文章编号：2095-2163(2021)08-0177-06 中图分类号：TP391.41 文献标志码：A

# 基于动态检测与静态分析的自动评分方法研究

薛 斌，胡建鹏
（上海工程技术大学 电子电气工程学院，上海 201620）

**摘 要**：为了给 C 语言编程题进行合理评分，本文提出了一种新型的自动评分方法，在动态检测阶段先利用 KMP 算法执行关键字匹配，若匹配相似度落入预期值区间，则将学生源程序转换为可执行文件，通过预先设置的测试用例来驱动评分；若关键字匹配未通过、程序无法运行或者运行期间出现异常，则执行静态分析。静态分析阶段选取控制结构作为静态评分的关键因素，采用抽象语法树作为源代码的中间转换形式，并对其标准化以消除代码语义的多样性；根据抽象语法树中的结点类型提取出控制结构子树；最后，利用基于结点权值的树编辑距离算法来匹配标准化后的学生源程序与模板程序的控制结构子树，计算相似度并给出综合评分结果。实验结果表明，该方法能够对程序进行合理有效地评分，并且具有较高的准确率。

**关键词**：自动评分；抽象语法树；KMP 算法；树编辑距离

# Research on automatic scoring system of programming questions based on dynamic detection and static analysis

XUE Bin, HU Jianpeng
(School of Electric and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

**【Abstract】** This article proposes a new type of automatic grading method to reasonably grade students' C language programming questions. In the dynamic detection stage, the KMP algorithm is used to perform keyword matching first. If the matching similarity falls within the preset value range, the student source program is converted into an executable file, and then the score is driven by the preset test cases. Perform static analysis when the keyword match fails, the program cannot run, or an exception occurs during operation. In the static analysis stage, the control structure is selected as the key factor of the static scoring. The abstract syntax tree is used as the intermediate conversion form of the source code, and it is standardized to eliminate the diversity of code semantics. Then, the control structure subtree is extracted from the node type in the abstract syntax tree Finally, use the tree edit distance algorithm based on node weights to match the standardized control structure subtrees of the student source program and the template program, calculate the similarity, and give a comprehensive scoring result. The final experimental results show that this method can reasonably score student programs and has a high accuracy rate.

**【Key words】** automatic scoring; abstract syntax tree; KMP algorithm; tree edit distance

## 0 引 言

C 语言是高校中工科专业的必修基础课程，也是国际上广为流行的高级程序设计语言。在 C 语言程序设计课程中，考核学生知识掌握程度的重要手段之一就是对编程能力的考察，学生只有通过大量的上机代码实践，才能更好地理解 C 语言程序的精髓。目前，高校普遍采用线上系统对 C 语言编程题进行上机考试，但此类系统大多仅有通过与错误两种运行结果。只关注程序运行的结果，而忽视程序本身，这既不能合理地考察学生对知识的掌握程度，也不能让学生得到及时有效的反馈。为了对运行结果异常以及不能运行的学生程序进行评分，教师仍需要把学生编写的答案打印出来，进行人工评分，不仅效率低下，而且评分结果可能会受到阅卷老师的主观因素影响。

本文结合动态检测与静态分析策略，设计了一种新型的 C 语言自动评分方法，并将该方法融合到线上实验系统中。首先，进行动态检测，利用 KMP 算法执行关键字匹配，若匹配相似度落入预期值区间，再将学生程序代码转换为可执行文件，通过预先设置的测试用例来驱动评分。若关键字匹配未通过、程序无法运行以及运行期间出现异常，则进行静态分析，对学生源程序代码和模板程序代码进行标准化，通过控制结构子树的匹配相似度来综合评分。

作者简介：薛 斌(1994—)，男，硕士研究生，主要研究方向：系统自动评分技术、网络负载均衡、云边协同；胡建鹏(1980-)，男，博士，副教授，主要研究方向：软件工程、服务计算、云计算与物联网等。
通讯作者：胡建鹏 Email：mr@ sues.edu.cn

收稿日期：2021-01-22

## 1 动态检测方案设计

基于动态检测的自动评分方法属于软件自动化测试方法[1],其核心的检测步骤如下:预先设置好测试用例作为输入数据;把源程序转换为可执行文件并运行;通过判断期望值与输出数据是否相等来进行自动评分。但是对于某些编程题来说,学生可以对照测试用例来编写代码,以达到欺骗系统并获取得分的目的。为了杜绝此类状况,并减少系统无效运行的次数,本系统在执行动态检测时,首先会通过 Knuth-Morri-Pratt[2](简称 KMP)算法进行关键字的匹配,当整体匹配相似度落入预期值区间时,便执行后续步骤,否则转入静态检测流程[3]。

### 1.1 基于 KMP 算法的关键字检测

本系统主要针对非计算机专业的学生群体,编程考题相对简单,而且很多 C 语言关键字也不在教学使用范围内,如 register、auto、volatile,故本文在原有关键字范围基础上做一些删减,并增加了有利于考察代码逻辑的特征词,如 stdio.h、stdlib.h、sqrt()、abs()等。针对关键字检测的操作步骤如下:

(1)从头至尾扫描程序代码,删掉代码中所有的空格、空行以及注释;

(2)对学生源程序进行词法分析,检查程序的 token 流,剔除掉所有的变量与自定义的函数名,剩下的即为 C 语言关键字所组成的字符串,把这些字符串作为关键字匹配的特征字符串;

(3)利用 KMP 算法对源程序模式串和预设的主特征字符串进行匹配并记录匹配结果,最终计算出整体的相似度。

### 1.2 动态检测执行

基于动态检测的评分方法主要解决的问题包括:设置合理的测试用例,尽可能完整地覆盖代码的执行路径;配置系统程序的沙盒安全保护机制;检测数据结果的分析。动态检测的整体步骤如下:

(1)关键字匹配:使用 KMP 算法对源程序与模板程序中的关键字进行匹配操作;

(2)预处理:读取系统配置文件、读取编程题目信息以及测试用例数据;

(3)编译:对程序进行编译并检查语法,以生成可执行程序;

(4)执行:运行程序,依次输入全部测试数据,监控代码的执行状态;

(5)测试:获取学生程序的执行结果,包括正常执行的输出结果或程序异常结束的结果,若正常执行并退出,则将程序的输出数据与模板的标准数据进行对比并评定分数。基于动态检测的自动评分模型如图 1 所示。

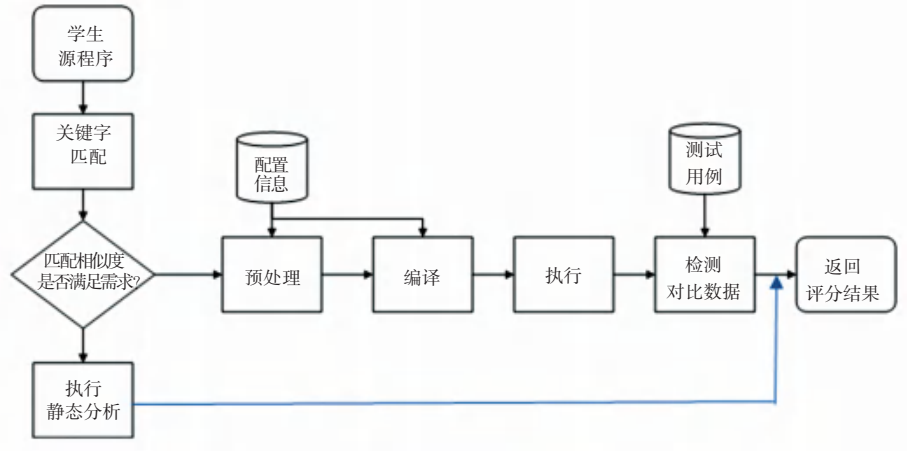


**图 1 动态检测评分模型**

**Fig. 1 Dynamic detection scoring model**

## 2 静态分析方案设计

通过对程序进行词法分析、语法分析等预处理操作,输出程序的中间转换形式——抽象语法树(Abstract Syntax Tree, AST)[3],继而模拟人工评分的思路:

(1)首先在组建编程题库时对每道题目提供得分点,本文选取控制结构作为静态评分的关键因素;

(2)对抽象语法树进行标准化处理,以消除程序语义的多样性,减少答案模板的数量,然后根据 AST 中结点的类型提取出对应的控制结构子树;

(3)利用基于结点权值的树编辑距离算法来匹配标准化后的源程序与模板程序的控制结构子树,并计算其相似度[4];求解控制结构各个模块的得分

值(模块的预设总分值乘以对应模块的相似度)；最后，综合各模块的评分结果求和并得出静态分析的最终评分结果。

### 2.1　程序标准化

抽象语法树作为一种数据载体，不仅包含了源程序对应结点的所有信息，还包括了结点之间的结构关系。针对抽象语法树进行标准化，包含两个部分：

(1)利用基于关键词 Trie 树的 GCC 抽象语法树消除冗余算法，对抽象语法树进行冗余优化处理[5]，冗余优化示意如图 2 所示；

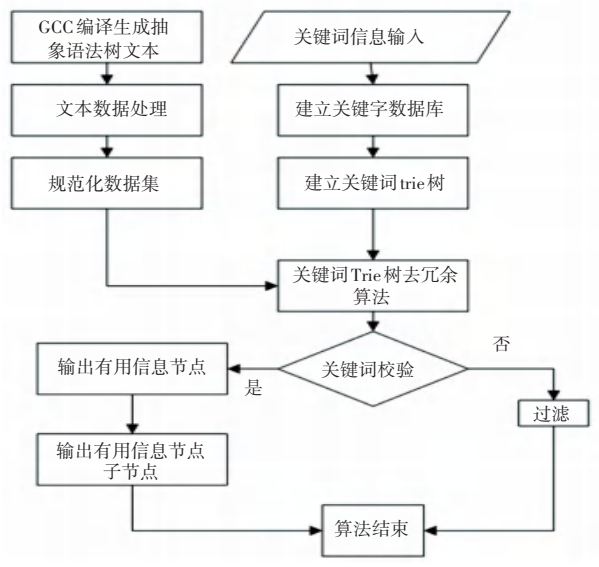


**图 2　冗余优化整体示意图**

**Fig. 2　Overall schematic of redundancy optimization**

(2)通过对程序运用一系列标准化规则，在抽象语法树的基础上，将不同表现形式的程序语法树转换成统一的表现形式，这样不仅可以消除学生源程序代码的多样性、减少标准答案模板的数量，还能提高相似度匹配的准确率。

为了后续更好地实现程序控制结构的相似度匹配，本文在程序标准化环节主要对程序控制结构进行标准化转换。在 C 语言中有 3 种基本的控制结构：顺序结构、选择结构和循环结构。每一种控制结构都可能有多种代码书写形式，这为源程序和模板程序的匹配带来困难，所以需要对控制结构进行标准化。

#### 2.1.1　选择结构的标准化

实现选择结构可采用 if 语句或 switch 语句，if 语句对应的抽象语法树有如下结构特点：if 结点有 3 个孩子结点，从左至右依次排开，分别是条件表达式结点、执行语句结点和 else 语句结点；而在 switch 语句中，switch 结点则可能有多个孩子结点，switch 括号内的表达式结点在最左边，default 结点在最右边。

针对选择结构的两种不同形式的抽象语法树，首先转换为统一的语法树形式，再进行标准化处理，包括：删除得不到执行的分支、删除空分支、提取各分支中的公共表达式、合并选择结构、将各分支深层次的嵌套结构转换为多分支选择结构。标准语法树形式以及深层次嵌套结构转换如图 3 所示。

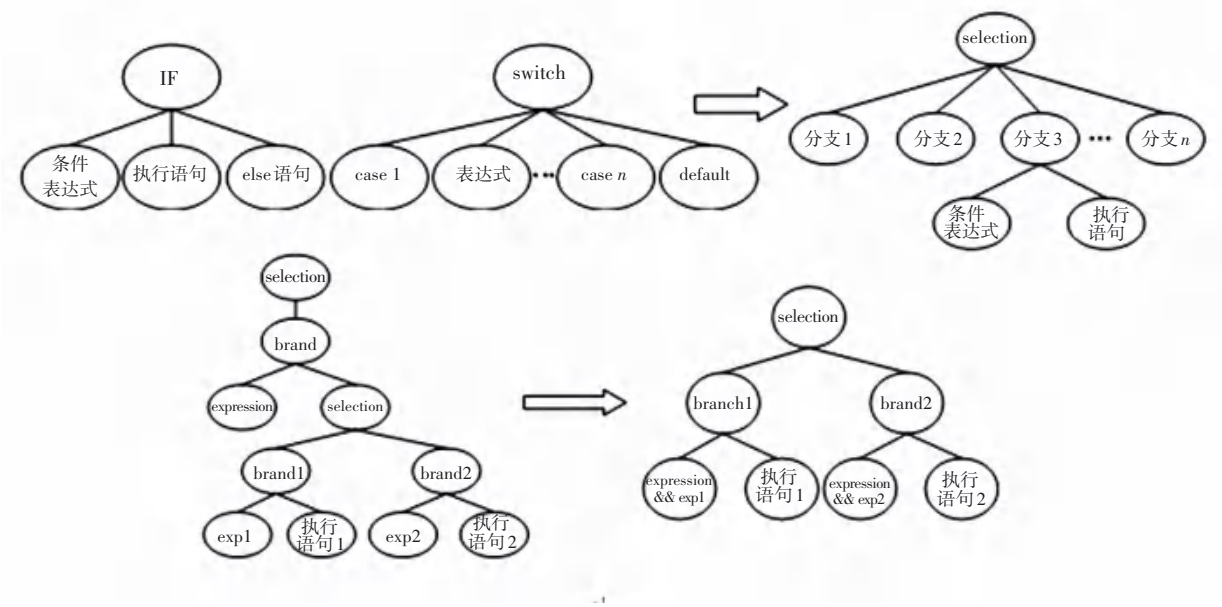


**图 3　选择结构标准语法树以及深层次嵌套结构转换示意图**

**Fig. 3　The standard syntax tree of the selected structure and the conversion diagram of the deep nested structure**

2.1.2 循环结构的标准化

实现循环结构可采用 while 语句、for 语句和 do…while 语句。相对 while 语句,do…while 语句至少会执行一次循环体,本文在语法分析阶段对 do…while 语句做了预处理,将其循环体拷贝一次后,再转换为 while 语句。因此,只讨论对 while 语句和 for 语句所对应的循环结构进行标准化。

针对上述两种循环语句所对应不同形式的抽象语法树,首先统一其表现形式,将循环语句的结点均表示为 *Loop* 结点,并对其分支结构进行统一的规范化处理和标准化转换,包括:重新排列各层循环的顺序、删除条件表达式恒为假的循环分支、将循环体中值不会发生改变的语句提取到循环体外等。

另外,for 语句需要进行额外的处理:将初始化表达式插入到 *Loop* 结点之前,条件表达式的位置保持不变,再将递增递减表达式放到循环体中。因为大多数条件表达式属于关系表达式,所以可对其进行适当的处理,如:$i \leqslant N \leftrightarrow i < N + 1$。

**2.2 基于树编辑距离的控制结构子树匹配**

控制结构子树主要包含选择结构子树(if、switch 语句)以及循环结构子树(while、for 语句),因此选取这两种控制结构作为主要得分点来进行相似度匹配。在求解树编辑距离时,需要让对应的整棵树进行分解,并生成一系列子树的集合,再去计算集合中每一棵子树与其对应模板子树的树编辑距离,递归地重复上述计算与分解步骤,直到集合中的子树不能再分解为止。

2.2.1 树编辑距离算法

树编辑距离指的是完成从有序树 $T_1$ 到有序树 $T_2$ 的树映射 $f(T_1 \rightarrow T_2)$ 所需要进行的树编辑操作的最小代价。其中,树编辑操作包含树结点的插入、删除以及替换操作,并且每一个编辑操作都有相应的代价,用公式(3)表示:

$$d(T_1, T_2) = \min\{\mu(f(T_1 \rightarrow T_2))\}. \quad (3)$$

其中,$\mu(f(T_1 \rightarrow T_2))$ 是完成有序树之间的映射 $f(T_1 \rightarrow T_2)$ 所需要的编辑操作的代价,而参数 $d(T_1, T_2)$ 则用来衡量两棵树之间的异构程度。

如图 4 所示,用虚线连接与编辑操作无关的树结点,同时参照各个考核点的重要程度来标注出对应结点的权值大小。因此,可以利用树编辑距离来衡量树之间的匹配相似度。编辑操作包括删除结点 *decl*2、插入结点 *expr*2,由树编辑距离的定义可知:$d(T_1, T_2) = 2$,并且当两棵树之间的树编辑距离越大,说明两棵树之间映射所需要的编辑操作越多,因而两棵树之间的匹配相似度越小。

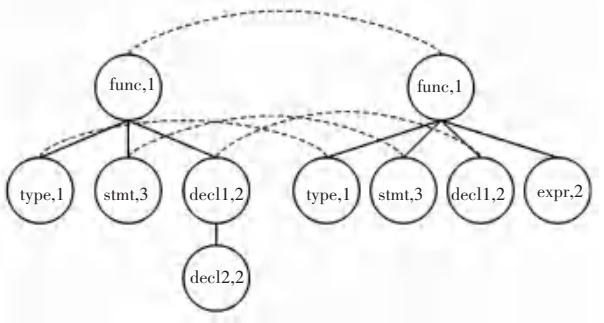


**图 4 $T_1 \rightarrow T_2$ 的映射**

**Fig. 4 Mapping between $T_1$ and $T_2$**

2.2.2 匹配子树并计算相似度

在实际考试中每道编程题所要考核的重点内容会有所不同,为了更好地完成特定的教学目标,本文在静态分析阶段对子树中的每个结点赋予了不同的权值以反映其对应考察点的难易性与重要程度。结点的权值越大,则代表该结点所对应考核的内容越重要或者难度越大。本文通过利用树中各结点所赋予的权值大小,来计算源程序与模板程序之间的匹配相似度。该算法描述见表 1。

根据算法得出加权树之间的编辑距离(即 $wEdist = M[m][n]$),同时,利用基于结点权值的树编辑距离公式(4)来求解源程序与模板程序之间的匹配相似度:

$$sim(T_1, T_2) = \frac{w(T_1) + w(T_2) - wEdist(T_1, T_2)}{w(T_1) + w(T_2)}. \quad (4)$$

以图 4 中的有序树映射 $f(T_1 \rightarrow T_2)$ 为例,由上述算法可知,加权树 $T_1$ 与 $T_2$ 之间的树编辑距离为 4,由公式(4)计算得到匹配相似度为 0.74。

## 3 系统实验结果与分析

为测试本系统的有效性和准确率,请 80 名学生使用本系统进行编程考核。系统模板试题库一共选取了 10 道具有代表性的编程题目,每一位学生会随机抽取一道题目进行解答,每道编程题的分值为 10 分。使用了动态检测与静态分析相结合的自动评分方案,该方案所采用的评分标准见表 2。

评分标准中首先进行关键字检测,若检测未通过则直接扣除 4 分,若检测通过便动态执行源程序,执行期间若出现异常或不能正常运行,则扣除 2 分。动态检测阶段完成后执行静态分析,结合控制结构匹配相似度进行综合评分。

为更好地对比自动评分与人工评分之间的差异,本文使用绝对误差与相对误差来衡量。

**表 1　算法描述**

**Tab. 1　Algorithm Description**

基于树编辑距离的匹配算法：

Input：　抽象语法树 $T_1$、$T_2$

Output：　$T_1$ 与 $T_2$ 之间的树编辑距离

Declare：　$m$、$n$ 分别代表树$T_1$、$T_2$ 的孩子结点数目；$t_1[k]$、$t_2[k]$ 分别为 $T_1$、$T_2$ 的第 $k$ 个孩子结点；$w$ 为对应结点的权值；$wEdist$ 为两棵加权树之间的编辑距离；针对结点 $N$ 与矩阵 $M$，有如下初始化：

Initialization：

$$C_i = \sum_{N \in T_c} w(N), T_c = Children(t_1[i]);$$

$$C_j = \sum_{N \in T_c} w(N), T_c = Children(t_2[j]); M[0][0] = 0;$$

$$M[i][0] = M[i-1][0] + \sum_{k}^{t_1[k] \in C_i} w(t_1[k]), \text{ for } i = 1 \text{ to } m$$

$$M[0][j] = M[0][j-1] + \sum_{k}^{t_1[k] \in C_j} w(t_2[k]), \text{ for } j = 1 \text{ to } n$$

Step1　for $i = 1$ to $m$

Step2　for $j = 1$ to $n$

Step3　$\lambda(d) = M[i-1][j] + \sum_{k}^{t_1[k] \in C_i} w(t_1[k]);$

Step4　$\lambda(i) = M[i][j-1] + \sum_{k}^{t_1[k] \in C_j} w(t_2[k]);$

Step5　*if* $t_1[i]$ 与 $t_2[j]$ 均为叶子结点

Step6　$$\lambda(r) = \begin{cases} M[i-1][j-1], \ if\ t_1[i] = t_2[j] \\ M[i-1][j-1] + w(t_1[k]) + w(t_2[k]), \text{ otherwise} \end{cases}$$

Step7　else if $t_1[i]$ 为叶子结点

Step8　$$\lambda(r) = \begin{cases} M[i-1][j-1] + \sum_{k}^{t_2[k] \in C_j} w(t_2[k]), if\ t_1[i] = t_2[j] \\ M[i-1][j-1] + w(t_1[k]) + \sum_{k}^{t_2[k] \in C_j} w(t_2[k]), \text{otherwise} \end{cases}$$

Step9　else if $t_2[j]$ 为叶子结点

Step10　$$\lambda(r) = \begin{cases} M[i-1][j-1] + \sum_{k}^{t_1[k] \in C_i} w(t_1[k]), if\ t_1[i] = t_2[j] \\ M[i-1][j-1] + w(t_2[k]) + \sum_{w}^{t_1[k] \in C_{ik}} (t_1[k]), \text{ otherwise} \end{cases}$$

Step11　else

Step12　$$\lambda(r) = \begin{cases} wEdist(t_1[i], t_2[j]), if\ t_1[i] = t_2[j] \\ M[i-1][j-1] + \sum_{w}^{t_1[k] \in C_{ik}} (t_1[k]) + \sum_{w}^{t_2[k] \in C_{ik}} (t_2[k]), \text{ otherwise} \end{cases}$$

Step13　end if

Step14　$M[i][j] = min\{wEdist, \lambda(i), \lambda(r)\}$

Step15　end for

Step16　end for

Step17　return$M[m][n]$

Step18　End

表 2 编程题的评分标准

Tab. 2 Scoring criteria for programming questions

| 总分值 | 关键字检测 | 编译正常执行 | 控制结构匹配 |
|---|---|---|---|
| 10 | 2 | 2 | 6 |

同时,每个试题编号所对应的学生抽取人数,见表 3。

表 3 各编程试题的抽取人数

Tab. 3 The number of students drawn for each program problem

| 试题编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 学生人数 | 5 | 8 | 11 | 7 | 6 | 10 | 9 | 8 | 9 | 7 |

对于第 $k$ 个编程试题, $T_k$ 代表人工评分所得到的分值; $M_k$ 代表自动评分得到的分值; $n$ 表示抽取到第 $k$ 个试题的学生人数。利用绝对误差 $\Delta$ 与相对误差 $\delta$ 的数学公式(5)进行计算,得出的误差统计数据,见表 4。

表 4 自动评分方案的误差统计

Tab. 4 Error statistics of automatic scoring scheme

| 试题编号 | 人工评分的平均得分 | 自动评分方案的平均得分 | 绝对误差值 | 相对误差值 |
|---|---|---|---|---|
| 1 | 8.56 | 8.95 | 0.39 | 4.6% |
| 2 | 7.63 | 8.12 | 0.49 | 6.4% |
| 3 | 7.20 | 7.84 | 0.64 | 8.9% |
| 4 | 8.12 | 7.69 | 0.43 | 5.3% |
| 5 | 7.74 | 8.37 | 0.63 | 8.1% |
| 6 | 6.75 | 7.17 | 0.42 | 6.2% |
| 7 | 6.92 | 6.58 | 0.14 | 4.9% |
| 8 | 7.40 | 7.02 | 0.38 | 5.1% |
| 9 | 6.89 | 7.43 | 0.54 | 7.8% |
| 10 | 8.13 | 8.69 | 0.56 | 6.9% |

$$\begin{cases} \Delta_k = |T_k - M_k|, \\ \delta_k = \left(\frac{1}{n}\sum_{k=1}^{n}\frac{\Delta_k}{T_k}\right) \times 100\%. \end{cases} \tag{5}$$

根据表 4 可知,自动评分方案符合人工评分的思路,并且误差值也均在合理的范围内,这说明自动评分方案具有较高的准确率,同时大大提高了评分的效率,具有很高的实用价值。

## 4 结束语

本文提出一种基于动态检测与静态分析相结合的评分方法,该方法已经实现并被集成到在线实验系统中。实验结果表明,本系统评分准确率较高,系统稳定高效,基本满足学生编程题的自动评分需求,达到预期效果。

## 参考文献

[1] WANG Xiaoyan, WANG Weijie. Design and Implementation of JAVA Online Examination System Based on WEB 2014. 3634: 2788-2791.

[2] 曾诚,李兵,何克清. KMP 算法在 Web 服务语义标注中的应用[J].微电子学与计算机,2010,27(8):1-3,8.

[3] Jonathan van den Berg Hirohide Haga. Matching Source Code using Abstract Syntax Trees in Version Control Systems[J]. Journal of Software Engineering, and Application, 2018,11(6): 19-24.

[4]刘守群,朱明, 谭晓彬.一种基于树匹配的网页语义块挖掘算法[J].小型微型计算机系统,2009,30(8):1541-1545.

[5] 韩磊,胡建鹏.基于关键词 Trie 树的 GCC 抽象语法树消除冗余算法[J].计算机科学,2020,47(9):47-51.

(上接第 176 页)

DWT-SVD 技术结合在一起,不在是传统意义上将水印信息直接嵌入与提取,而是利用视觉密码技术将两幅水印信息分享份图像同时嵌入到载体图像中,即使水印算法被攻击者窃取,分享份图像被别人提取出来,也不会获取关于水印信息的任何内容,具有一定的安全性。本算法同时实现了在载体图像上嵌入两个分享份图像,一定程度上减轻了保存的工作量。通过实验结果,嵌入两幅水印图像后的载体图像,在受到攻击后与原始载体图像的峰值信噪比依旧可以取得不错的清晰度,达到很高的完整性,受到攻击后提取出的水印分享图像与原始水印分享图像 $NC$ 值依旧很接近 1,相似程度很高。

## 参考文献

[1] 赵翔,郝林. 数字水印综述[J]. 计算机工程与设计,2006,27(11):1946-1950.

[2] 温泉,孙锬锋,王树勋. 零水印的概念与应用[J]. 电子学报,2003,31(2):214-216.

[3] 曲长波,杨晓陶,袁铎宁. 小波域视觉密码零水印算法[J]. 中国图象图形学报,2014,19(3):365-372.

[4] 曲长波,李栋栋. 基于视觉密码和边缘检测的零水印算法[J]. 计算机应用与软件,2016,33(9):328-333.

[5] 曲长波,吴德阳. 基于 Curvelet-DSVD 和视觉密码的强鲁棒零水印算法[J]. 计算机应用研究,2019,36(2):532-537.

[6] 李春燕. 基于像素不扩展视觉密码的水印算法[J]. 大理大学学报, 2017,2(6):19-21.

[7] NOAR M, SHAMIR A. Visual Cryptography[C]//Advances in Cryptology: EUROCRYPT' 94. Berlin, Heidelberg: Spring, 1995: 1-12.

[8] 王洪君,牟晓丽,鲁晓颖,等. 像素不扩展的(2,3)视觉密码方案[J]. 吉林大学学报(信息科学版),2014,32(1):82-87.

[9] 王洪君,赵腾飞,尚大龙,等. 具有掩盖图像的像素不扩展的(2,2)视觉密码方案[J]. 南京大学学报(自然科学),2018,54(1):157-162.