OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ДЕТЕКТИРОВАНИЕ ИЗМЕНЕНИЙ В XML-ПОДОБНЫХ СТРУКТУРАХ ДОКУМЕНТОВ

Глоба Л.С., Молчанов Ю.Н.

Национальный технический университет Украины «Киевский политехнический интститут», г. Киев, Украина

lgloba@its.kpi.ua

molchanov@ukr.net

В работе представлен метод детектирования изменений в XML- подобных структурах документов, в основу которого положен метод генетического локального поиска для детектирования изменений, что обеспечивает удовлетворительное время детектирования изменений при сохранении достоверности полученных результатов. Предложенный метод обеспечивает удовлетворительную временную сложность решения задачи БЛП: при детектировании изменений в деревьях среднего размера (10-100 элементов) удалось значительно сократить минимальное время детектирования изменений по сравнению с точными методами.

Ключевые слова: детектирования изменений в XML- подобных структурах; древовидные структуры; метод генетического локального поиска; задача поиска хорошего соответствия.

Введение

В среде Internet в настоящее время распространена такая схема взаимодействия между публикующими информацию и потребителями информации, когда каждый человек, имеющий доступ к сети Internet, в любое время может выступать в роли как публикующего, так и потребителя. Эта ситуация приводит к постоянному неконтролируемому обновлению информации и проблемы отслеживания изменений на определенном ресурсе или по определенной тематике во всей сети Internet. В таких условиях от пользователя требуется постоянно обрабатывать большой объем данных только для поиска полезной информации. Вместе с тем, общий объем информации в сети Internet будет постоянно увеличиваться, поэтому пользователь вынужден постоянно увеличивать время для поиска необходимой информации.

Данная проблема решается в рамках разработки событийно-ориентированных систем публикации/подписки, обеспечивающих полное разделение во времени, пространстве и синхронизацию между тем, кто публикует, и подписчиком. В таких системах потребитель сообщает о своем интересе в определенной информации или изменениях на определенном ресурсе, и проблема отслеживания, детектирования и оповещения об изменениях возлагается на соответствующие функции системы. Однако в настоящее время разработанные системы публикации/подписки характеризуются следующими недостатками [Chakravarthy et al., 2004], [Coruscio et al., 2002], [Chand et al., 2004]:

– недостаточной гибкостью в определении информационного объекта – невозможность определения отдельного объекта или группы объектов на веб-странице, на которой необходимо отслеживать изменения, недостаточное удобство представления информации пользователю, ограниченный набор значений параметров, который можно передать в запросе, возможность отслеживание только определенного типа изменений (например, невозможность определения вставки или обновления узлов в документе);

– некоторые системы не ориентированы на среду Internet и требуют заранее заданной информации о топологии системы;

– отсутствие поддержки структурированных источников информации (HTML, XML);

– необходимость в установке дополнительного программного обеспечения на стороне публикующего и потребителя;

– некорректное или неудобное отображение найденных изменений в документе и т.д.

Основным узлом веб-ориентированной системы публикации/подписки считается модуль отслеживания изменений в иерархически структурированной информации, входные данные

которого задают в виде XML-документа любой структуры, а алгоритм определения изменений работает в условиях неопределенности структуры (DTD, XML-schema) документов и должен поддерживать корректное определение добавления, удаления, обновления узла, а также изменение структуры, включая и изменение порядка следования узлов одного уровня, что не всегда реализовано в существующих системах публикации/подписки.

1. Анализ подходов к определению изменений в XML-документах

Система публикации/подписки требует сравнительно быстрого алгоритма детектирования изменений, обеспечивающего минимальную задержку определения и доставки уведомления об изменении определенной информации.

В настоящее время не существует эффективного механизма, который мог бы корректно определять и удобно представлять изменения в веб-страницах. Существующие решения, представленные в [changedetection.com, followthatpage.com, changedetect.com] основаны на детектировании изменений в текстовых файлах и некорректно отражают изменения в структуре документов.

Анализ существующих алгоритмов детектирования изменений в XML-документах позволил определить основные типы данных алгоритмов: алгоритмы, вычисляющие минимальную последовательность редактирования всех возможных редакционный скриптов [Chawathe, et al., 1997], [Chawathe, et al., 1998], [Алексеев и др., 2009], алгоритмы, основанные на хеш-подписях [Khan et al., 2003], [Jyoti et al., 2005], [Khandagale et al., 2010] и семантические алгоритмы [Flesca et al., 2007].

Основной задачей метода детектирования изменений является поиск соответствия между узлами двух древовидных структур - «хорошего соответствия», которое согласно работам [Chawathe, et al., 1997], [Chawathe, et al., 1998] предусматривает наличие такого соответствия между узлами древовидных структур, когда превращение старого дерева в новое требует минимального количества элементарных операций редактирования. Учитывая, что перед системой детектирования изменений ставится задача отображения произведенных изменений, а не задача преобразования старой древовидной структуры в новую структуру, то понятие «хорошего соответствия» было пересмотрено и сформулировано следующим образом. Хорошим соответствием между двумя древовидными структурами называется такое соответствие, когда суммарный критерий соответствия между всеми парами соответствующих узлов в двух деревьях является максимальным. Здесь под критерием соответствия понимается численная величина, показывающая степень соответствия двух узлов по критериям сходства содержания, атрибутов и положения в иерархии дерева.

Существующие методы сравнения древовидных структур основываются на сравнении всех пар узлов двух деревьев на соответствующих местах в иерархии дерева. Если положение узлов совпадает, то проводится сравнение содержания узлов. Остальные узлы, которые не совпали, считаются добавленными или удаленными соответственно. К данным методам относится метод сравнения узлов на основе хэш-подписей, а также семантические методы. К недостаткам этих методов следует отнести рассмотрение только одного возможного соответствия между деревьями, когда структура дерева не меняется. Тогда как другие варианты соответствия, учитывающие все возможные операции редактирования над узлами деревьев, могли бы привести к более корректному результату. Данные методы имеют значительные ограничения и не подходят для решения задачи детектирования изменений в XML-документах.

При выборе математического аппарата для решения поставленной задачи был также проанализирован математический аппарат теории графов, так как древовидная структура в общем виде представляет собой связанный ациклический граф.

Можно выделить два основных подхода при сравнении графов - поиск максимального изоморфного пересечения и поиск минимальной последовательности редактирования.

Существующие методы поиска максимального изоморфного пересечения проводят поиск изоморфного пересечения, которое имеет максимальное количество вершин, определяющих меру сходства между графами. Основной особенностью методов поиска минимальной последовательности редактирования является предположение о наличии «хорошего соответствия» между древовидными структурами.

В существующих системах хранения структурированных документов как структура документа может изменяться, так и содержание в узлах дерева, причем как в случае изменившейся структуры, так и при той же самой структуре. Это требует решения о «хорошем соответствии» на основании критерия, который бы учитывал все возможные изменения в документах, а не предполагал изменение только содержания или только структуры.

Кроме того, два графа называются изоморфными, если существует взаимно-однозначное соответствие между их вершинами и ребрами, которое сохраняет смежность и инцидентность. Понятие изоморфности предусматривает наличие большего количества ограничений при поиске соответствия графов, чем поставленная задача поиска изменений между древовидными структурами, которая

предусматривает возможность изменения структуры дерева, удаление и добавление узла и т.п., исключает возможность рассмотрения соответствующих деревьев как изоморфных графов. Таким образом, существующие методы поиска максимального изоморфного пересечения не могут быть использованы для решения поставленной задачи.

Проведенный анализ показал, что задача поиска хорошего соответствия должна рассматриваться как задача поиска соответствия между связанными ациклическими графами, и на данный момент не существует метода сравнения таких структур. Это обусловлено тем, что при преобразовании древовидных структур не будут сохранены такие свойства, как смежность, инцидентность узлов и ребер, а также изоморфность графов, так как в рассматриваемой предметной области при преобразовании заданных деревьев могут быть изменены как отдельные узлы (содержание, атрибуты), так и вся структура дерева.

Рассмотренные алгоритмы имеют ряд недостатков:

– сосредоточены на производительности, временной сложности операций детектирования изменений, а также оптимизации полученных «дельт», но инструментарию и моделям для эффективного внедрения данных алгоритмов в системах публикации/подписки на практике уделяется мало внимания;

– не поддерживают отслеживание всех возможных типов изменений в структурированных документах, поэтому некорректно отражают результирующие изменения;

– не поддерживают отслеживание изменений в условиях неопределенности структуры (DTD, XML-schema) документов;

– обладают слишком большой временной сложностью для анализа XML документов большого объема.

Таким образом, можно сделать вывод, что задача поиска «хорошего соответствия» между древовидными структурами на данный момент требует разработки алгоритма детектирования изменений в древовидных структурах в условиях неопределенности структуры дерева. Такой алгоритм даст возможность разработки системы публикации/подписки, которая позволит уменьшить количество отсылаемой информации об изменениях в документах конечному пользователю, предоставляя ему только данные о проведенных изменениях и только после того, как они реально сделаны.

2. Метод определения изменений в древовидных структурах

Метод детектирования изменений между древовидными структурами состоит из следующих шагов:

1. Введение исходных данных в виде начальной и обновленной древовидных структур.

В задаче детектирования изменений в древовидных структурах на основе веб-страниц формируют первоначальную и обновленную древовидные структуры, которые далее рассматривают как входные объекты для задачи детектирования изменений [Flesca et al., 2007].

2. Индексирование древовидных структур, необходимое для учета порядка узлов слева направо в результирующем дереве и для отслеживания местоположения узла в иерархии дерева.

Индексирование проводится по правилам, приведенным в [Lim et al., 2001], и представляется в виде дополнительного атрибута в каждом узле древовидных структур.

3. Расчет критериев соответствия и поиск изменений проводиться для узлов древовидных структур.

Данный этап необходим для исследования характеристики соответствия между двумя узлами и ее выражения через численные параметры.

4. Определение значимых для поиска изменений узлов.

На этом этапе происходит поиск значимых для пользователя узлов в полученной древовидной структуре, которые будут участвовать в алгоритме поиска изменений. К таким узлам можно отнести узлы дерева с текстовым содержанием, представляющим интерес для подписчика, они обычно представлены в виде листьев в результирующем дереве. Это обусловлено особенностями структуры веб-страниц, которые представлены на языке XML. Это определение не является ограничением для данного метода, так как его можно использовать и для внутренних узлов дерева. В рамках использования метода детектирования изменений для веб-страниц, такое допущение приемлемо.

5. Расчет параметров соответствия листьев древовидных структур, выбранных на предыдущем шаге.

В рассматриваемой математической модели, листья обоих деревьев сравнивают по критериям содержание, перечню атрибутов и положению узла в структуре дерева. Использование такой модели позволяет исследовать максимальное количество характеристик отдельного узла и находить наиболее точный показатель соответствия между узлами. Эти параметры будут рассчитываться попарно для всех выбранных узлов в старой и новой версии дерева.

6. Создание матрицы интегральных критериев соответствия для всех выбранных узлов в обеих версиях XML дерева.

На этом этапе завершается процесс исследования математической модели дерева и ее результатом является матрица интегральных критериев соответствия узлов дерева. Эта матрица показывает степень сходства между листьями деревьев в виде множества полученных значений.

7. Формулировка задачи линейного программирования на основе матрицы интегральных критериев соответствия.

На этом этапе происходит математическое моделирование задач поиска «хорошего соответствия» для листьев деревьев. Ограничениями этой задачи является необходимость нахождения целочисленного решения, поскольку параметр связности может иметь только два значения: 0 или 1. Результат решения этой задачи будет решением задачи хорошего соответствия начальной и обновленной древовидных структур. Если количество листьев в обоих деревьях не соответствует друг другу, то в процессе решения будет найдено, какие узлы были удалены или добавлены в новом дереве.

8. Решение задачи линейного программирования с помощью модифицированного метода генетического локального поиска.

В ходе решения задачи находят целочисленное решение, которое бы удовлетворяло всем поставленным условиям. Это решение позволит определить взаимное соответствие между узлами старой и новой версии.

9. Выполнение функции сравнения содержания в найденных соответствующих узлах на этапе решения задачи линейного программирования.

После нахождения пар соответствующих узлов в деревьях происходит сравнение текстового содержания в каждой паре. В данном случае можно использовать алгоритмы для сравнения неструктурированных текстов, которые представлены во многих работах, в том числе в [Myers, 1986]. Кроме сравнения текстового содержания соответствующих пар узлов, на этом этапе можно определить и изменение в структуре дерева относительно листьев дерева. Эти изменения могут соответствовать изменению местоположения узла с текстовым содержанием в обновленной древовидной структуре.

Приведенный метод детектирования изменений можно описать в виде следующей блок-схемы, представленной на рис. 1.

Рисунок 1 – Блок-схема метода детектирования изменений между древовидными структурами

3. Постановка задачи поиска хорошего соответствия между древовидными структурами

Сформулируем задачу определения «хорошего соответствия» в общем виде для древовидной структуры.

Дано:

деревья $T_1 = T_1(a_i, n, R, AT, con(a_i)) \big| i = \overline{1,n}$

и $T_2 = T_2(a_j, n, R, AT, con(a_j)) \big| j = \overline{1,m}$,

где a_i – i-ый узел дерева,

n – количество узлов в дереве,

$R = \{r_i \big| i = 1..m\}$ – совокупность родительских узлов в дереве T_1, где m – количество родительских узлов, r_i – родительский узел i-го узла,

$AT = \{at_i \big| i = 1..n\}$ – совокупность атрибутов узлов в дереве, at_i – атрибут i-го узла;

$con(a_i)$ – содержимое i-го узла.

$$\langle C, X \rangle = c_{11} \cdot x_{11} + c_{12} \cdot x_{12} + c_{13} \cdot x_{13} + c_{14} \cdot x_{14} +$$
$$+ c_{21} \cdot x_{21} + c_{22} \cdot x_{22} + \ldots + c_{(n-1)m} \cdot x_{(n-1)m} + c_{nm} \cdot x_{nm},$$
$$a_{11}x_{11} + a_{12}x_{12} + \ldots + a_{1m}x_{1m} = b_1$$
$$\ldots$$
$$a_{n1}x_{n1} + a_{n2}x_{n2} + \ldots + a_{nm}x_{nm} = b_n$$
$$a_{(n+1)1}x_{11} + a_{35}x_{21} + \ldots + a_{(n+1)(m+1)}x_{n1} \le b_{n+1}$$
$$\ldots$$
$$a_{(n+m)1}x_{1m} + \ldots + a_{(n+m)(m+n)}x_{nn} \le b_{n+n}$$

или в сокращенной записи $Ax_{ij} \le B$, где матрица условий A и матрица ограничений B имеют значения:

$$A = \begin{pmatrix} 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & \dots & 1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 1 & \dots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

где C – матрица интегральных критериев CS соответствия узлов, $C = \{c_{ij}\}_{n \times m}$,

$$c_{11} = CS(A_1, B_1), c_{12} = CS(A_1, B_2),$$
$$c_{13} = CS(A_1, B_3), c_{14} = CS(A_1, B_4),$$
$$c_{21} = CS(A_2, B_1), c_{22} = CS(A2, B_2), \dots,$$
$$c_{nm} = CS(A_n, B_m),$$

x_{ij} – параметр связности узлов A_i и B_j.

Найти: $\max_{x \in B^n : Ax \le B} \langle C, X \rangle$.

При рассмотрении сложности решения поставленной задачи линейного программирования необходимо отметить, что данная задача предусматривает наличие решения только в булевых переменных, что требует решения задачи булева линейного программирования (БЛП). Эти задачи формулируются как задачи ЛП с дополнительным ограничением - принадлежности переменных множеству булевых переменных. В общем случае эта задача относится к задачам NP -сложности.

Для некоторых видов задач БЛП существуют точные методы решения: метод ветвей и границ, метод динамического программирования, метод Балаша; неточные: метод случайного поиска, локальной оптимизации, эвристические методы.

Вычислительная сложность имеющихся методов поиска решения задачи БЛП не удовлетворяет потребностей системы публикации/подписки из-за требования эффективного поиска соответствующих узлов для древовидных структур с большим количеством узлов, поэтому предложена модификация решения задачи БЛП с использованием генетических алгоритмов.

Генетические алгоритмы представляют собой итерационный процесс, конечной целью которого является получение оптимального решения.

Общую схему генетических алгоритмов проще понять, рассматривая задачу безусловной оптимизации. Которая формулируется как

найти:

$$\min\{F(x) \mid x \in X^n\}, X^n = \{0,1\}^n$$

Генетический алгоритм начинает работу по формированию начальной популяции x_0 - конечного набора допустимых решений задачи. Эти решения могут быть выбраны случайным образом, получены с помощью вероятностных «жадных» алгоритмов или другими методами. Выбор начальной популяции не имеет значения для сходимости процесса в асимптотике. Однако формирование «хорошей» начальной популяции (например, из множества локальных оптимумов) может заметно сократить время достижения глобального оптимума.

Основные операторы этого алгоритма: селекция, скрещивание и мутация.

Таким образом, с ненулевой вероятностью решение может перейти в любое другое решение, которое гарантирует в асимптотике получения точного решения задачи. Применение локального спуска позволяет генетическим алгоритмом сосредоточиться только на локальных оптимумах.

4. Модифицированный метод генетического локального поиска

Применим метод генетического локального поиска [Алексеева и др., 2007] для решения поставленной задачи.

Запишем матрицу C в следующем виде:

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & & & \\ c_{k1} & c_{k2} & \dots & c_{km} \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{pmatrix}.$$

На шаге 1 найдем допустимое решение для задачи булевого линейного программирования (БЛП). Для этого необходимо выполнить следующие действия.

Найти максимальное значение среди элементов матрицы C. Пусть максимальным элементом будет c_{k2}, тогда присвоим $x_{k2} = 1$. Из этого следует, что в матрице X:

$$x_{ki} = 0 \,|\, i \in [1,2) \cup (2,n], x_{j2} = 0 \,|\, j \in [1,k) \cup (k,n].$$

Поскольку данные величины в матрице X найдены, то в матрице C необходимо уменьшить зону поиска максимальных значений, убрав элементы, которые не находятся во втором столбце и k -й строке. Эти действия необходимо повторять, пока не будут найдены все элементы X^k.

На основании указанных действий сформулируем алгоритм для шага 1:

1. Поиск максимального элемента матрицы C – элемент $c_{kl} = \max(C^*) \,|\, C^* \subset \{c_{11} .. c_{nm}\}$.

2. Присвоить $x_{kl}=1$. Из этого следует, что $x_{ki}=0 \big| i\in[1,l]\cup(l,n], x_{j2}=0 \big| j\in[1,k]\cup(l,n]$.

Если не получено X^k, то ограничить зону поиска максимального значения $C^* \subset \{c_{11}..c_{nm}\}\cap(\{c_{k1}..c_{km}\}\cup\{c_{1l}..c_{n1}\})$ и повторить шаг 1.

Как результат работы алгоритма получаем решение X^k. Это решение на шаге 2 необходимо проверить на оптимальность с помощью метода генетического локального поиска. Найдем окрестность $G(X^k)$, на которой необходимо оптимизировать значения функции.

В данной работе под окрестностью $G(X^k)$ подразумевается множество всех допустимых решений, полученных с помощью метода генетического локального поиска.

Рассмотрим метод адаптации генетического алгоритма к нахождению окрестности X^k для оптимизации целевой функции. В качестве пространства поиска выступает пространство решений X^n, в качестве функции пригодности – целевая функция $\langle C, X\rangle$.

На этапе селекции используем алгоритм «лучший и случайный», когда в решении X^k будут выбраны две строки: x_i^k при $c_{ki}=\max\{c_{ij}\}\big| i=1..n,\ j=1..m$ и случайная строка x_j^k.

В качестве оператора скрещивания выбран одноточечный оператор. Пусть $x_{il}^k=1, x_{ip}^k=0, l\neq p$ и $x_{it}^k=1, x_{iq}^k=0, t\neq q$. Тогда после скрещивания: $x_{jt}^k=1, x_{jp}^k=0$ и $x_{il}^k=1, x_{iq}^k=0, t\neq q$. В процессе скрещивания два вектора «обменялись» единицами между собой.

В процессе мутации в каждой из строк единица может переходить на места, находящиеся в столбцах, которые содержат только 0. Среди всех вариантов выбирается наилучший по критерию максимума целевой функции.

После проверки, проведенной на шаге 2, будет получен оптимальный результат.

Сформулируем алгоритм для шага 2:

1. Для двух строк x_i^k и x_j^k в найденном на шаге 1 решении X^k, выбранных по алгоритму «полезный и случайный», заменим единицы на соответствующих местах.

2. Проводится мутация каждой строки по модифицированным правилам – с помощью поиска максимального значения целевой функции по окрестности 1-замена для всех возможных перестановок в рамках строк x_i^k и x_j^k. Находим значение $f(X^{k+1})$ для всех $X_1^{k+1}...X_{l-1}^{k+1}$. Если $f(X^k)\geq f(X^{k+1})$, то $f(X^k)$ – локальный оптимум. Если же $f(X^k)<f(X^{k+1})$, то заменяем X^k на X^{k+1} и выбираем новую мутацию до достижения $m-n$ шагов для каждой строки.

3. При достижении шага n выбирается максимальное значение $f\max(X^n)=\max\langle C,x\rangle=\max\{f(X^K)\}\big| k=1..n$. В противном случае переходим на шаг 1.

После работы алгоритма будет получено оптимальное решение задачи БЛП. Остановка алгоритма определяется количеством полученных локальных оптимумов, работа алгоритма останавливается при достижении n локальных максимумов.

Определим сложность работы алгоритма, оценивая численно временную сложность решения заданной задачи методом локальной оптимизации.

Ранее было принято без потери общности, что $n<m$. Пусть n – количество строк в матрицах C и X.

В алгоритме первого шага генетического локального поиска необходимо найти максимальное значение в каждой строке матрицы C. При этом на каждом шагу количество возможных вариантов будет уменьшаться на 1.

Таким образом, сложность задачи первого шага z_1 будет составлять разницу двух арифметических прогрессий S_m – суммы всех возможных вариантов перебора от 1 до m и S_{m-n} – суммы неиспользуемых вариантов от 1 до $(m-n)$:

$$R(z_1)=S_m-S_{m-n}=$$
$$=\frac{1+m}{2}\cdot m-\frac{1+(m-n)}{2}\cdot(m-n)=$$
$$=\frac{n^2+2nm+n}{2}.$$

Временная сложность второго шага обусловлена количеством перестановок в процессе мутации и составляет: $R(z_2)=S_{m-n}=2(m-n)$.

Таким образом, общая временная сложность решения задачи методом линейной оптимизации составляет:

$$R(z) = R(z_1) + R(z_2) =$$
$$= \frac{n^2 + 2nm + n + 2(m-n)}{2} =$$
$$= \frac{n^2 + 2nm - n + 2m}{2},$$

$$R(z) \infty (n^2 + m^2) \le 2n^2.$$

Предложенный алгоритм линейной оптимизации относится к неточным методам решения задачи БЛП, но он обеспечит нахождения локального максимума для поставленной задачи, который в большинстве случаев совпадает с глобальным максимумом, имея при этом более низкую временную сложность $O(n^2 + m^2)$.

5. Сравнение эффективности предложенного метода поиска соответствия между древовидными структурами и существующими аналогами

Для решения задачи поиска соответствия между XML документами использовался комбинаторный метод, сложность которого определяется как $O(n^2 + m^2)$, где n и m – количество листьев в начальной и обновленной древовидных структурах.

Для определения целесообразности такого подхода необходимо сравнить временную сложность поставленной задачи в этом подходе с временной сложностью алгоритмов, основанных на поиске минимальной последовательности редактирования, семантических алгоритмов и алгоритмов, основанных на хэш подписях.

Алгоритм Чанга и Шаша рассматривает два дерева для нахождения минимальной последовательности редактирования, может быть применим для XML документов. Условия этого алгоритма следующие. Иерархически структурированная информация может быть представлена как упорядоченные деревья – деревья, в которых потомкам каждого узла присвоен порядок. Рассматриваются дерева, в которых каждый узел имеет метку и значение, подразумевается, что каждый узел дерева имеет уникальный идентификатор. Минимальная временная сложность этого алгоритма определяется, как $O(p^2)$ [Wang et al., 2003], где $p = n + m$ – общее количество узлов в двух деревьях. Тогда сложность алгоритма равна: $O((n+m)^2)$.

Семантические алгоритмы рассматривают XML документ как структуру XML тегов. При сравнении учитывается иерархия синтаксиса языка и его закономерности. С помощью этого алгоритма можно определить только перемещения узлов, и невозможно определить удаление и вставку узла. Поэтому этот алгоритм не всегда находит корректное соответствие между узлами XML документов. Сложность этого подхода [Lim et al., 2001]:

$$O(|X| \cdot |Y| \cdot (|X| \log |X| + Y \log |Y|)),$$

где X и Y – количество ветвей в семантической иерархии каждого из деревьев. Поскольку количество ветвей иерархии можно принять равным количеству листьев в каждом из деревьев, то сложность этого метода можно представить следующим образом: $O(m \cdot n \cdot (n \log n + m \log m))$.

В системе детектирования веб-страниц Коура рассматривается похожий с данной работой подход. Этот алгоритм рассматривает задачу нахождения соответствия между XML деревьями как задачу венгерского алгоритма. Сложность этой задачи в [Khoury et al., 2007] определена как $O((n+m)^3)$, что значительно больше, чем сложность предложенного алгоритма.

На рис. 2 показана сравнительная характеристика рассмотренных методов сравнения XML документов с помощью различных подходов.

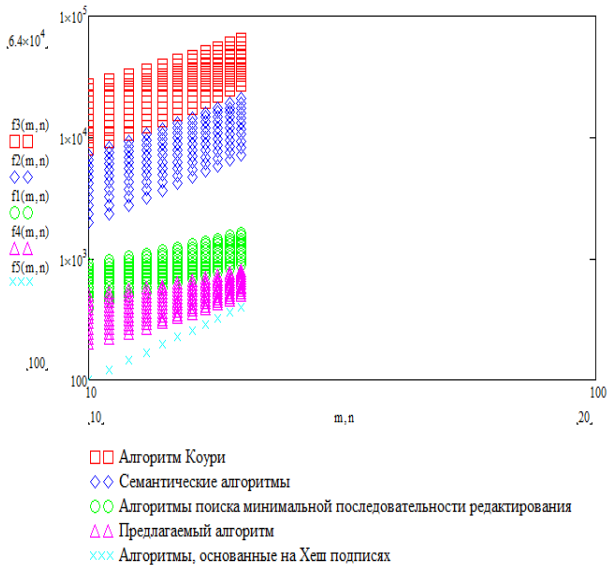

Рисунок 2 – Сравнительная характеристика рассмотренных методов

Найдем разницу между количеством операций, необходимых для выполнения предложенного алгоритма и самого эффективного из существующих, который удовлетворяет поставленным условиям – алгоритма Чанга и Шаша: $\Delta = (n+m)^2 - (n^2 + m^2) = 2nm$.

Эта разница будет существенная при увеличении значений n и m.

Ресурсом http://www.websiteoptimization.com/ было проведено исследование, в котором было вычислено среднее значение количества элементов страницы HTML в среде Internet в 2007 году, которое составило 592 элемента, также было показано, что эта величина непрерывно растет. Например, количество HTML элементов на

странице выросла с 2006 по 2007 год от 281 до 592 элементов.

Пусть исходный документ HTML содержит 592 элемента, а в обновленном документе были убраны теги и их общее количество составит 100 элементов. Найдем выигрыш в эффективности в данном случае:

$$\eta = \frac{(n+m)^2 - (n^2 + m^2)}{(n+m)^2} = \\ = \frac{(592+100)^2 - (592^2 + 100^2)}{(592+100)^2} \cdot 100\% = 24{,}7\%.$$

При сравнении предложенного метода решения задачи поиска соответствия между XML документами с другими алгоритмами было показано, что предложенный алгоритм является эффективным с точки зрения временной сложности. Это достигнуто за счет решения задачи поиска соответствия между XML документами как задачи булевого линейного программирования и ее эффективного решения с помощью метода генетического локального поиска.

Заключение

Проведенные исследования позволяют сделать вывод о возможности использования метода генетического локального поиска в методе детектирования изменений, который обеспечивает удовлетворительное время детектирования изменений при сохранении достоверности полученных результатов.

Получил дальнейшее развитие метод генетического локального поиска для решения задачи булевого линейного программирования с учетом особенностей применения его в предметной области детектирования изменений в древовидных структурах, что позволило обеспечить удовлетворительную временную сложность решения задачи БЛП: при детектировании изменений в деревьях среднего размера (10-100 элементов) удалось уменьшить минимальное время детектирования изменений по сравнению с точными методами с 384 с до 6,72 мс.

Эффективность предложенных модификаций была доказана на основе анализа вычислительной сложности предложенных и существующих методов – время детектирования изменений в деревьях малого размера (до 10 элементов) было уменьшено в модифицированном методе Балаша с 181 мс до 0.22 мс, а в модифицированном методе динамического программирования с 142 с до 0.01 мс.

Библиографический список

[Алексеев и др., 2009] Алєксєєв О. М. Система публікування/передплати на науково-технічні інформаційні ресурси / Алєксєєв М. О., Молчанов Ю. М., Алексєєв О. М. // Вісник Сумського державного університету, серія "Технічні науки", – Вип. 2. 2009. – С.7-14.

[Алексеева и др., 2007]Алексеева Е. В. Генетический локальный поиск для задачи о р–медиане с предпочтениями клиентов / Е. В. Алексеева, Ю. А. Кочетов // Дискретный анализ и исследование операцій, 2007, серия 2, том 14,№1.

[Chakravarthy et al., 2004] Chakravarthy S. A learning-based approach for fetching pages in WebVigiL / S. Chakravarthy, S. Sanka, J. Jacob // Proceedings of the 2004 ACM symposium on Applied computing. – 2004. – pp. 1725-1731.

[Chand et al., 2004] Chand R. XNET: a reliable content-based publish/subscribe system / R. Chand, P. Felber, S. Antipolis // Reliable Distributed Systems, 2004. – pp. 264-273.

[Chawathe, et al., 1997] Chawathe S. Meaningful change detection in structured data / S. Chawathe, Garcia-Molin // Proceedings of the ACM, SIGMOD International Conference on Management of Data, Tuscon, Arizona – 1997 - pp. 26–37.

[Chawathe, et al., 1998] Chawathe, S. Representing and querying changes in semistructured Data /Chawathe, S., Abiteboul,, S, Widom, // Proceedings of the International Conference on Data Engineering, Orlando, Florida. – 1998. – pp. 4–13.

[Coruscio et al., 2002] Coruscio M. Formal Analysis of Clients Mobility in the Siena Publish/Subscribe Middleware / M. Coruscio, P. Inverardi, P. Pelliccione // Technical Report, Department of Computer Science, University of Colorado – 2002. – pp. 1-18.

[Flesca et al., 2007] Flesca E. Efficient and affective Web change Detection / E. Flesca, S. Masciari // Proceedings of the International Conference on Data Engineering, Tuscon, Arizona – 2007. – pp. 4–13.

[Jyoti et al., 2005] Jyoti J. CX-DIFF: A Change Detection Algorithm for XML Content and Change Visualization for WebVigiL / J. Jyoti, A. Sachde, and S. Chakravarthy // Data and Knowledge Eng., vol. 52, no. 2, 2005.– pp. 209-230.

[Khan et al., 2003] Khan L. Change Detection of XML Documents Using Signatures / L. Khan, L. Wang and Y. Rao // Department of Computer Science University of Texas at Dallas, 2003.

[Khandagale et al., 2010] Khandagale H. A Novel Approach for Web Page Change Detection System / H. Khandagale, P. Halkarnikar // International Journal of Computer Theory and Engineering – 2010.

[Khoury et al., 2007] Khoury F. An Efficient Web Page Change Detection System Based on an Optimized Hungarian Algorithm / F. Khoury, R. El-Mawas, O. El-Rawas, E. Mounayar, H. Artail // IEEE Transactions on Knowledge and Data – Vol.19. – 2007. – pp.48-59.

[Lim et al., 2001] Lim S. An Efficient Algorithm to Compute Differences between Structured Documents / S. Lim, N. Yiu-Kai // Information theory, IEEE. – 2001. – pp. 171-180.

[Myers, 1986] Myers E. An O(ND) difference algorithm and its variations / E. Myers // Algorithmica, 1(2), 1986. – pp. 251-266.

[Wang et al., 2003] Wang Y. A fast change detection algorithm for XML documents / Y. Wang, D. DeWitt, X. Cai. // In International Conference on Data Engineering (ICDE'03). Citeseer. – 2003. – pp. 235-258.

[**changedetection.com**] http://www.changedetection.com/

[**followthatpage.com**] https://www.followthatpage.com/

[**changedetect.com**] http://www.changedetect.com]

CHANGES DETECTION OF XML DOCUMENTS

Globa L.S., Molchanov Y.N.

National Technical University of Ukraine 'Kyiv Polytechnik Institute', Kyiv, Ukraine

lgloba@its.kpi.ua

This paper presents a method for detecting changes in the XML-documents, which is based on genetic local search algorithm for the changes detection in the XML-documents to provide a satisfactory time of changes detection while saving the validity of the results. The proposed method provides a satisfactory time complexity: in changes detecting for the trees of medium size (10-100 members) there is reduced significantly the minimum changes detection time in comparison with exact methods.