Improved Algorithm for Identification of Switch Tables in Executable Code

Andrei Gedich, Artur Lazdin
Saint–Petersburg National Research University of Information Technologies, Mechanics and Optics
Saint-Petersburg, Russia
muzhedgehog@list.ru, lazdin@yandex.ru

Abstract—**This paper introduces novel approach for identification of switch tables in executable code. Compared to existing solutions based on SSA intermediate representation and different propagation techniques, developed algorithm is more accurate and has greater flexibility. Set of iterative algorithms based on Pearson, weighted Pearson and Spearman correlation coefficients is introduced in this paper. Simple noise value filtering and improved segmentation algorithm are described.**

I. Introduction

Switch-case statement is used in many modern high level programming languages (HLL). It allows transfer of program control flow to executable code block, based on value of index expression [1]. First work that stated problems of executable code generation for switch-case statement was [2]. It contained only two basic switch-case implementations, using switch tables and balanced trees.

Identification of switch tables in executable code during code analysis is critical, because switch-case statement is one of the commonly used constructs. Switch tables store addresses of unconditional control transfer and their identification has great impact on code coverage percentage, when applying recursive traversal algorithm for executable code analysis.

In this paper switch-case statement is considered as low level language (LLL) implementation that utilizes switch tables. Any other implementations, such as binary trees, hash functions [3], index tables and other optimization strategies [2] are not considered, as they do not affect code coverage.

There is number of existing solutions that may allow switch table identification. They rely on static analysis method called slicing. Slicing is computation of set of programs statements called program slice that may affect values at some point of interest, referred as a slicing criterion. Slicing methods try to evaluate and predict range of indexes, used to access switch table elements through index register. Prediction can be based on slice [4, 5, 6] retrieved during control flow graph backtrace, using SSA intermediate representation [7], range calculation [8], constant propagation through control flow graph [8] etc.

II. Switch-Case Statement Types

Standard switch-case statement is composed from three parts: unconditional control transfer instruction that accesses switch table, switch table that contains control flow destination addresses and case expressions, where control is transferred. Default case in this context is special case of case expression. Typically compilers generate following instruction to access switch table:

JMP DWORD PTR DS:[reg*ptrsize+address].

Reg is register used as index of switch table element. Ptrsize is size of pointer. For 32-bit architecture size of pointer will be 4. Address is associated address of switch table. Associated address is stored as displacement in instruction and may not be equal to real address of switch table because sometimes compilers align switch tables with nop or semantically equivalent nop instructions. When associated address is not equal to real table address switch statement is called N-Based where N is number of switch table elements that do not exist due to alignment. Number of different types of N-Based switch constructions do exist and are described in [9].

Typical switch-case statement that can be generated by compiler is illustrated by the next assembly listing example which shows index expression and corresponding switch table containing destination addresses:

N-Based switch negative Index expression
00754CFD: JMP DWORD PTR DS:[EAX*4+754D10]
00754D04: JMP DWORD PTR DS:[ECX*4+754E0C]
00754D0B: NOP

N-Based switch negative Table data, surrounding code
00754DF3: JMP DWORD PTR DS:[EDX*4+754DFC]
00754DFA: MOV EDI,EDI
00754DFC: DD PoL_NoCd.00754E0C
00754E00: DD PoL_NoCd.00754E14
00754E04: DD PoL_NoCd.00754E20
00754E08: DD PoL_NoCd.00754E34
00754E0C: MOV EAX,DWORD PTR SS:[EBP+8]

ISSN 2305-7254

Desirable result of the switch table identification can be shown in the Table I.

TABLE I. RESULT OF SWITCH TABLE IDENTIFICATION

index	element address	destination
-4	0x00754DFC	0x00754E0C
-3	0x00754E00	0x00754E14
-2	0x00754E04	0x00754E20
-1	0x00754E08	0x00754E34

To organize access through additional index table, compilers generate pairs of following instructions. First pair assumes that index is signed value while second example assumes it is unsigned.

```
MOVZX reg2,BYTE PTR DS:[reg1+address1]
JMP DWORD PTR DS:[reg2*4+address

MOVSX reg2,BYTE PTR DS:[reg1+address1]
JMP DWORD PTR DS:[reg2*4+address2]
```

Typical switch-case statement that can be generated by compiler is illustrated by the next assembly listing example which shows index expressions and corresponding tables.

Indexed switch positive Index expressions

```
00670606: MOVZX ECX,BYTE PTR DS:[ECX+670CAC]
0067060D: JMP DWORD PTR DS:[ECX*4+670C9C]
00670614: XOR EDX,EDX
```

Indexed switch positive Tables data, surrounding code

```
00670C99: LEA ECX,DWORD PTR DS:[ECX]
00670C9C: DD PoL_NoCd.006706DA
00670CA0: DD PoL_NoCd.00670800
00670CA4: DD PoL_NoCd.00670614
00670CA8: DD PoL_NoCd.006708C4
00670CAC: DB 00
00670CAD: DB 01
00670CAE: DB 02
00670CAF: DB 03
    ...
00670CB8: DB 03
00670CB9: DB 00
00670CBA: DB 00
00670CBB: DB 00
00670CBC: INT3
```

Desirable result of the switch table identification can be shown in the Table II.

TABLE II. RESULT OF SWITCH TABLE IDENTIFICATION

index	element address	destination
0	0x00670C9C	0x006706DA
1	0x00670CA0	0x00670800
2	0x00670CA4	0x00670614
3	0x00670CA8	0x006708C4

There are cases when compilers use different approach to generate switch-case statements. This approach includes usage of offset tables instead of address tables. To organize access through additional offset table, compilers generate following sequences of instructions. It is important that offset size can be either one byte as shown on fist example or two bytes as shown on second example.

```
MOVZX reg2, BYTE PTR DS:[reg1 + address1]
ADD reg2, address2
JMP reg2

MOVZX reg2, WORD PTR DS:[reg1 * 2 + address1]
ADD reg2, address2
JMP reg2
```

Typical switch-case statement that can be generated by compiler is illustrated by the next assembly listing example which shows index expressions and corresponding offset table used to calculated destination address.

Offset switch positive Index expressions

```
01131055:JA intel_he.011312AA
0113105B:MOV EAX,DWORD PTR SS:[EBP-24]
0113105E:MOVZX EAX,WORD PTR DS:[EAX*2+113768C]
01131066:ADD EAX,intel_he.0113106D
0113106B:JMP EAX
0113106D:MOV DWORD PTR SS:[EBP-4],0
01131074:PUSH 4
```

Offset switch positive Offset table

```
113768C:2A01
113768E: F200
1137690: BA00
1137692: 7C00
1137694: 3E00
1137696: 0000
```

Desirable result of the switch table identification can be shown in the Table III.

TABLE III. RESULT OF SWITCH TABLE IDENTIFICATION

index	element address	destination
0	0x0113768C	0x01133A6E
1	0x0113768E	0x0114026D
2	0x01137690	0x0113CA6D
3	0x01137692	0x01138C6D
4	0x01137694	0x01134E6D
5	0x01137696	0x0113106D

There is couple of other ways in which switch-case statements can be generated by compilers. These include optimized versions like unwinding into sequence of dec, cmp; dec, jcc; inc, cmp; inc, jcc instructions. Such cases are not considered in this paper.

III. Properties of Switch Table Elements

If switch table elements are represented as set of values they have following properties:

- In many cases values of switch table have low scatter rate because they point to same location inside procedure. Many compilers use this property of low scatter rate when generating switch-case statements and clustering them into multiple groups [20]. It should be mentioned though that switch tables can contain strong outliers that should be taken into account.
- Elements of switch table are aligned on associated switch table address. In other words they follow without gaps because they are accessed by index.
- Each element of table represents address that belongs to address space of executable.
- Values of switch table are unordered and can be duplicated. It is obvious that number of duplicates affects probability of value to be true value of switch table.
- In many cases ordered set of unique switch table values forms dependency that is close to linear. It should be mentioned though that strong outliers may appear.
- By using properties that are listed above it can be stated that linear approximation model can be used for identification of switch table borders.

There is number of dependencies that can be formed by values of switch table. Some of them are illustrated by next figures Fig. 1, 2, 3. These figures were plotted by using scatter plot from R package called car. Fig. 1 shows strong linear dependency of switch table values, Fig. 2 shows that polygonal curve formed by switch table values can be approximated using straight line. Fig. 3 shows that there are cases where dependency is even more closer to linear.

IV. Generalized Algorithm

Generalized algorithm consists of two steps where each step contains several sub steps. In general this algorithm can be described as algorithm that reads set of possible switch table values near associated switch table address that is specified in index expression. Then set of values is filtered to exclude irrelevant noise values. After filtering few linear models are applied to estimate quality of the set.

During first step of the algorithm possible switch table is read from executable. Algorithm iteratively performs scan in both directions starting from associated switch table address. Scan in corresponding direction is stopped when invalid address value or another code or data block is encountered. In case of N-Based switch algorithm first locates accessible element and only then performs scan.

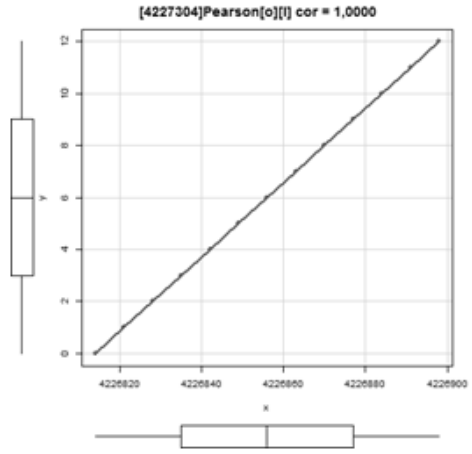

Fig. 1. Strong linear dependency between values

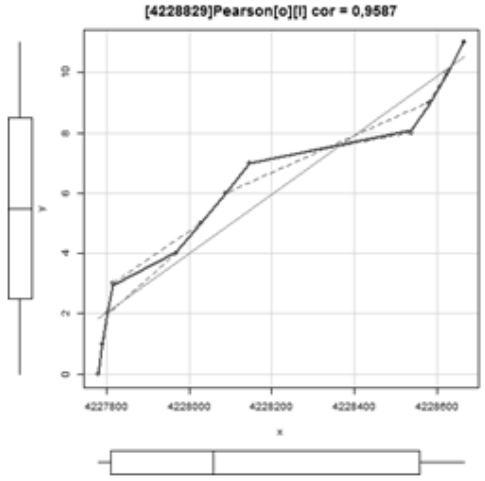

Fig. 2. Polygonal curve fit by straight line

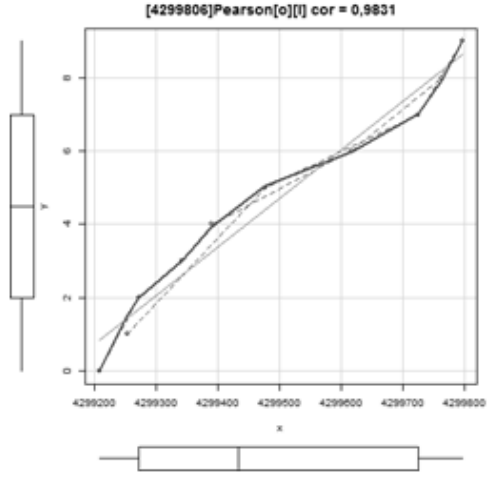

Fig. 3. Smoother polygonal curve fit by straight line

During second step algorithm analyzes value set that was acquired on step 1. In case when acquired set contains two or less values algorithm throws an error. Compilers typically optimize such cases into sequence of if-then-else statements instead of generating switch statement.

If acquired value set contains more than two values then Pearson correlation coefficient (PCC) is calculated. To calculate PCC two variables are needed. One is represented by switch table values. Another is acquired by ranking the values just as it is done when calculating Spearman correlation coefficient. If PCC value is greater than specified threshold then switch table is considered to be identified. Otherwise iterative algorithm is applied to possible switch table values to filter noise values located on outer borders of the value set.

Filtered value set is again approximated using PCC value. If calculated PCC value is greater than specified threshold then switch table is considered to be identified. Otherwise iterative segmentation algorithm is applied to value set. When segmentation is over segmentation quality criterion is calculated. If criterion value is greater than specified threshold then switch table is considered to be identified. Otherwise iterative algorithm that uses weighted PCC value for approximation is applied.

If weighted PCC value is greater than specified threshold switch table is considered to be identified. Otherwise iterative algorithm that uses Spearman correlation coefficient (SCC) is applied. If calculated SCC value is greater than specified threshold then switch table is considered to be identified. Otherwise algorithm throws the error being unable to identify switch table.

V. Iterative Noise Filtering Algorithm

Main task of this algorithm is to find special data points that are located at outer borders of possible switch table value set. This points form sharp corners on the graph leading to deviation from linear dependency. Such points can come from initial scan step. These noise values are introduced by code that was misinterpreted as data-address or by displacement part of nearest instruction.

To identify corners algorithm scans value set two times- from the start and from the end. On each iteration of the algorithm last data point is excluded from the value set and PCC value is calculated for the subset. Calculated value is then compared to the value from previous iteration. If absolute value of delta between calculated values is greater than specified threshold and current PCC value is greater than PCC value from previous iteration then algorithm continues its work. Otherwise algorithm stops.

Usage of absolute value allows small deviation of PCC value from linear dependency even if on intermediate iteration current value of PCC is less than previous value of PCC. In this case increase of PCC value means improvement of linear dependency.

VI. Applying Pearson Correlation

As is was said before PCC value is used to approximate switch table values with linear model because PCC is a classical measure of linear dependency between two variables. It takes values from +1 to -1 where +1 is maximum positive correlation and -1 is maximum negative correlation. In general case PCC is calculated as (1).

$$p = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}} \quad (1)$$

This equation is inconvenient because it requires calculation of mean. To calculate PCC equivalent formula (2) was used.

$$px = \frac{n\times\sum_{i=1}^{n}(x_i \times y_i) - \left(\sum_{i=1}^{n} x_i \times \sum_{i=1}^{n} y_i\right)}{\sqrt{\left[n\times\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\times\left[n\times\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}} \quad (2)$$

As it was said two variables X and Y are required to calculate PCC. In this case X stands for possible values of switch table which are ordered because this is not guaranteed by compilers. Y is calculated by ranking values of variable X as it is done when calculating SCC. Values of variable X are ordered.

VII. Segmentation Algorithm

In some cases acquired switch table value set has low value of PCC. This can be related to the fact that values set consist of segments where each segment has linear dependency. This situation happens when compilers put two switch tables together without gap between them. Segmentation algorithm is used to split original value set into groups and select one that belongs to switch table accessed by index expression.

In this paper improved segmentation algorithm based on [9] is introduced. First, initial value set is formed which is subset that contains first two data points from original switch table value set that is analyzed. Next, algorithm iteratively scans original value set and includes each value in the subset. Calculating PCC value allows detecting rapid change of linear dependency. If inclusion of the data point in the subset changes PCC value so that delta is greater than specified threshold then data point is considered to be beginning of next segment. Algorithm continues its work until original value set is scanned till the end.

It should be mentioned that this algorithm is applicable only if original value set contains 4 or more points that can form at least two linear segments. Original value set should not contain noise values on the outer borders which are removed on previous step.

VIII. SEGMENTATION QUALITY CRITERION

To give an estimate of segmentation quality some numeric criterion that can be compared with specified threshold is required. In this paper criterion Q_1 is introduced which is calculated as (3). This criterion is product of segment sizes containing unique values.

$$Q_1(S_U) = \prod_{i=1}^{k} \left| S_{U(i)} \right|. \quad (3)$$

S_U–segment set where each segment represents set of unique values.
U()–unique value set operator.
$S_{U(i)}$–i'th segment containing unique values, calculated as:

$$S_{U(i)} = U(S_i). \quad (4)$$

S_i–i'th segment of original segment set.
S–segment set where each segment represents value set.
||–set size operator.
k–number of segments that is calculated as:

$$k = \left| S_U \right|. \quad (5)$$

For Q_1 criterion to be estimated it should be normalized. For normalization min and max value of criterion should be known. Normalized Q_1 criterion is calculated as:

$$Q_{1norm} = \frac{Q_1 - Q_{1\min}}{Q_{1\max} - Q_{1\min}}. \quad (6)$$

Q_{1max}–max product of segment sizes from S_U segment set that is calculated as:

$$Q_{1\max} = \left[\frac{m}{k} \right]^{k-\mathrm{mod}(m,k)} * \left(\left[\frac{m}{k} \right] + 1 \right)^{\mathrm{mod}(m,k)}. \quad (7)$$

Q_{1min}–min product of segment sizes from S_U segment set that is calculated as:

$$Q_{1\min} = m - k + 1. \quad (8)$$

[]–integer division operator.

mod(n,d)–modulo operator.
m–number of elements in P_U set that is calculated as:

$$m = \left| P_U \right|. \quad (9)$$

P_U–original value set of unique values calculated as:

$$P_U = U(P). \quad (10)$$

P–original value set.

Introduced criterion Q_{1norm} represents normalized product of segment sizes containing unique values from original value set. Criterion takes values in range of 0 to 1. This criterion can be interpreted as maximum equal filling of segments with elements. If criterion value is low then segments are unbalanced. In this case it can be stated that segmentation failed. If criterion value is high then segments are nearly equally filled. In this case it can be stated that segmentation has succeeded.

IX. APPLYING WEIGHTED PEARSON CORRELATION

As it was said before segmentation is applied when PCC value is low for original value set that represents possible switch table values. In some cases segmentation may fail. Typically this can happen when value set is unordered because ordering cannot be applied during segmentation.

In some cases there can be data points in the set that have duplicates. Such points represent strong outliers that cannot be removed by methods described above because they are located in the middle of value set.

This paper introduces iterative algorithm that allows to estimate deviation of strong outliers from main value set when PCC value is low. This algorithm is based on calculation of weighted PCC value.

As in the case of PCC original value set is ordered before calculations. Task of the algorithm is to distribute weights between elements of original value set and calculated weighted PCC value.

PCC parameterized by weight can be calculated as:

$$corr(x, y; w) = \frac{\mathrm{cov}(x, y; w)}{\sqrt{\mathrm{cov}(x, x; w)\,\mathrm{cov}(y, y, w)}}. \quad (11)$$

Where cov(x,y;w) – is covariance parameterized by weight and calculated as:

$$\mathrm{cov}(x, y; w) = \frac{\left(\sum_{i=1}^{n} w_i (x_i - m(x; w))(y_i - m(y; w)) \right)}{\sum_{i=1}^{n} w_i} \quad (12)$$

Where m(x;y)–is mean parameterized by weight and calculated as:

$$m(x; w) = \left(\sum_{i=1}^{n} w_i x_i \right) \div \sum_{i=1}^{n} w_i. \quad (13)$$

To calculate weighted PCC value additional weight set is required that contains weight for each element of original value set. It is obvious that number of duplicates can be used as weight but experiments showed that in some cases it is not enough.

This paper introduces weight function that allows calculation of series of weight sets and can be manipulated by additional input parameter called factor.

First, mean weight value is calculated as:

$$\overline{W} = \left(\sum_{i=1}^{n} W_i \right) \div n \cdot \quad (14)$$

W_i – element of original weight set that is composed of duplicate element counts as it was described before.

Parameterized function that gives weight set as result can be calculated as:

$$\overline{W_f}(W; f) = \overline{W} + (W - \overline{W}) \times f \,, \quad (15)$$

f–degree of amplitude distortion of weight curve.

When algorithm is applied it iteratively increases factor f that has start value of 1. On each iteration it calculates weight set using equation (15). Each element of weight set represents f times increased sum of mean weight value and deviation of i'th value from the mean.

When weight set is calculated weighted PCC value is calculated for current iteration. If calculated value is greater than specified threshold then switch table is considered to be identified. Otherwise algorithm continues its work till it converges or reaches limit specified for parameter f. It is important that algorithm is not guaranteed to converge so if limit f is reached then identification of switch table is considered to be failed.

X. Applying Spearman Correlation

There can be cases where previously introduced algorithms fail to identify switch tables. This paper introduces last approach based on SCC value. This criterion represents non parametric measure of dependency between two variables. It shows how well can dependency be expressed using monotonic function. Advantage of this method is lower sensitivity to strong outliers compared to PCC. Value of SCC can be calculated as:

$$s = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2} \sqrt{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}} \quad (16)$$

It should be mentioned that during calculation of SCC value ordering of original value set is restricted because this automatically leads to monotonic dependency. SCC estimate should be used only as last approach when trying to identify switch table.

XI. Conclusion

This paper introduces improved algorithm for identification of switch tables in executable code which is based on [9]. Main advantage of this algorithm is simplicity in implementation and understanding compared to methods based on slicing techniques. Moreover, introduced algorithm has higher accuracy and can be configured in multiple ways by its user.

XII. Acknowledgments

Research was supported and funded by grant 12-07-00376-a of the Russian Foundation for Basic Research

References

[1] R.A. Sayle, "A Superoptimizer Analysis of Multiway Branch Code Generation", *Proceedings of the GCC Developers Summit*, 2008, pp. 1–16.

[2] A. Sale, "The Implementation of Case Statements in Pascal", *Software – Practice and Experience*, vol. 11, 1981, pp. 929–942.

[3] H.G. Dietz, "Coding Multiway Branches Using Customized Hash Functions", *ECE Technical Report,* School of Electrical Engineering, 1992, pp. 1–30.

[4] C. Cifuentes, A. Fraboulet, "Intraprocedural static slicing of binary executables", *In International Conference on Software Maintenance*, 1997, pp. 188–195.

[5] H. Agrawal, "On Slicing Programs with Jump Statements", *In Proceedings of ACM SIGPLAN'94 Conference on Programming Language Design and Implementation*, 1994, pp. 60–73.

[6] C. Cifuentes, M. Van Emmerik, "Recovery of Jump Table Case Statements from Binary Code", *Science of Computer Programming*, 2001, pp. 171–188.

[7] M. Van Emmerik, "Static Single Assignment for Decompilation", *School of Information Technology and Electrical Engineering*, The University of Queensland, 2007

[8] J. Patterson, "Accurate Static Branch Prediction by Value Range Propagation", *Proceedings of the ACM SIGPLAN'95 Conference on Programming Language Design and Implementation*, 1995, pp. 67–78.

[9] A. Gedich, "Automatic search of switch-case statements and identification of switch tables in executable code", *Science and Education*, vol. 5(2), 2014, pp. 308-316.