look-s	svr 4.0 1.1	1363	100	32.4	Dictionary lookup	Infinite loop
atris	1.0.6	21553	6470	34.0	Graphical Tetris game	Local stack buffer exploit
uniq	ultrix 4.3	1146	81	81.5	Duplicate text processing	Segfault
look-u	ultrix 4.3	1169	90	213.0	Dictionary lookup	Segfault
deroff	ultrix 4.3	2236	1604	251.4	Document processing	Segfault
nullhttpd	0.5.0	5575	1040	768.5	Web server	Remote heap buffer exploits
indent	1.9.1	9906	2022	1435.9	Source code processing	Infinite loop
units	svr4.0 1.1	1504	240	2159.7	Metric conversion	Segfault
flex	2.5.4a	18775	3635	3836.6	Lexical analyzer generator	Segfault

Results

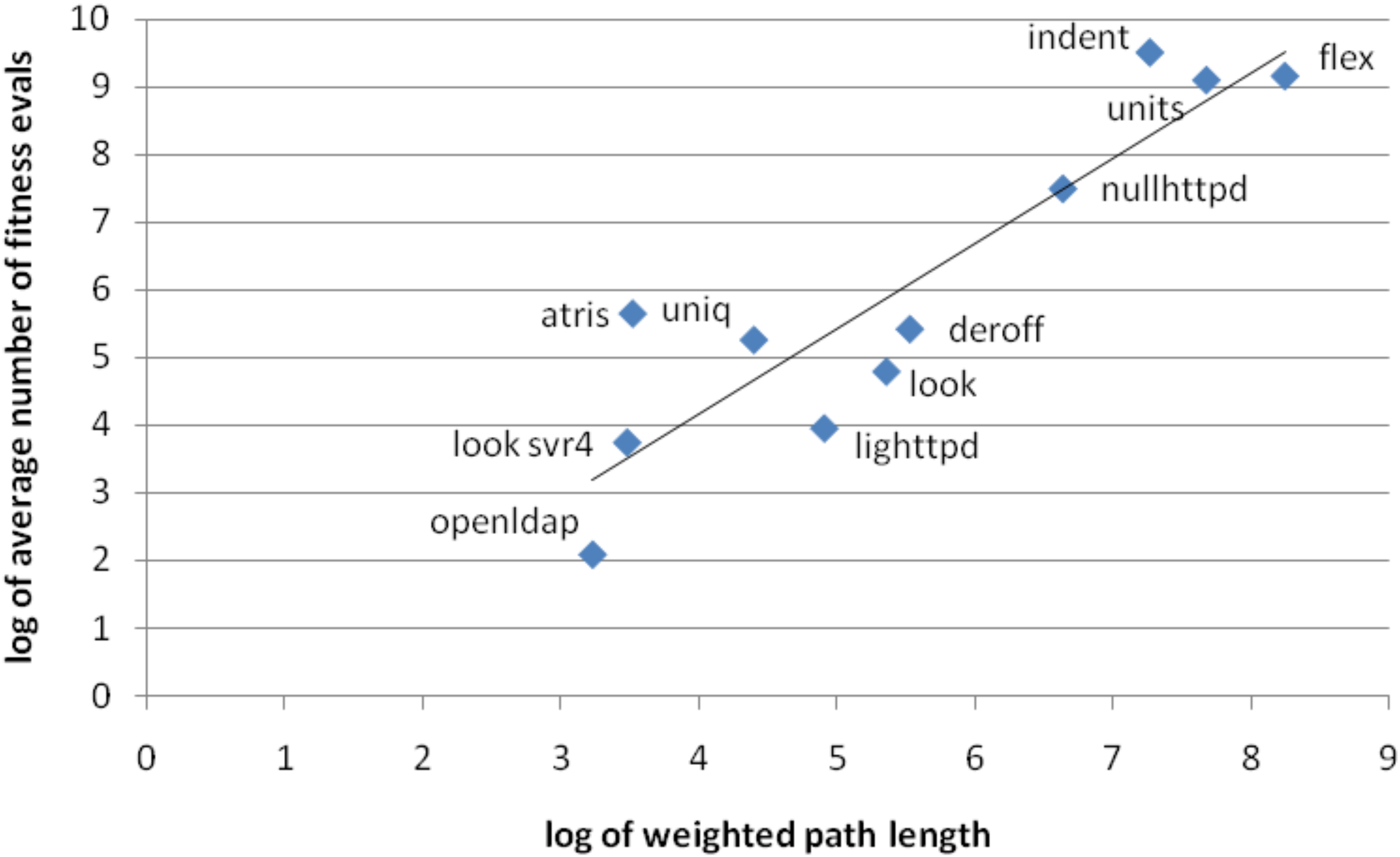
Program	LoC	Path Len	Repair			
			Fitness	Time	Success	Size
gcd	22	1.3	41.0	149 s	54 %	21
look-s	1363	32.4	8.5	51 s	100 %	21
atris	21553	34.0	13.2	69 s	82 %	19
uniq	1146	81.5	9.5	32 s	100 %	24
look-u	1169	213.0	11.1	42 s	99 %	24
deroff	2236	251.4	21.6	129 s	97 %	61
nullhttpd	5575	768.5	79.1	502 s	36 %	71
indent	9906	1435.9	95.6	533 s	7 %	221
units	1504	2159.7	55.7	107 s	7 %	23
flex	18775	3836.6	33.4	233 s	5 %	52

Most Recent Results

Program	Version	LoC	Stmts	Path Len	Program Description	Fault
OpenLDAP io.c	2.3.41	6519	25	1.3	Directory Protocol	Non-overflow denial of service
Php string.c	5.2.1	26044	52	34.0	Scripting Language	Integer overflow
Lighttpd fastcgi.c	1.4.17	13984	136	32.4	Web server	Remote heap buffer overflow
Wu-ftpd	2.6.0	35109	149	81.5	Ftp server	Format string

- Traditional 1-point ***crossover***
  - Works better than ***crossback*** in some programs and worse in others

Scalability



THE UNIVERSITY of NEW MEXICO

Conclusion

- **SRGP**
  - Focus on execution path to reduce search space complexity
  - Use GP to evolve code
  - Combine Positive and Negative test cases for fitness evaluation
    - Positive test cases: preserve the core functionality of the program
    - Negative test cases: identify the bug
  - Work on real world applications and different types of bugs
- **Future Work**
  - Explore different GP techniques
  - Integrate anomaly detection methods