

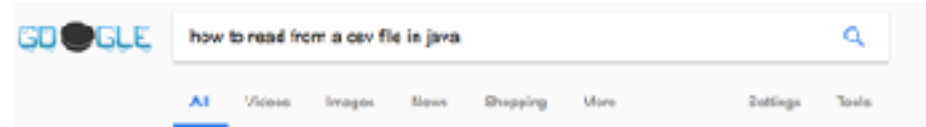
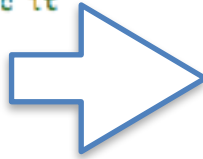


Program Splicing

Yanxin Lu, Swarat Chaudhuri,
Christopher Jermaine, David Melski
Presented by Yanxin Lu

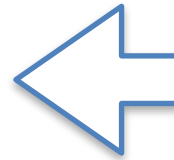
Copying and Pasting

```
/**  
 * Read a matrix from a csv file and square it  
 */  
int[][] csvmat(String filename) {  
    int[][] mat = new int[N][N];  
}
```



How to read and parse CSV file in Java - Mkyong
<https://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/>
Apr 26, 2013 · FileReader; import java.io.IOException; public class CSVReader { public static void
main(String[] args) { String csvFile = "7Users/mkyong/csv/country.csv"; BufferedReader br = null; String
line = ""; String csvSplitBy = ","; try { br = new BufferedReader(new FileReader(csvFile)); while ((line =
br.readLine()) ...

```
int[][] mat = new int[N][N];  
String fileName = "read_ex.csv";  
File file = new File(fileName);  
  
// this gives you a 2-dimensional array of strings  
List<List<String>> lines = new ArrayList<>();  
Scanner inputStream;  
  
try {  
    inputStream = new Scanner(file);  
  
    while (inputStream.hasNextLine()) {  
        String line = inputStream.nextLine();  
        String[] values = line.split(",");  
        // this adds the currently parsed line to the 2-dimensional string array  
        lines.add(Arrays.asList(values));  
    }  
  
    inputStream.close();  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
}  
}
```



```
jav: All right using the Apple CSV library as mentioned by @Mkyong, it's perfectly fine, I try to provide  
http: a solution that is closer to your code and maybe easier for you to follow:  
File: Scanner inputStream;  
try {  
    inputStream = new Scanner(file);  
  
    while (inputStream.hasNextLine()) {  
        String line = inputStream.nextLine();  
        String[] values = line.split(",");  
        // this adds the currently parsed line to the 2-dimensional string  
        List<List<String>> lines = new ArrayList<>();  
        lines.add(Arrays.asList(values));  
    }  
  
    inputStream.close();  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
}  
}
```

Time consuming and bugs introduced

Program Synthesis

- Automatically generating programs
- Specification
 - Logic formula
 - $\text{index} \geq 0$ and $\text{array}[\text{index}] = x$
 - Unit testing
 - $\text{bsearch}([], 1) = -1$, $\text{bsearch}([1, 2], 1) = 0$, ...
 - Natural language
 - Give me a binary search!
- Correctness

Problem

Can we use program synthesis to help the process of copying and pasting?

Related work

- Sketch (FMCAD 2013)
 - Cannot synthesize statements
 - Does not use a code database
- Code Transplantation (ISSTA 2015)
 - Not efficient
 - Does not search for relevant code snippets

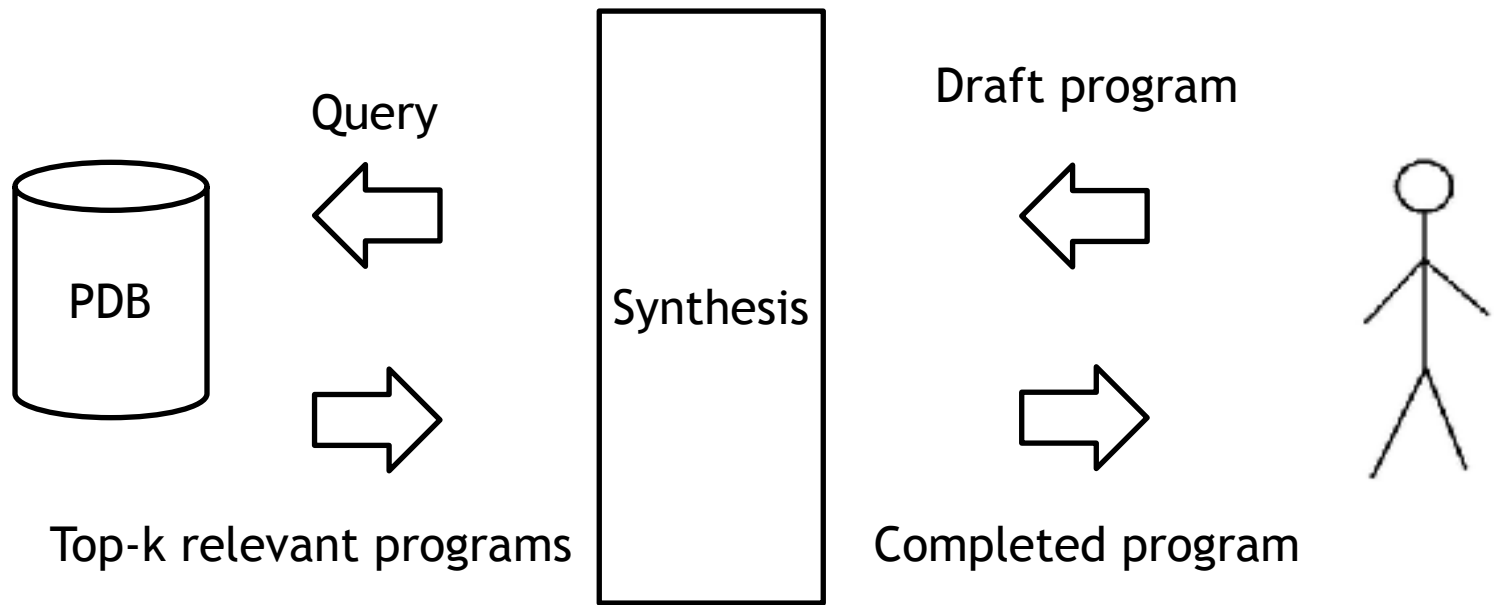
Program Splicing

- Use a large corpus of over 3.5 million programs
- Automate the process of copying and pasting
- Ensure correctness

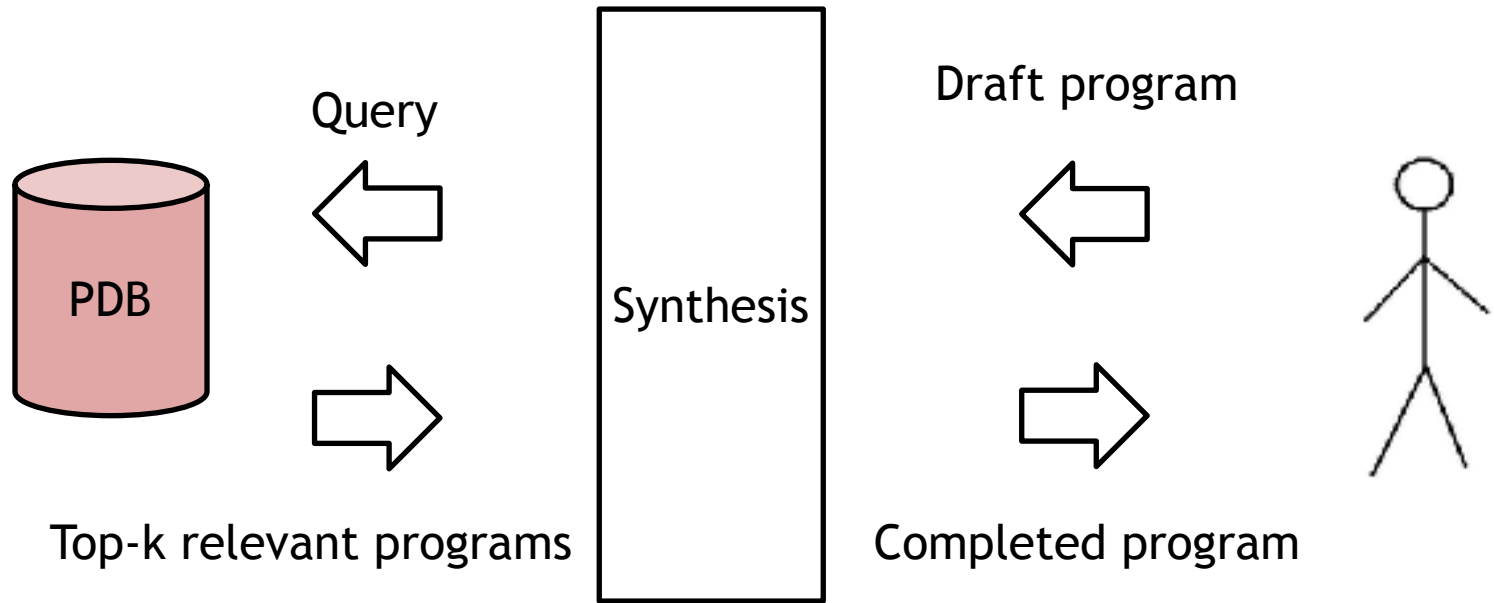
Summary

- Architecture
 - Corpus and Pliny database
 - Synthesis algorithm
- Experiment
- Conclusion

Architecture



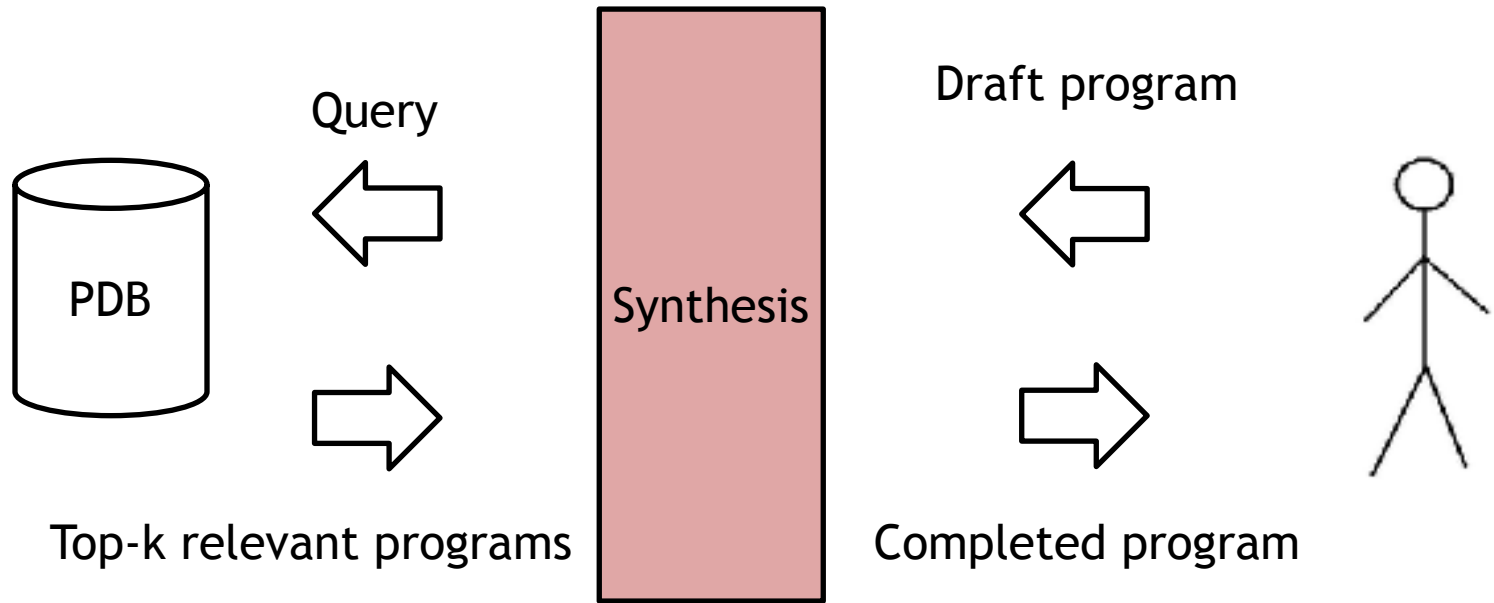
Architecture



PDB

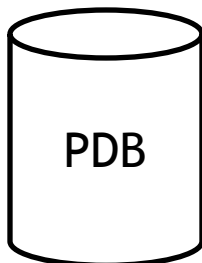
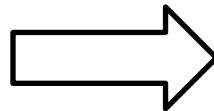
- 3.5 million Java programs with features
 - Github, SourceForge
 - Natural language terms
 - "read": 0.10976, "matrix": 0.65858, ...
- Similarity metrics
- Fast top-k query
 - 1-2 orders of magnitude faster than no-SQL database systems

Architecture



Relevant programs

```
int [][] csvmat(String filename) {  
    int[][] mat = new int[N][N];  
  
    /* COMMENT:  
    * Read a matrix from a csv file  
    *  
    * REQ:  
    * String filename = "matrix.csv";  
    * int[][] m = new int[N][N];  
    * __solution__  
    * return test_matrix(m);  
    */  
    ??  
}
```



```
List<String[]> read_csv(String infile) {  
    List<String[]> rowList = new ArrayList<String[]>();  
    try {DufferedReader br = new DufferedReader(new FileReader(infile));  
        String line;  
        while ((line = br.readLine()) != null) {  
            String[] lineItems = line.split(",");  
            rowList.add(lineItems);  
        }  
        br.close();  
    }  
    return rowList;  
}
```

```
int[][] read_csv(int[][] m, int r, int c, String filename) {  
    File f = new File(filename);  
    Scanner scanner = new Scanner(f);  
    for(int i=0; i<r; ++i){  
        String line = scanner.nextLine();  
        String[] fields = line.split(",");  
        for(int j = 0; j < c; ++j) {  
            m[i][j] = parseInt(fields[j]);  
        }  
    }  
    return m;  
}
```

```
public void main(String arg[]) {  
    String file = arg[0];  
    int row = Integer.parseInt(arg[1]);  
    int col = Integer.parseInt(arg[2]);  
    int[][] test = new int[row][col];  
    String delimiter = ",";  
    Scanner sc = new Scanner(file);  
    for(int i = 0; i < row; ++i) {  
        String line = sc.nextLine();  
        String[] testStr = line.split(delimiter);  
        for(int j = 0; j < col; ++j) {  
            test[i][j] = Integer.parseInt(testStr[j]);  
        }  
    }  
    return test;  
}
```

Filling in the holes

- Enumerative search

```
int [][] csvmat(String filename) {  
    int[][] mat = new int[N][N];  
  
    /* COMMENT:  
    * Read a matrix from a csv file  
    *  
    * REQ:  
    * String filename = "matrix.csv";  
    * int[][] m = new int[N][N];  
    * __solution__  
    * return test_matrix(m);  
    */  
    ??  
}
```

```
public void main(String arg[] ) {  
    String file = arg[0];  
    int row = Integer.parseInt(arg[1]);  
    int col = Integer.parseInt(arg[2]);  
    int[][] test = new int[row][col];  
    String delimiter = ",";  
    Scanner sc = new Scanner(file);  
    for(int i = 0; i < row; ++i) {  
        String line = sc.nextLine();  
        String[] testStr = line.split(delimiter);  
        for(int j = 0; j < col; ++j) {  
            test[i][j] = Integer.parseInt(testStr[j]);  
        }  
    }  
    return test;  
}
```

Filling in the holes

- Enumerative search

```
int [][] csvmat(String filename) {  
    int[][] mat = new int[N][N];  
  
    /* COMMENT:  
     * Read a matrix from a csv file  
     *  
     * REQ:  
     * String filename = "matrix.csv";  
     * int[][] m = new int[N][N];  
     * __solution__  
     * return test_matrix(m);  
     */  
    ??  
}
```

```
public void main(String arg[]) {  
    String file = arg[0];  
    int row = Integer.parseInt(arg[1]);  
    int col = Integer.parseInt(arg[2]);  
    int[][] test = new int[row][col];  
    String delimiter = ",";  
    Scanner sc = new Scanner(file);  
    for(int i = 0; i < row; ++i) {  
        String line = sc.nextLine();  
        String[] testStr = line.split(delimiter);  
        for(int j = 0; j < col; ++j) {  
            test[i][j] = Integer.parseInt(testStr[j]);  
        }  
    }  
  
    return test;  
}
```

Filling in the holes

- Enumerative search

```
int [][] csvmat(String filename) {  
    int[][] mat = new int[N][N];  
  
    /* COMMENT:  
    * Read a matrix from a csv file  
    *  
    * REQ:  
    * String filename = "matrix.csv";  
    * int[][] m = new int[N][N];  
    * __solution__  
    * return test_matrix(m);  
    */  
    ??  
}
```

```
public void main(String arg[]) {  
    String file = arg[0];  
    int row = Integer.parseInt(arg[1]);  
    int col = Integer.parseInt(arg[2]);  
    int[][] test = new int[row][col];  
    String delimiter = ",";  
    Scanner sc = new Scanner(file);  
    for(int i = 0; i < row; ++i) {  
        String line = sc.nextLine();  
        String[] testStr = line.split(delimiter);  
        for(int j = 0; j < col; ++j) {  
            test[i][j] = Integer.parseInt(testStr[j]);  
        }  
    }  
    return test;  
}
```


Variable Renaming

- Undefined variables

```
int [][] csvmat(String filename) {  
    int [][] mat = new int[N][N];
```

```
/**  
 * ...  
 */
```

```
String delimiter = ",";  
Scanner sc = new Scanner(File);  
for(int i = 0; i < row; ++i) {  
    String line = sc.nextLine();  
    String[] testStr = line.split(delimiter);  
    for(int j = 0; j < col; ++j) {  
        test[i][j] = Integer.parseInt(testStr[j]);  
    }  
}
```

```
}
```

```
int [][] csvmat(String filename) {  
    int [][] mat = new int[N][N];
```

```
/**  
 * ...  
 */
```

```
String delimiter = ",";  
Scanner sc = new Scanner(File);  
for(int i = 0; i < row; ++i) {  
    String line = sc.nextLine();  
    String[] testStr = line.split(delimiter);  
    for(int j = 0; j < col; ++j) {  
        test[i][j] = Integer.parseInt(testStr[j]);  
    }  
}
```

```
}
```

Testing

- Filtering out incorrect programs

```
/* COMMENT:
 * Read a matrix from a csv file
 *
 * REQ:
 * String filename = "matrix.csv";
 * int[][] mat = new int[N][N];
 * __solution__
 * return test_matrix(mat);
 */
String file = arg[0];
int row = Integer.parseInt(arg[1]);
int col = Integer.parseInt(arg[2]);
int[][] test = new int[row][col];
String delimiter = ",";
Scanner sc = new Scanner(file);
for(int i = 0; i < row; ++i) {
    String line = sc.nextLine();
    String[] testStr = line.split(delimiter);
    for(int j = 0; j < col; ++j) {
        test[i][j] = Integer.parseInt(testStr[j]);
    }
}
```

```
/* COMMENT:
 * Read a matrix from a csv file
 *
 * REQ:
 * String filename = "matrix.csv";
 * int[][] mat = new int[N][N];
 * __solution__
 * return test_matrix(mat);
 */
String file = arg[0];
int row = Integer.parseInt(arg[1]);
int col = Integer.parseInt(arg[2]);
```

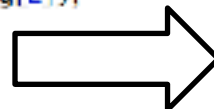
```
int [][] csvmat(String filename) {
    int[][] mat = new int[N][N];
```

```
/* COMMENT:
 * Read a matrix from a csv file
 *
 * REQ:
 * String filename = "matrix.csv";
 * int[][] mat = new int[N][N];
 * __solution__
 * return test_matrix(mat);
 */
```

```
String delimiter = ",";
Scanner sc = new Scanner(filename);
for(int i = 0; i < N; ++i) {
    String line = sc.nextLine();
    String[] testStr = line.split(delimiter);
    for(int j = 0; j < N; ++j) {
        mat[i][j] = Integer.parseInt(testStr[j]);
    }
}
```

```
/* COMMENT:
 * Read a matrix from a csv file
 *
 * REQ:
 * String filename = "matrix.csv";
 * int[][] mat = new int[N][N];
 * __solution__
 * return test_matrix(mat);
 */
String delimiter = ",";
Scanner sc = new Scanner(filename);
for(int i = 0; i < N; ++i) {
    String line = sc.nextLine();
    String[] testStr = line.split(delimiter);
    for(int j = 0; j < N; ++j) {
        mat[i][j] = Integer.parseInt(testStr[j]);
    }
}
```

```
/* COMMENT:
 * Read a matrix from a csv file
 *
 * REQ:
 * String filename = "matrix.csv";
 * int[][] mat = new int[N][N];
 * __solution__
 * return test_matrix(mat);
 */
for(int i = 0; i < row; ++i) {
    String line = sc.nextLine();
    String[] testStr = line.split(delimiter);
    for(int j = 0; j < col; ++j) {
        mat[i][j] = Integer.parseInt(testStr[j]);
    }
}
```



Benchmark

Benchmarks	Synthesis Time (s)	LOC	Var	Holes (expr-stmt)	Test	μ Scalpel
Sieve Prime	4.6	12-17	2	2-1	3	162.1
Collision Detection	4.2	10-15	2	2-1	4	N/A
Collecting Files	3.0	13-25	2	1-1	2	timeout
Binary Search	15.4	12-20	5	1-1	3	timeout
HTTP Server	41.1	24-45	6	1-2	2	N/A
Prim's Distance Update	61.1	53-58	11	1-1	4	timeout
Quick Sort	77.2	11-18	6	1-1	1	timeout
CSV	88.4	13-23	4	1-2	2	timeout
Matrix Multiplication	108.9	13-15	8	1-1	1	timeout
Floyd Warshall	110.4	9-12	7	1-1	7	timeout
HTML Parsing	140.4	20-34	5	1-2	2	N/A
LCS	161.5	29-36	10	0-1	1	timeout

Benchmark

Benchmarks	Synthesis Time (s)	LOC	Var	Holes (expr-stmt)	Test	μ Scalpel
Sieve Prime	4.6	12-17	2	2-1	3	162.1
Collision Detection	4.2	10-15	2	2-1	4	N/A
Collecting Files	3.0	13-25	2	1-1	2	timeout
Binary Search	15.4	12-20	5	1-1	3	timeout
HTTP Server	41.1	24-45	6	1-2	2	N/A
Prim's Distance Update	61.1	53-58	11	1-1	4	timeout
Quick Sort	77.2	11-18	6	1-1	1	timeout
CSV	88.4	13-23	4	1-2	2	timeout
Matrix Multiplication	108.9	13-15	8	1-1	1	timeout
Floyd Warshall	110.4	9-12	7	1-1	7	timeout
HTML Parsing	140.4	20-34	5	1-2	2	N/A
LCS	161.5	29-36	10	0-1	1	timeout

Synthesis algorithm is efficient.

Benchmark

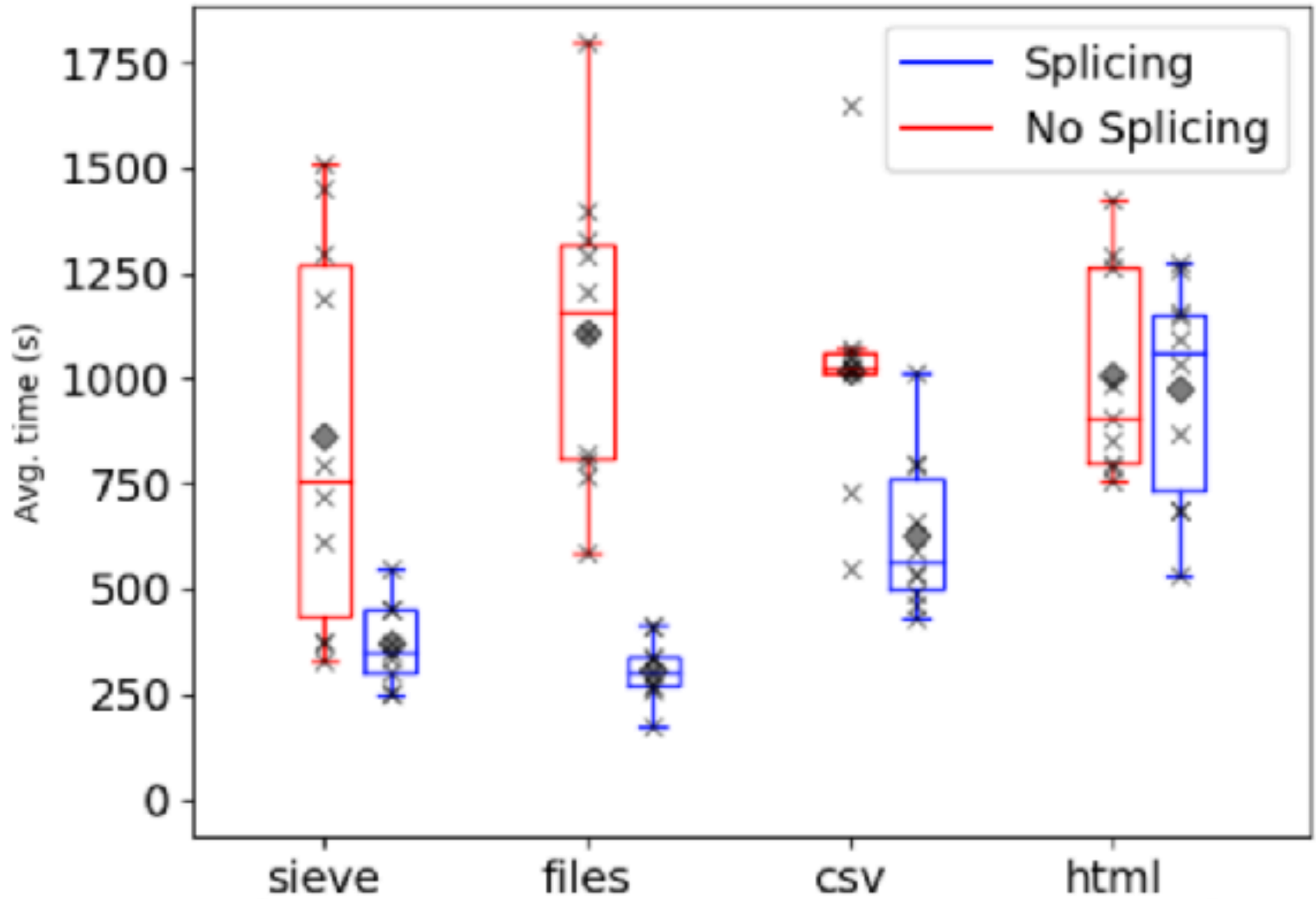
Benchmarks	Synthesis Time (s)	LOC	Var	Holes (expr-stmt)	Test	μ Scalpel
Sieve Prime	4.6	12-17	2	2-1	3	162.1
Collision Detection	4.2	10-15	2	2-1	4	N/A
Collecting Files	3.0	13-25	2	1-1	2	timeout
Binary Search	15.4	12-20	5	1-1	3	timeout
HTTP Server	41.1	24-45	6	1-2	2	N/A
Prim's Distance Update	61.1	53-58	11	1-1	4	timeout
Quick Sort	77.2	11-18	6	1-1	1	timeout
CSV	88.4	13-23	4	1-2	2	timeout
Matrix Multiplication	108.9	13-15	8	1-1	1	timeout
Floyd Warshall	110.4	9-12	7	1-1	7	timeout
HTML Parsing	140.4	20-34	5	1-2	2	N/A
LCS	161.5	29-36	10	0-1	1	timeout

No need to write many tests.

User study

- Evaluate whether program splicing can help human developers
 - 12 graduate students and six professionals
 - Web-based programming environment
 - Four programming problems
 - Two were done using our system
 - Two were done without using our system
 - Internet search encouraged

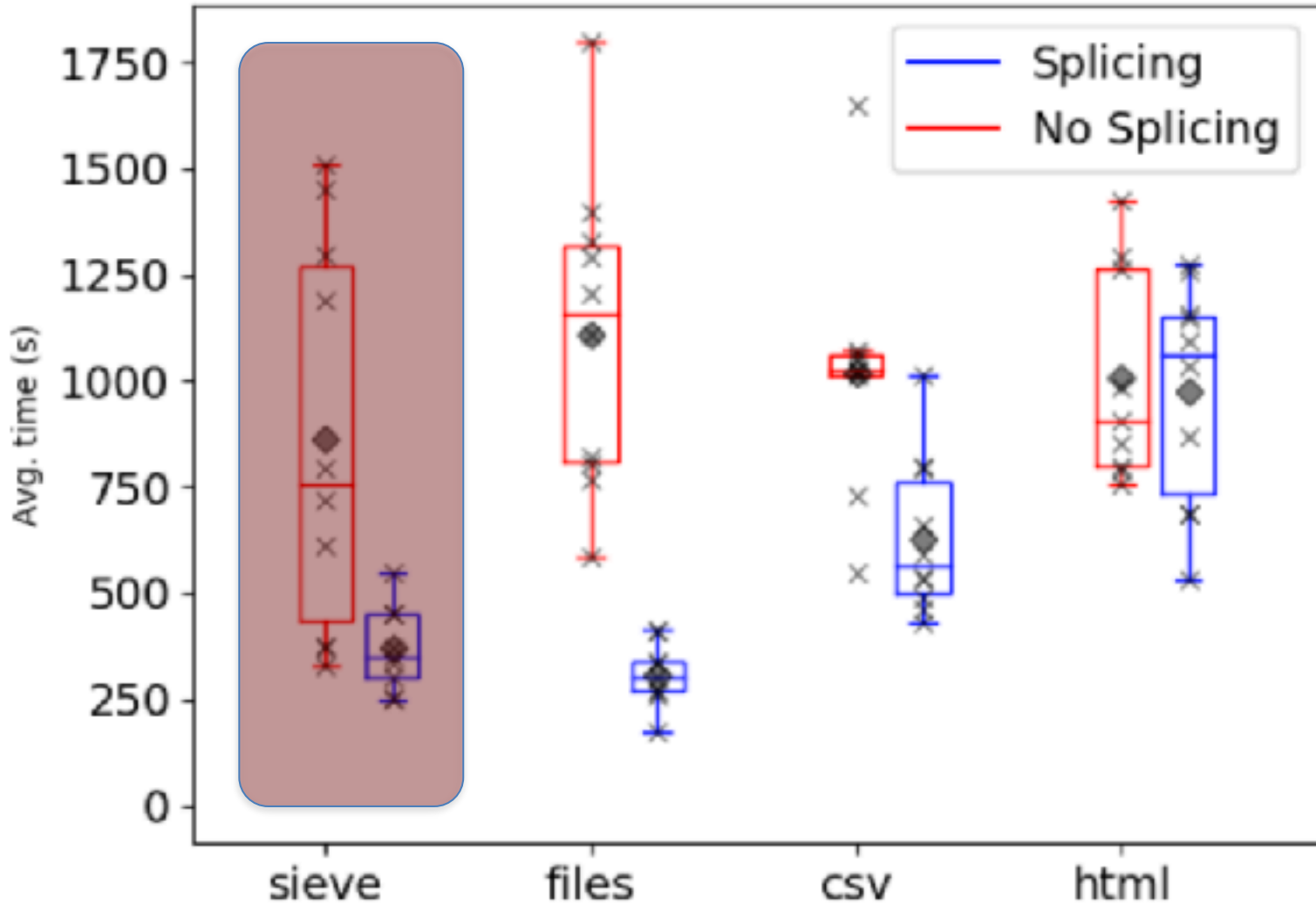
User study



Algorithmic programming

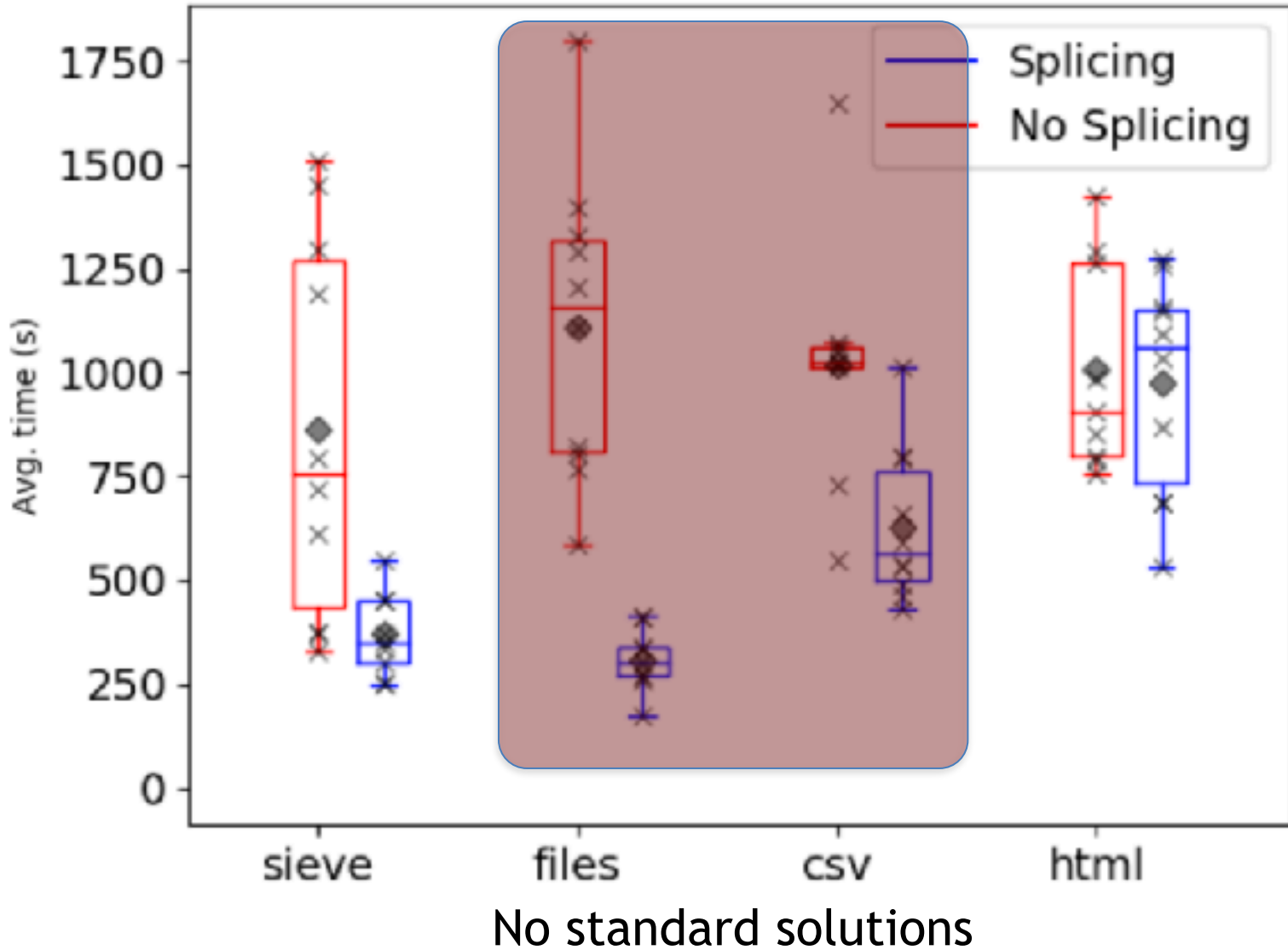
System programming

User study

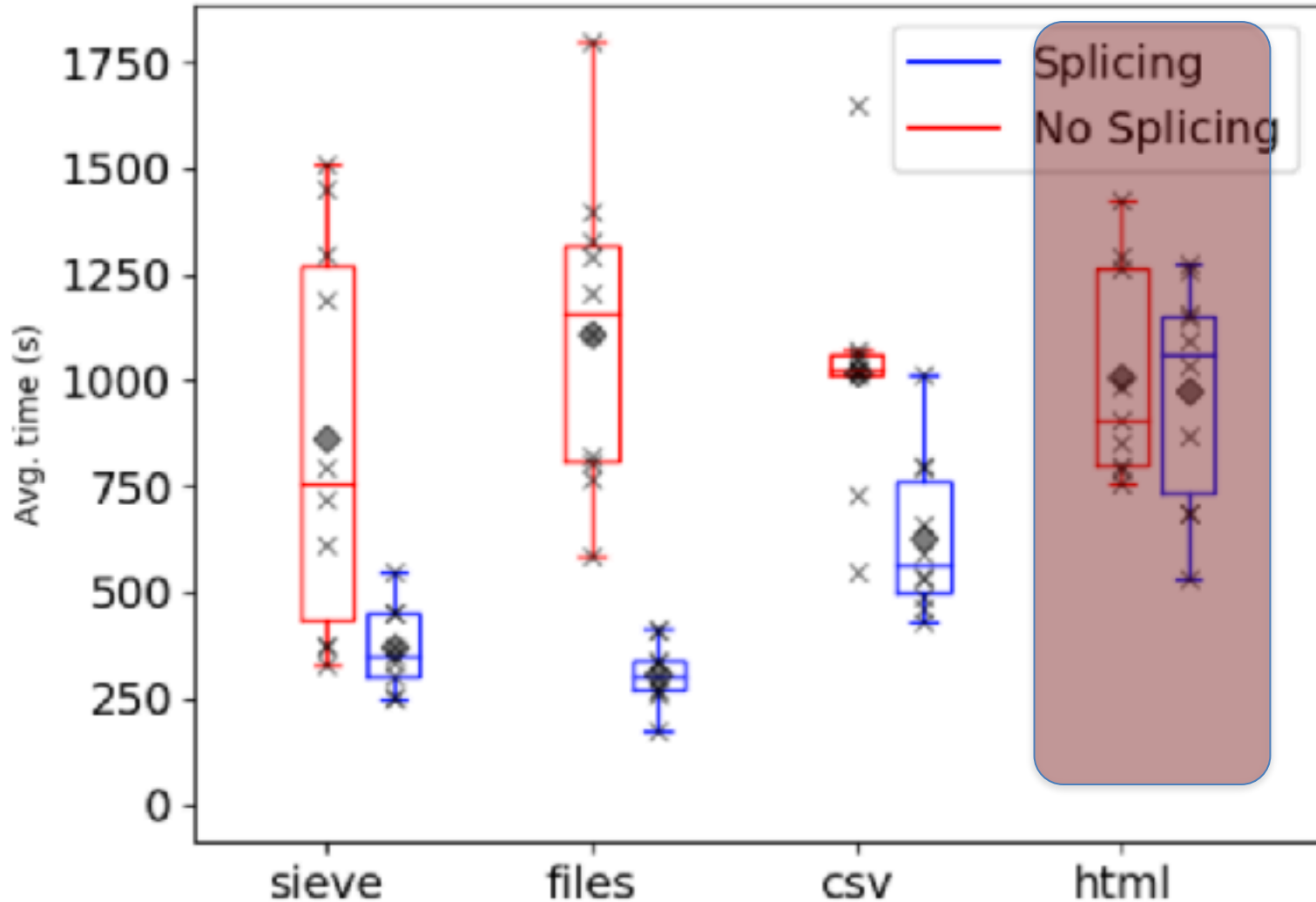


Appears to be simple, but it was not

User study



User study



Good documentations and tests were hard to write.

Conclusion

- Data-driven program synthesis
 - A large code corpus
 - Enumerative search
- User study
 - Good for tasks that do not have standard solutions



Heuristics

- A set of heuristics
 - Types
 - Context
- Huge search space reduction

Heuristics - types

- Ignore expressions with incompatible types

```
int binsearch(int x, int v[], int n) {
    int low, high, mid;

    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = ?? / 2;
        if (x ? v[mid]) {
            high = mid - 1;
        } else if (x > v[mid]) {
            low = mid + 1;
        } else { /* found match */
            return mid;
        }
    }
    return -1; /* no match */
}
```

int or double

```
int binsearch(int a[], int x, int n){
    int l, r, i;
    l=0;
    r=n-1;
    while (l != r){
        i=(l+r)/2;
        if (x < a[i])
            r=i-1;
        else if (x > a[i])
            l=i+1;
        else
            return i;
    }
    return -1;
}
```

Heuristics - context

- Ignore expressions with no common parents

```
int binsearch(int x, int v[], int n) {
    int low, high, mid;

    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = ?? / 2;
        if (x < v[mid]) {
            high = mid - 1;
        } else if (x > v[mid]) {
            low = mid + 1;
        } else { /* found match */
            return mid;
        }
    }
    return -1; /* no match */
}
```

Binary operation

```
int binsearch(int a[], int x, int n){
    int l, r, i;
    l=0;
    r=n-1;
    while (l != r){
        i=(l+r)/2;
        if (x < a[i])
            r=i-1;
        else if (x > a[i])
            l=i+1;
        else
            return i;
    }
    return -1;
}
```