International Journal of Innovative
Computing, Information and Control
Volume **7**, Number **8**, August **2011**

ICIC International ©2011 ISSN 1349-4198
pp. **4729–4739**

A TWO-STEP IN-CLASS SOURCE CODE PLAGIARISM DETECTION METHOD UTILIZING IMPROVED CM ALGORITHM AND SIM

Asako Ohno[1] and Hajime Murao[2]

[1]Department of Life Design
Shijonawate Gakuen Junior College
4-10-25, Hojo, Daito, Osaka 574-0011, Japan
asako.ohno@mulabo.org

[2]Graduate School of Intercultural Studies
Kobe University
1-2-1, Tsurukabuto, Nada 657-8501, Japan
murao@i.cla.kobe-u.ac.jp

Received May 2010; revised October 2010

Abstract. *Source code plagiarism is becoming one of the most serious problems in academia. There have been many proposed methods that attempt to detect source code plagiarism in programming classes. Most of them extract algorithmic features from the source code and measure the similarity between them. These methods show high levels of accuracy in evaluation experiments, and however, it is concerning that the similarity detected by the methods might not be caused by plagiarism. As a result, we propose a method called the CM Algorithm, which utilizes a student's coding style, the way the student writes source code, to check whether the source code submitted by the student was produced by him/her. In this paper, we propose a combined method that measures the similarity between source codes by using SIM* [7]*, one of the well-known in-class source code plagiarism detection systems, and then checks the outputs of SIM against our improved CM Algorithm. The new method is expected to reduce false positives in plagiarism detection systems. This paper also gives a detailed explanation of the improved CM Algorithm, which assumes fluctuations in the source code produced by a student's coding style.*
Keywords: Coding style, In-class source code plagiarism, Coding model, Hidden Markov model

1. **Introduction.** World-wide Internet access enables students to find many examples of programming source code, and so it is not difficult for these to be a source of plagiarism. Also, since it is becoming more common for students to edit or submit exercises electronically, it has become easier for a student to copy another student's work and misrepresent it as his/her own work. Recently, a number of methods have been proposed to solve one of the serious and growing problems in academia, namely source code plagiarism. The most popular approach is to measure structural, algorithmic, or metric-based similarities of the source code [1, 2, 3]. This type of method has been successfully utilized in industrial fields, such as source code refactoring. Many methods make tokenization and normalization as a preprocessing step to represent the source code by using graphs [4, 5, 6] or token sequences [7, 8, 9, 10, 11], and then make pair-wise comparisons. Other methods generate feature vectors, which are quantitative representations of source code based on such features such as size, complexity, or a number of specific types of tokens contained in the source code [12, 13]. In another example, D'Souza [14] developed a tool that finds similar contents that might be plagiarized from the Internet. Source codes produced as exercises in programming classes, hereinafter called *in-class source codes*, generally have