Анализ проектной документации на основе поиска частых подграфов

А. А. Горячев[1], Н. Е. Новакова[2]
Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
avgoryachev@gmail.com, nenovakova@gmail.com

***Аннотация*. Рассмотрена проблема анализ проектной документации. Описаны различные способы представления документации. Представлена задача поиска частых подграфов. Предложен алгоритм поиска документации на основе поиска частых подграфов.**

Ключевые слова: документация; анализ документов; поиск частых подграфов; алгоритм AprioriGraph; алгоритм PatternGrowthGraph

I. Введение

При разработке новых проектов часто используются ранее реализованные проектные решения, что позволяет минимизировать затраты на разработку и прогнозировать возможные проблемы реализации проекта. Для оптимизации процесса проектирования прибегают к представлению проектной документации в электронном виде и её последующему анализу.

На практике для осуществления автоматизированного анализа проектной документации разрабатываются специальные системы и прикладные программы, реализующие разнообразные методы анализа данных. В частности, уделяется внимание обработке имеющейся информации с целью приобретения и новых, ранее неизвестных, но практически полезных данных. Благодаря структурированности проектной информации её обработка может быть осуществлена методами интеллектуального анализа данных (Data Mining), позволяющими извлекать ранее неизвестные зависимости из больших объёмов данных, которые впоследствии могут быть полезны для принятия решений на различных этапах проекта [1]. Результаты, полученные в ходе интеллектуального анализа проектной документации, впоследствии могут быть применены для классификации, кластеризации, поиска, сортировки и группировки информации и т. д.

II. Способы представления проектной документации

Проектная документация определяет архитектурные, технологические, конструктивные, а также инженерные решения, используемые при реализации проекта. Так как для реализации проекта могут потребоваться данные, представленные в различном виде, проектная информация разнородна, может быть представлена как в текстовом, так и в графическом виде. Так, в процессе проектирования могут использоваться чертежи, отчёты, конструкторские спецификации, планы, графические схемы, таблицы и др.

XML-технологии часто применяются как средства информационного обмена между приложениями САПР. В данном случае на основе конструкций языка XML обеспечивается построение модели предметной области. Такая модель строится в зависимости от структуры и семантики проектной информации. Так, при построении информационной модели решаются следующие задачи:

- идентификация понятий;
- организация понятий в иерархию классов;
- определение связей, множественности и ограничений;
- определение атрибутов для конкретизации понятий, связанных с объектами.

Таким образом, проектные документы в формате XML можно рассматривать как деревья с точки зрения теории графов.

В качестве объектов анализа могут выступать объём работ, состав материалов и потребность в них и др. Для анализа этих объектов производятся альтернативные расчёты, подтверждающие либо опровергающие правильность первоначальных расчётов. По результатам анализа составляются сметы, позволяющие минимизировать расходы материалов и времени на осуществление работ.

III. Задача поиска частых подграфов

Задача поиска частых подграфов (Frequent Subgraph Mining – FSM) является одной из задач интеллектуального анализа графов (Graph Mining) – задачей извлечения ранее неизвестной, практически полезной информации из данных, представленных в виде графов. Частыми называются подграфы, которые встречаются на данном множестве графов не реже заданного порога встречаемости, называемого минимальной поддержкой (minimum support) [2].

Задачу поиска частых подграфов (FSM) можно решать как на множестве графов, так и на одном большом графе.

Если поиск осуществляется на одном графе, минимальная поддержка означает, сколько раз подграф встречается в графе. Если поиск осуществляется на множестве графов, минимальная поддержка может также определять минимальное число графов, в которых встречается данный подграф. Алгоритмы, используемые для решения этой задачи, разнятся в зависимости от типа графов и ограничений на поиск частых подграфов.

Рассмотрим задачу поиска частых подграфов на множестве графов, где минимальная поддержка $\min\_sup$ означает минимальное отношение числа найденных графов, содержащих частый подграф, к общему числу графов во множестве.

Для каждого графа g во множестве размеченных графов $D=\{G_1, G_2, ..., G_n\}$ определены множества вершин $V(g)$ и рёбер $E(g)$. Функция разметки L назначает каждым вершине и ребру некоторые метки. Граф g является подграфом другого графа g', если граф g изоморфен графу g'. Частым подграфом в данном случае называется подграф, чья поддержка (частота встречаемости в графах) не меньше минимальной $\min\_sup$. Если подграф является частым, то все его подграфы также являются частыми [27].

Основные проблемы в алгоритмах поиска частых подграфов [28]:

1. Изоморфизм графов (определение, встречается ли данный подграф в других графах).

2. Перечисление всех частых подграфов без повторяющихся подграфов (дубликатов).

Поиск частых подграфов обычно выполняется в 2 шага.

Шаг 1. Формирование подграфов-кандидатов.

Формирование кандидатов обычно выполняется посредством наращивания кандидатов в ширину либо в глубину. На этом шаге подграфы также проверяются на изоморфизм, причём вычислительная сложность этой проверки достаточно высока (NP-полная задача) [2]. Поэтому важно найти способ эффективного формирования кандидатов без повторений (чтобы не проверять изоморфные графы повторно).

Шаг 2. Проверка поддержки каждого кандидата. На этом шаге определяется, достаточно ли часто встречаются найденные подграфы-кандидаты в наборе графов.

В зависимости от способа формирования кандидатов выделяют 2 подхода к решению задачи поиска частых подграфов [2]:

1. Алгоритмы, основанные на алгоритме Apriori (Apriori-based approach). Подход заключается в формировании и проверке кандидатов с использованием поиска в ширину (Breadth-First Search – BFS) подграфов в данном множестве графов. Своё название алгоритм получил от алгоритма Apriori, опирающегося на следующее свойство: поддержка любого набора объектов не может превышать минимальной поддержки любого из его подмножеств. В соответствии с этим свойством для поиска подграфов размером $k+1$ сначала надо найти все подграфы размером k, после чего 2 подграфа размера k можно объединить для формирования подграфа размера $k+1$. Таким образом, получится, что все подграфы частого графа также будут частыми, а проверять нечастые подграфы не понадобится, так как если граф нечастый, то и его подграфы нечастые.

2. Алгоритмы, основанные на наращивании наборов (Pattern growth approach). Подход наращивания наборов применяет стратегию поиска в глубину, где каждый найденный подграф рекурсивно расширяется, пока все частые подграфы не будут найдены.

IV. Анализ проектной документации

Анализ проектной документации подразумевает анализ проектных документов, представленных в формате XML, посредством поиска частых подграфов. XML-документы имеют структуру дерева (связного ациклического графа) и зачастую представляют всевозможные списки и перечни (перечни технологического оборудования и др.). В связи с этим объектом анализа являются XML-документы, имеющие структуру списка деревьев.

Алгоритмы поиска частых подграфов позволяют находить подграфы, встречающиеся в данном графе с заданной частотой. При анализе XML-документов в качестве графов рассматриваются деревья, подграфов – поддеревья, а поддержка означает частоту встречаемости поддеревьев в деревьях (отношение количества деревьев, содержащих поддерево, к числу всех деревьев). Таким образом, предметом анализа являются XML-поддеревья, которые встречаются в XML-деревьях чаще установленного значения. Далее под графами и подграфами будут подразумеваться их частные случаи – деревья и поддеревья соответственно.

Анализ производится по следующей схеме:

1. Загрузка XML-документа.

Исходными данными является текст проектного документа, представленный в формате XML. Документ должен представлять собой список объектов-деревьев, обёрнутый в некоторый корневой элемент (узел с глубиной 0). Узлы дерева представляют собой теги, атрибуты и содержимое узлов при построении деревьев и проведении анализа не учитываются. Для анализа берутся теговые последовательности (XML-деревья) с корнями на глубине 1. Корневой элемент документа при анализе не учитывается.

2. Задание минимального и максимального размеров подграфов и минимальной поддержки.

В качестве ограничений поиска задаются минимальный и максимальный размеры подграфов (число узлов в подграфах) и поддержка – отношение числа графов, в которых встречается подграф, к общему числу графов.

3. Запуск алгоритма анализа XML-документа.

В связи с особенностями строения деревьев как графов использование алгоритмов, основанных на алгоритме Apriori и наращивании наборов, для деревьев напрямую

является нецелесообразным. Например, если по свойству Apriori каждый подграф частого графа является частым, то у деревьев может возникнуть ситуация, когда поддержка поддерева будет меньшей, чем поддержка самого дерева, при наличии узлов с одинаковыми метками. Для разрешения этого конфликта вводится следующее ограничение: 2 поддерева, состоящих из узлов одинаковой разметки, связанных теми же отношениями, что и узлы в исходном дереве, считаются разными тогда и только тогда, когда у них разные корни. Стоит также отметить, что под поддеревьями подразумеваются те поддеревья, у которых отношения родитель-ребёнок между узлами сохраняются такими же, как и в исходном дереве.

Для решения задачи поиска поддеревьев в деревьях предлагается алгоритм, представляющий собой последовательное расширение деревьев, представленных в виде DFS-кодировок (Depth First Search – DFS) [3].

После завершения работы алгоритма поиска и формирования частых подграфов на выходе имеется список поддеревьев, составленных из XML-тегов, а также их поддержка. На этом процесс анализа документа завершается.

Поиск частых поддеревьев выполняется в 2 шага:

Шаг 1. Формирование поддеревьев-кандидатов.

Шаг 2. Проверка поддержки каждого кандидата.

Данные шаги выполняются в цикле по глубинам, от листьев к корню. Для эффективного формирования кандидатов деревья размечаются в соответствии с их обходом в глубину (Depth First Search – DFS), после чего используются их DFS-представления (DFS-кодировки) для последующего наращивания в глубину. DFS-кодировка представляет собой уникальное строчное представление дерева, полученное обходом дерева в глубину: при спуске от корня к листьям записываются коды узлов, при возврате к корню – некоторая метка возврата (например «^»).

Для установки порядка между узлами-братьями деревьев, при котором каждый ребёнок родительского узла имеет определённую позицию по отношению к своим братьям, деревья (и их DFS-кодировки) приводят к каноническому виду, присваивая каждому узлу некоторый индекс [3]. Представление деревьев в виде DFS-кодировок и приведение их к каноническому виду позволяют определить крайний правый путь дерева для предотвращения формирования дубликатов кандидатов при расширении.

Пусть даны 3 дерева (рис. 1) и поддержка 0,3 (поддеревья должны встречаться хотя бы в одном из трёх представленных деревьев).

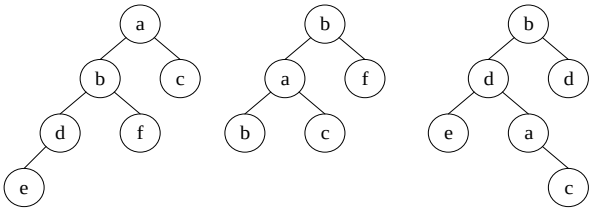

Рис. 1.

Деревья имеют следующие DFS-кодировки:

a, b, d, e, ^, ^, f, ^, ^, c, ^, ^

b, a, b, ^, c. ^, ^, f, ^, ^

b, d, e, ^, a, c, ^, ^, ^, d, ^, ^

Для предотвращения ситуаций нахождения поддеревьев одинаковой структуры, но с разным порядком перечисления потомков деревья приводятся к каноническому упорядоченному виду посредством присвоения каждому узлу номера следования при обходе в глубину, в качестве порядка следования узлов-братьев выбирается лексикографический порядок:

a[0], b[1], d[2], e[3], ^, ^, f[4], ^, ^, c[5], ^, ^

b[0], a[1], b[2], ^, c[3], ^, ^, f[4], ^, ^

b[0], d[1], a[2], c[3], ^, ^, e[4],^, ^, d[5], ^, ^

Различия в кодировках деревьев в каноническом и неканоническом видах можно заметить на примере дерева № 3. Данные деревья представляют собой связанные ациклические графы, составленные из узлов, соединённых рёбрами. Число узлов в дереве определяет его размер. В деревьях узлы связаны отношениями родитель-ребёнок: если узел *p* – родитель узла *c*, то существует ребро, ведущее из *p* в *c*. Родительские узлы могут как иметь, так и не иметь детей. Число детей некоторого родительского узла называется степенью узла. Дети одного и того же родителя называются братьями (либо сёстрами). Узлы без детей (узлы степени 0) называются листьями. Узел без родителя называется корнем.

Так как узлы в деревьях связаны, между двумя узлами можно построить путь – множество рёбер, связывающих эти 2 узла. Благодаря свойству ацикличности пути между двумя узлами в дереве уникальны.

Одной из характеристик деревьев является глубина дерева – длина самого длинного пути от корня к листьям. При этом глубиной узла считается число рёбер между корнем и этим узлом. На рис. 2 представлены кандидаты размерами 1 и 2 для деревьев на рис. 1. Формирование кандидатов начинается с разбиения исходных деревьев на поддеревья размерами 1 и 2.

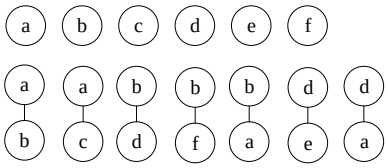

Рис. 2.

Такой шаг формирования кандидатов обоснован тем фактом, что если корень некоторого дерева имеет *n* детей, то дерево может быть разбито на *n* поддеревьев с тем же корнем и одним ребёнком. Полученные деревья будут иметь размер не меньший двух и далее могут быть разделены на 2 поддерева: одно из них будет состоять из двух узлов и иметь тот же корень, что и разбиваемое дерево, и ребёнка-лист, а второе – из оставшихся узлов (без корня разбиваемого дерева) с корнем в листе первого

поддерева. Рекурсивное повторение такого разбиения позволит получить множество поддеревьев размерами 1 и 2 для исходных деревьев (рис. 3). Причём повторение аналогичных действий в обратную сторону (расширение) позволит получить подграфы всех возможных размеров, а проверка поддержки – сократить перебор подграфов для выделения частых подграфов из всего множества графов.

После формирования кандидатов размера 1 и 2 проверяется их поддержка. Удовлетворяющие требованиям кандидаты считаются частыми и участвуют в дальнейшем формировании кандидатов.

Деревья больших размеров формируются последовательным расширением поддеревьев, начиная с поддеревьев размеров 1 и 2. Расширение поддеревьев обеспечивается двумя операциями: комбинированием и соединением – и начинается снизу вверх (от листьев к корню). Поэтому для начала выбираются кандидаты размерами 2, корень которых имеет наибольшую глубину. На рис. 3 представлено разбиение дерева на поддеревья размерами 1 и 2.

Таким образом, общая схема алгоритма поиска частых поддеревьев следующая:

Вход:

- множество деревьев $\{T_n\}$, их канонические DFS-представления,
- минимальное значение поддержки min_sup

Выход:

- список частых поддеревьев S

Метод:

1) Определение всех частых поддеревьев размером 1 и 2 из $\{T_n\}$
2) Добавление частых поддеревьев размером 1 и 2 в список S
3) Установить CurrentDepth = максимальная глубина $\{T_n\} - 1$

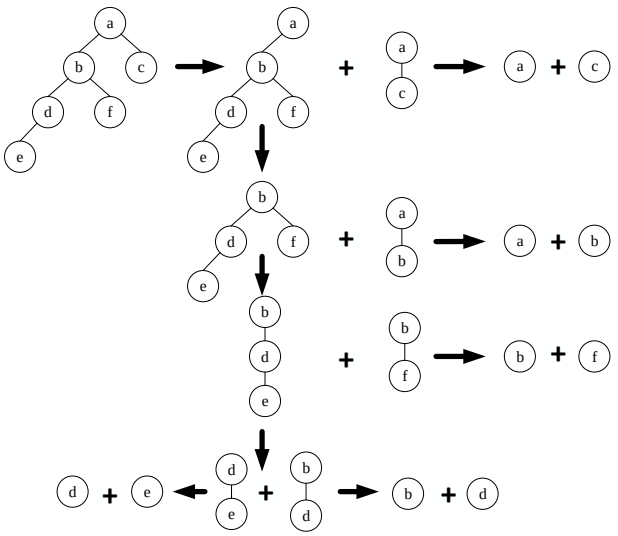

Рис. 3.

4) Определение всех частых поддеревьев размером 1 и 2 из $\{T_n\}$
5) Добавление частых поддеревьев размером 1 и 2 в список S
6) Установить CurrentDepth = максимальная глубина $\{T_n\} - 1$
7) пока CurrentDepth >= 0 выполнять комбинирование поддеревьев на глубине CurrentDepth:
8) для каждого частого поддерева $t_i \in S$, корень которого встречается на глубине CurrentDepth и имеет единственного потомка, выполнять
9) для каждого частого поддерева $t_j \in S$, имеющего тот же корень на глубине CurrentDepth, что и t_i, выполнять
10) комбинирование t_i и t_j с формированием нового кандидата t
11) если поддержка $\sup(t) \geq \min\_sup$, то
12) добавить t в S CurrentDepth
13) Соединение поддеревьев на глубине CurrentDepth:
14) для каждого частого поддерева $t_i \in S$ размером 2, корень которого встречается на глубине CurrentDepth, выполнять
15) для каждого частого поддерева $t_j \in S$, имеющего тот же корень на глубине CurrentDepth+1, что и лист t_i, выполнять
16) соединение t_i и t_j с формированием нового кандидата t
17) если поддержка $\sup(t) \geq \min\_sup$, то добавить t в S
18) Выход

Для апробации предложенного алгоритма было разработано приложение. Приложение написано на языке C# с применением Web-технологий на платформе ASP.NET по шаблону MVC. Приложение, позволяющее анализировать документ, представленный в формате XML, на предмет часто встречающихся тэговых структурных последовательностей (частых подграфов).

V. Заключение

Так как XML-документы имеют древесную структуру, а проектная документация может быть представлена в XML-формате, данный алгоритм позволяет проводить анализ проектной документации, представленной в XML-формате. Разработанный алгоритм может применяться для анализа проектной документации

Список литературы

[1] Han J., Kamber M. Data Mining: Concepts and Techniques// University of Illinois at Urbana-Champaign. 2006. 770 p.

[2] Zaki M. J. Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications // IEEE Transactions on Knowledge & Data Engineering. 2005 Vol. 17. P. 1021–1035.

[3] Zaki M.J. Efficiently Mining Frequent Trees in a Forest // Intern. Conf. Knowledge Discovery and Data Mining (SIGKDD'02), Edmonton, Canada, July 23–26, 2002.