TECHNICAL SCIENCES

РАЗРАБОТКА МЕТОДОВ ОЦЕНКИ СХОДСТВА АЛГОРИТМОВ НА ОСНОВЕ ГРАФОВЫХ МОДЕЛЕЙ

Никитин В. Д.,
Магистрант Днепропетровского национального университета железнодорожного транспорта имени академика В. Лазаряна, г. Днепр, Украина
Иванов А. П.
Доцент кафедры компьютерных информационных технологий Днепропетровского национального университета железнодорожного транспорта имени академика В. Лазаряна, г. Днепр, Украина

DEVELOPMENT OF METHODS FOR ASSESSING THE SIMILARITY OF ALGORITHMS BASED ON GRAPH MODELS

Nikitin V.,
Master student of Dnepropetrovsk National University Railway Transport named after Academician V. Lazaryan, Dnipro, Ukraine
Ivanov A.
Associate Professor at the Department of Computer Information Technologies Dnipropetrovsk National University of Railway of Transport named after Academician V. Lazaryan, Dnipro, Ukraine

Аннотация:
В статье рассматривается вопрос оценки сходства алгоритмов и программ для ЭВМ. Изучается построение графовых моделей алгоритмов на основе текстов программ на прикладных языках программирования, а также приводится метод сравнения данных моделей. Анализируется эффективность разработанной методологии на примерах простых алгоритмов.

Abstract:
The article discusses the issue of assessing the similarity of algorithms and computer programs. We study the construction of a graph model of an algorithm based on the texts of programs in applied programming languages and also provides a method for comparing model data. The efficiency of the developed methodology is analyzed with examples of simple algorithms.

Ключевые слова: алгоритм, схема программы, граф потока управления, сравнение графов.
Keywords: algorithm, program diagram, control flow diagram, graph comparison

В современном информационном пространстве проблема плагиата занимает довольно значимое место. Кроме распространенного явления нарушения авторских прав на тексты (научные работы и т.д.) также существует проблема плагиата программ для ЭВМ.

Если для анализа сходства текстовой информации пока существуют решения, в случае программ возникают сложности, связанные с разнообразием языков программирования.

Программы для ЭВМ является одним из наиболее распространенных объектов интеллектуальной собственности. Алгоритмы не охраняются авторским правом, но могут быть защищены патентом на изобретение. Возможность сравнивать алгоритмы, реализованные на разных языках программирования, может помочь в решении патентных споров. Также для получения патента на алгоритм программы для ЭВМ необходимо наличие в нем новизны - для доказательства этого также полезным будет наличие методов оценки сходства оцениваемого алгоритма с потенциально похожими.

Таким образом, возникает проблема эквивалентности программ: даются две программы и ставится вопрос, вызывают ли они вычисления, приводящие к одинаковым результатам [4].

Программы являются объектами более сложной структуры, чем традиционные системы описания алгоритмов. Теория схем программ использует сложных реальных объектов более простыми абстрактными объектами [8].

Согласно теории схем программ [8], требования к модели устанавливаются следующим образом:

1. модель позволяет изучать свойства достаточно широких классов программ, а не отдельный конкретных программ;
2. сохраняет все интересующие исследователя свойства и особенности рассматриваемого класса программ;
3. позволяет игнорировать несущественные для данной проблемы свойства, например, синтаксические детали;
4. модель модифицируема и допускает нововведения, отслеживающие развитие языка программирования;

5. даёт возможность отделить разрешимые и неразрешимые свойства программ и классов программ;

6. удобно, если структура модели изобразительно подобна структуре программ, что даёт возможность привлекать на некоторых этапах исследований программистскую интуицию.

Кроме таких распространённых методов формального описания алгоритмов, как блок-схема и диаграмма Насси-Шнайдермана, для оценки сходства алгоритмов более целесообразно использовать диаграммы потока данных (data flow diagrams) и граф потока управления (control flow graph). Применение этих методов призвано упростить автоматический анализ и сравнение.

Диаграммы потока данных - нотация, предназначенная для моделирования информационных систем с точки зрения хранения, обработки и передачи данных. Цель данного представления - продемонстрировать, как входные данные превращаются в выходные.

Граф потока управления - совокупность всех возможных путей выполнения программы, представленный в виде графа. Это представление используется при анализе алгоритмов оптимизаторами для удаления недостижимых блоков кода обнаружения бесконечных циклов, и тому подобное.

В рамках данного исследования было принято решение использовать графы потока управления, представляя все возможные пути выполнения программы в виде ориентированного графа с петлями (также две вершины могут соединяться двумя разнонаправленными дугами).

Для построения графа необходимо привести программу к общему виду, не имеющему привязки к особенностям конкретной реализации. В общем случае данная процедура сводится к следующим этапам:

1. Лексический анализ – преобразование исходного текста программы в набор лексем, представленных в виде пары *номер класса* : *номер в классе.*

2. Синтаксический анализ – проверка соответствия текста программы спецификациям языка.

3. Построение графа потока управления – на основании результатов лексического анализа формируется ориентированный граф, демонстрирующий все возможные пути выполнения исследуемой программы.

4. Оптимизация графа – удаление недостижимых блоков, которые не влияют на результат. Таким образом, добавление в текст программы невыполнимых условий не повлияет на его схожесть с эталоном. Данная процедура, как правило, производится транслятором на этапе оптимизации кода.

5. Сравнение графов – на данном этапе необходимо сравнить между собой графы управления и выразить их схожесть в числовом эквиваленте.

В целях упрощения можно предположить, что поступаемые на вход исходные тексты программ являются корректными (соответствуют спецификации языка) и не содержат недостижимых блоков. Это позволит исключить этапы 2 и 4 и тем самым значительно упростить задачу.

На этапе лексического анализа происходит преобразование исходного текста программы в набор лексем, представленных в виде пары *номер класса:номер в классе*. Различаются следующие классы лексем:

- Ключевые (зарезервированные) слова
- Разделители
- Операции
- Литералы
- Идентификаторы

Поскольку действует предположение, что входные данные заведомо корректны, в случае, если лексему не удаётся отнести ни к одному из вышеперечисленных классов, она считается идентификатором и заносится в таблицу идентификаторов. Данный подход значительно упрощает построение таблицы идентификаторов, поскольку в различный языках синтаксис объявления переменных, функций и др. сильно различается.

При построении графа потока управления в исходном тексте программы, представленном в виде упорядоченного набора лексем, выделяются следующие конструкции:

- Условный оператор (if..else)
- Оператор ветвления (switch)
- Цикл с счётчиком (for)
- Цикл с предусловием (while)
- Цикл с постусловием (do..while)

Выделение управляющих конструкций в тексте программы осуществляется по ключевым словам, характерным для конкретного языка. В результате данной процедуры получаем ориентированный граф, отображающий варианты выполнения исследуемого алгоритма без привязки к особенностям языка программирования, используемого для реализации. В частности, циклы с предусловием и счётчиком (рис. 1) а также условный оператор и ветвление с одним вариантом (рис. 2) в графовом представлении будут выглядеть одинаково.

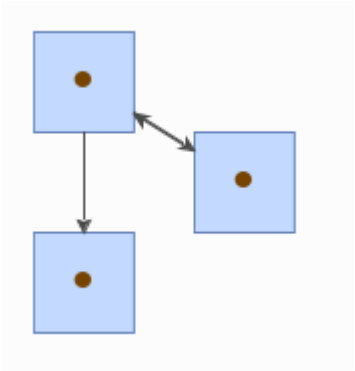


Рис. 1 — Цикл с предусловием

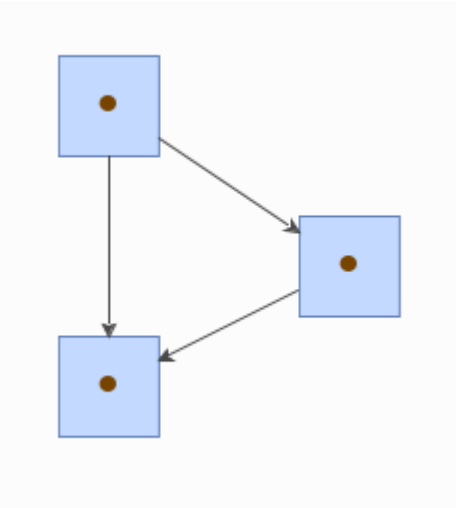


Рис. 2 — Условный оператор

Таким образом, в случае, если замена данных конструкций друг на друга не приведёт к различиям при выполнении программы, её анализ также покажет идентичность.

Сравнение графов выполняется на основе метода поиска в ширину: алгоритм перечисляет все достижимые из S вершины в порядке возрастания расстония от S [6]. Просмотр вершин производится одновременно для обоих исследуемых графов (A, B) и на каждом шаге алгоритма совпадающие вершины помечаются (рис. 3). В случае, если количество вершин, достижимых из текущей, различны для графов A и B, предпочтение отдаётся тем вершинам, для которых количество рёбер, выходящих из них, отличается минимально. Таким образом гарантируется, что непомеченной останется именно та часть первого графа, которая не имеет аналога во втором графе.

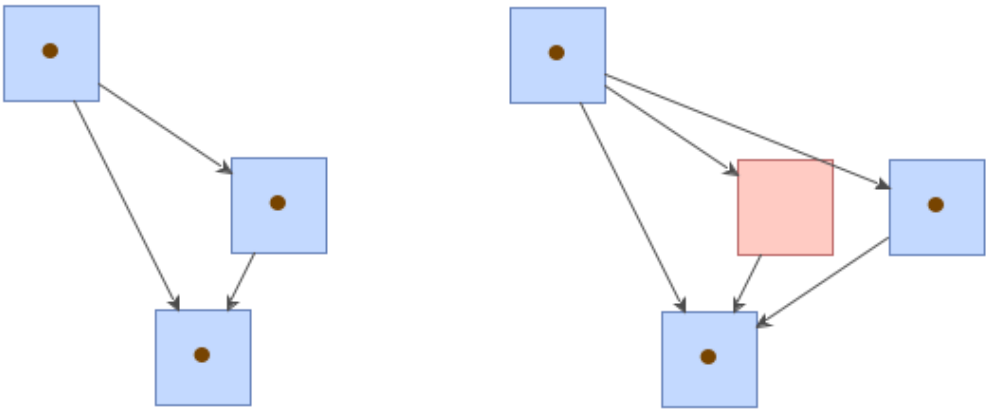


Рис.3 — Пример сравнения графов

Далее сходство графов выражается в числовом эквиваленте по формуле:

$$Match(A,B) = \frac{|\acute{A}| + |\acute{B}| - 2}{|A| + |B| - 2} \times 100$$

Где A, B – исследуемые графы, $|A|, |B|$ – количество вершин в соответствующих графах, а $|\acute{A}|, |\acute{B}|$ – количество помеченных вершин. В результате получаем число в диапазоне 0..100 выражающее сходство в процентном соотношении.

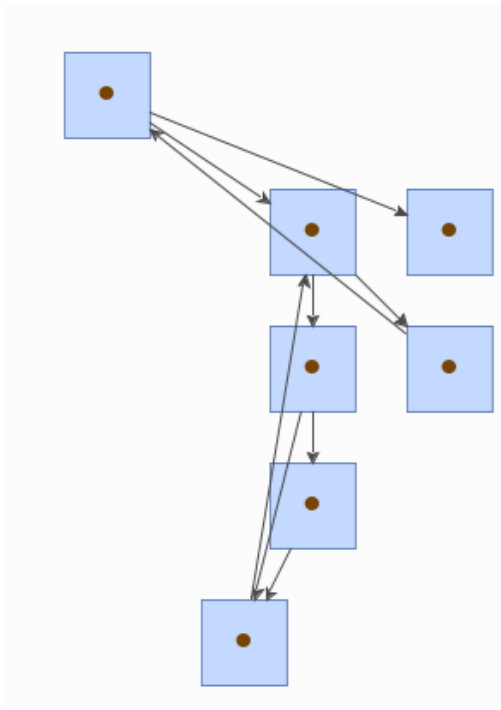
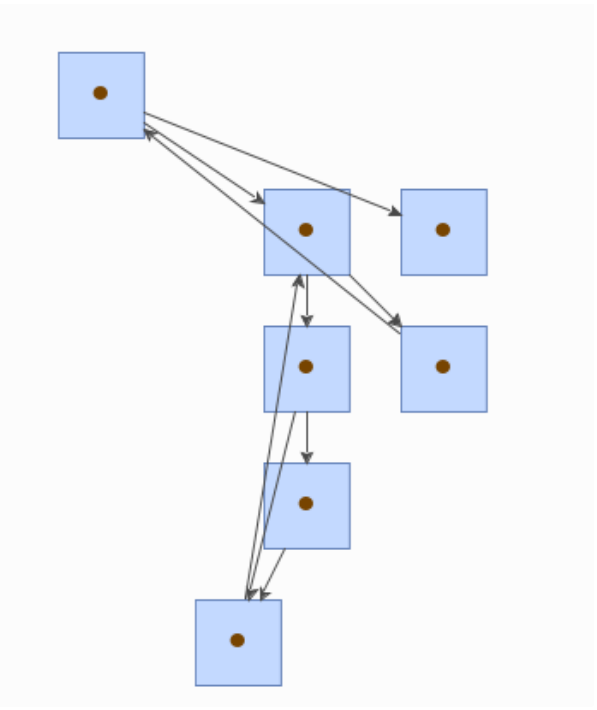
В качестве проверки корректности разработанной методики было проведено два эксперимента. В первом случае для сравнения берётся две реализации алгоритма сортировки пузырьком на языках C и Pascal. Текст программ был разобран на набор лексем, а затем по данным лексемам построены графы потока управления (табл. 1).

Табл. 1

Сравнение различных реализаций сортировки пузырьком

C	**Pascal**

long c, d, t;
 	
for (c = 0 ; c < n - 1; c++)
{
for (d = 0 ; d < n - c - 1; d++)
{
if (list[d] > list[d+1])
{
t = list[d];
list[d] = list[d+1];
list[d+1] = t;
}
}
}
	var
	n:integer;
	newn:integer;
	i:integer;
	begin
	n := high(a);
	while (n <> 0) do begin
	newn := 0;
	for i := 1 to n do begin
	if a[i - 1] > a[i] then begin
	swap(a[i - 1], a[i]);
	newn := i;
	end;
	end;
	n := newn;
	end;
	end;

В результате, как и ожидалось, было подтверждено 100% сходство.

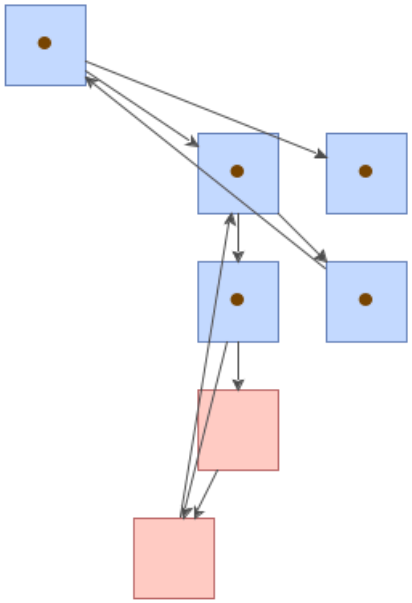
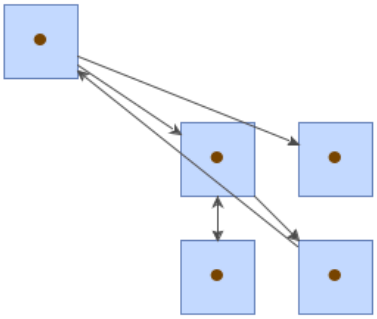
Во втором случае для сравнения берутся два различных алгоритма, реализованных на языке Pascal: сортировка пузырьком и сортировка вставками [3].

Табл. 2

Сравнение различных алгоритмов

Сортировка пузырьком	**Сортировка вставками**

n := high(a);
while (n <> 0) do begin
newn := 0;
for i := 1 to n do begin
if a[i - 1] > a[i] then begin
swap(a[i - 1], a[i]);
newn := i;
end;
end;
n := newn;
end;
	for j := 2 to A.length do begin
	key := A[j];
	i := j-1;
	while (i>0 and A[i]>key) do begin
	A[i + 1] = A[i];
	i := i - 1;
	end;
	A[i+1] := key;
	end;

Как можно заметить, алгоритм сортировки пузырьком, по сравнению с сортировкой вставками, содержит «лишний» условный оператор, что также видно на графе. В результате программного анализа было определено сходство в 80%.

Для автоматического сравнения алгоритмов была разработана программа, выполняющая лексический анализ исходных текстов программ, а также построение и сравнение графов потоков управления. На рис. 4 представлен внешний вид главного окна. Программа реализована на языке Kotlin и предназначена для выполнения на ЭВМ в среде Java Runtime Environment

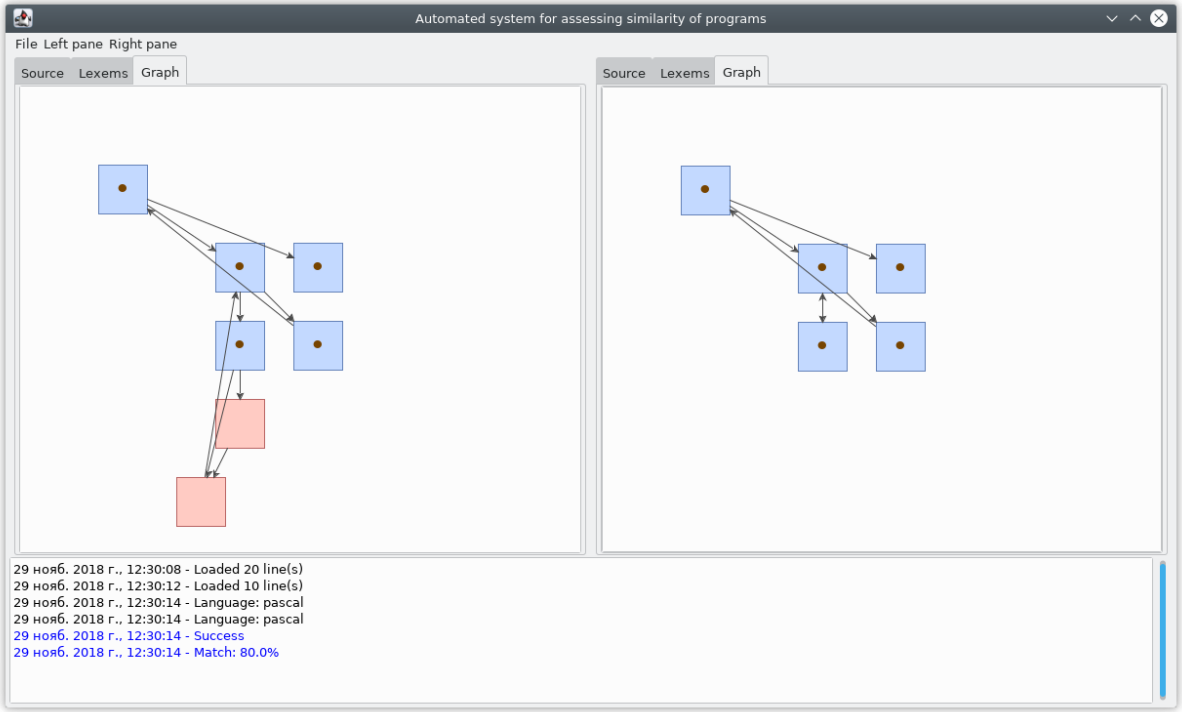


Рис. 3 Внешний вид программы

Таким образом, использование графов потока управления позволяет, абстрагировавшись от конкретной реализации алгоритма с использованием определенного языка программирования, оценивать сходство алгоритмов программ для ЭВМ. Приведённая методика сравнения алгоритмов призвана упростить патентные споры и защиту интеллектуальной собственности. Однако стоит отметить, что данный подход позволяет сравнивать только алгоритмы, реализованные на императивных языках.

Список литературы:

1. Chandra A.K., Manna Z. Program Schemes with Equality
2. Robert Gold: Control flow graphs and code coverage
3. Sedgewick, Robert. Algorithms In C: Fundamentals, Data Structures, Sorting, Searching, Parts 1-4
4. Дал У., Дейкстра Э., Хоор К. Структурное программирование М.: Мир, 1975. - 245 с.
5. Камерон П. Теория графов. Теория кодирования и блок-схемы — М.: Наука, 1980. - 140с.
6. Кормен, Лейзерсон, Ривест, Штайн. Глава 22. Элементарные алгоритмы для работы с графами // Алгоритмы: построение и анализ. — 2-е. — М: Вильямс, 2005. — С. 609 - 643. — 1296 с.
7. Кормен Т., Лейзерсон Ч., Ривест Р. Глава 22. Элементарные алгоритмы для работы с графами // Алгоритмы: построение и анализ(второе издание). — М.: «Вильямс», 2005. — С. 622—632.
8. Котов В., Сабельфельд В. Теория схем программ — М.: Наука, 1991. - 248 с.
9. Оре О. Теория графов — М.: Наука, 1980. - 336 с.
10. Сабельвельд В.К. Функциональная эквивалентность сквозных схем // Тез. докл. II Всесоюзн. конф. по прикладной логике. - Новосибирск: ИМ СО АН СССР, 1988 - с.202-204