

Maximal non-branching paths approach to solve (sub)graph isomorphism problem

Abstract

An approach based on maximal non-branching paths analysis to solve graph isomorphism problem is introduced.

An algorithm to solve the particular case of the problem of finding in a some graph A all subgraphs that are isomorphic to given graph B is proposed (only “inscribed” subgraphs can be found this way, not all of them). Here we shall name a subgraph of some given graph A as “inscribed” if (1) this subgraph is “glued” to other parts of A only by edges that connected to those vertices of this subgraph that are begin/ end ones of any max-length non-branching path of it, or (2) this subgraph is a separate connected component of the graph A.

The proposed algorithm of finding “inscribed” subgraphs is expanded to solve a whole problem of finding all (not only “inscribed”) subgraphs of a graph A that are isomorphic to given template graph B. To do so one should consider all edges of these graphs instead of their max-length non-branching paths.

These proposed approach and algorithm may be implemented to:

- directed or undirected graphs,
- graphs that have more than one connected components/ strongly connected components,
- graphs that contain multiple edges and multiple loops.

Key words

Subgraph isomorphism, subgraph isomorphism problem, graph isomorphism, “inscribed” subgraph, subgraph isomorphism algorithm, graph isomorphism algorithm.

С.А. Черноухов
Санкт-Петербург, РФ
chernouhov@rambler.ru

Проверка изоморфности двух графов и поиск изоморфных подграфов: подход, основанный на анализе максимально протяженных неразветвляющихся путей

Аннотация

Предложен подход к решению проблемы проверки изоморфности двух графов исходя из анализа их максимально протяженных неразветвляющихся путей.

На его основе предлагается подход и алгоритм решения частного случая задачи поиска в некотором графе A всех подграфов, изоморфных заданному графу B (а именно, поиск только «вписанных» подграфов), а также определяется само понятие «вписанного»

подграфа. «Вписанным» подграфом графа A здесь называется такой его подграф, который может быть «приклеен» к другим частям графа A только за счет ребер, инцидентных лишь граничным вершинам его (подграфа) неразветвляющихся путей максимальной длины (при этом граф A может содержать и иные компоненты связности).

Решение частного случая задачи поиска «вписанных» подграфов обобщается для поиска в графе A всех подграфов, изоморфных данному графу-образцу B . Для этого вместо рассмотрения их максимально протяженных неразветвляющихся путей необходимо рассматривать все их ребра.

Предложенные подход и алгоритм применимы:

- как для ориентированных, так и для неориентированных графов,
- для графов, содержащих более одной компоненты связности/ сильной связности,
- для графов, содержащих кратные (множественные) ребра и петли.

Ключевые слова

Изоморфизм графов, задача поиска изоморфного подграфа, подграф, «вписанный» подграф, алгоритм проверки изоморфности графов, алгоритм поиска изоморфного подграфа.

1. Общие сведения. Идея подхода, его преимущества и ограничения

Задача проверки гипотезы изоморфности двух графов, и, шире, – поиска в заданном графе A всех подграфов, изоморфных некоторому данному графу B , актуальна для различных областей, в т.ч. для хемоинформатики и вычислительной химии (поиск химических соединений по «шаблону»), решения задачи «сопоставления с образцом», и т.д.

Классическим подходом к решению данной задачи является «умный перебор» возможных соответствий вершин графа-образца B и графа A , в котором идет его поиск. Основной задачей при этом является максимально возможно более раннее отсечение заведомо неподходящих вариантов, для чего вводятся различные необходимые условия и ограничения, а также может проводиться различная предварительная обработка исходных данных.

Основа классического подхода была заложена Дж. Ульманом в 1976 году [3]; впоследствии данный алгоритм был существенно обновлен его автором [4]. Классический подход также получил свое развитие в работах Корделлы и соавторов [2] (алгоритм VF2), Бонници и соавторов [1] (алгоритм RI), и др.

В то же время, к решению задачи установления изоморфизма двух графов и, отчасти, поиска подграфа в графе по образцу можно подойти и с другой стороны. Действительно, если **два графа изоморфны друг другу**, то для любого представления первого графа в виде *последовательности максимально протяженных неразветвляющихся путей (maximal non-branching paths)* всегда существует соответствующее ему такое же представление второго графа, и при этом:

- для ориентированных графов соответствующие друг другу пути будут сопоставлены,

- степени соответствующих друг другу вершин для неориентированных графов (а для ориентированных – количества входящих и исходящих ребер соответственно) будут равны,
- при «совмещении» таких представлений в результате мы будем иметь соответствие вершин первого и второго графов.

Пример. Граф $A = \{1 \rightarrow 2, 1 \rightarrow 6, 4 \rightarrow 5, 5 \rightarrow 1, 3 \rightarrow 3\}$. Граф $B = \{3 \rightarrow 4, 3 \rightarrow 5, 1 \rightarrow 2, 2 \rightarrow 3, 6 \rightarrow 6\}$
 PathsA (*maximal non-branching paths of A*): $1 \rightarrow 2, 1 \rightarrow 6, 4 \rightarrow 5 \rightarrow 1, 3 \rightarrow 3$
 PathsB (*maximal non-branching paths of B*): $3 \rightarrow 4, 3 \rightarrow 5, 1 \rightarrow 2 \rightarrow 3, 6 \rightarrow 6$
 Соответствие вершин: $1(A):3(B), 2(A):4(B), 6(A):5(B), 4(A):1(B), 5(A):2(B), 3(A):6(B)$.

Таким образом, задачу проверки изоморфизма двух графов можно решить нахождением таких соответствующих друг другу путей и, затем, проверкой равенства друг другу матриц смежности, построенных исходя их сохранения порядка вершин в полученных последовательностях путей (при этом каждая вершина добавляется в порядок следования единожды, при ее первом «упоминании»). В нашем примере для построения матриц смежности будут использоваться следующие порядки вершин:

A: 1, 2, 6, 4, 5, 3

B: 3, 4, 5, 1, 2, 6

Матрица смежности для A для заданного порядка вершин:

Номера вершин	1	2	6	4	5	3
1	0	1	1	0	0	0
2	0	0	0	0	0	0
6	0	0	0	0	0	0
4	0	0	0	0	1	0
5	1	0	0	0	0	0
3	0	0	0	0	0	1

Матрица смежности для B для заданного порядка вершин:

Номера вершин	3	4	5	1	2	6
3	0	1	1	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
1	0	0	0	0	1	0
2	1	0	0	0	0	0
6	0	0	0	0	0	1

Матрицы равны, следовательно, графы A и B – изоморфны.

Разумеется, данный подход будет иметь свои преимущества и ограничения.

Преимущества:

- Применим как для ориентированных, так и неориентированных графов,
- Применим для графов, содержащих более одной компоненты связности/ сильной связности,
- Применим для графов, содержащих кратные (множественные) ребра и петли.

Ограничения:

- Не учитывает вершины, для которых отсутствуют инцидентные им ребра,
- Применение данного подхода для поиска подграфов графа **A**, **изоморфных некоторому графу B**, сможет находить лишь «вписанные» подграфы. «Вписанным» подграфом графа A здесь называется такой его подграф, который может быть «приклеен» к другим частям графа A только за счет ребер, инцидентных лишь граничным вершинам его (подграфа) неразветвляющихся путей максимальной длины (при этом граф A может содержать и иные компоненты связности).

При решении задачи поиска в графе A всех его «вписанных» подграфов, **изоморфных графу B**, помимо поиска соответствий максимально протяженных неразветвляющихся путей графов A и B по длине, также необходимо отдельно искать и следующие возможные соответствия между ними:

- Соответствия максимально протяженных неразветвляющихся путей-простых цепей графа B максимально протяженным неразветвляющимся путям-простым цепям либо простым циклам графа A. При этом изначально такие пути из графа A могут быть длиннее – в этом случае берутся их отрезки, равные по длине искомому пути из B.
- Соответствия максимально протяженных неразветвляющихся «концевых» путей графа B (т.е. таких, одна и только одна из граничных вершин которого соединена (причем независимо от направления такой связи, если граф – ориентированный) с одной и только одной другой вершиной графа B) каким-либо максимально протяженным неразветвляющимся путем графа A (при этом пути из графа A также могут быть длиннее – в этом случае берутся их отрезки, равные по длине искомому пути из графа B).

Пример.

При поиске «вписанного» изоморфного подграфа графа B в графе A (см. рис. 1) будут обнаружены следующие соответствия:

- внутренний путь 2->3->4 графа B: внутренний путь 2->3->4 графа A,
- концевые пути 1->2 и 10->2 графа B: концевой путь 0->2 графа A и фрагмент концевого пути 7->1->2 графа A, а именно 1->2,
- простая цепь 7->8 графа B: отрезки простой цепи 9->10->11 графа A, а именно 9->10 и 10->11, а также отрезки простого цикла 12->13->14->12 графа A, а именно 12->13, 13->14, и 14->12.

Таким образом, в качестве «изоморфных графу B «вписанных» подграфов графа A могут быть найдены следующие 5 вариантов:

$A1 = \{0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 4 \rightarrow 6, 9 \rightarrow 10\}$
 $A2 = \{0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 4 \rightarrow 6, 10 \rightarrow 11\}$
 $A3 = \{0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 4 \rightarrow 6, 12 \rightarrow 13\}$
 $A4 = \{0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 4 \rightarrow 6, 13 \rightarrow 14\}$
 $A5 = \{0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 5, 4 \rightarrow 6, 14 \rightarrow 12\}$

Однако, исходя из ограничений данного метода, при добавлении к графу А дополнительного ребра 3->8 (см. граф А' на том же рисунке) «вписанные» изоморфные графу В подграфы не будут найдены: ребро 3->8 «разбивает» путь 2->3->4 графа А на два и, следовательно, путей-кандидатов для внутреннего пути 2->3->4 графа В обнаружено не будет.

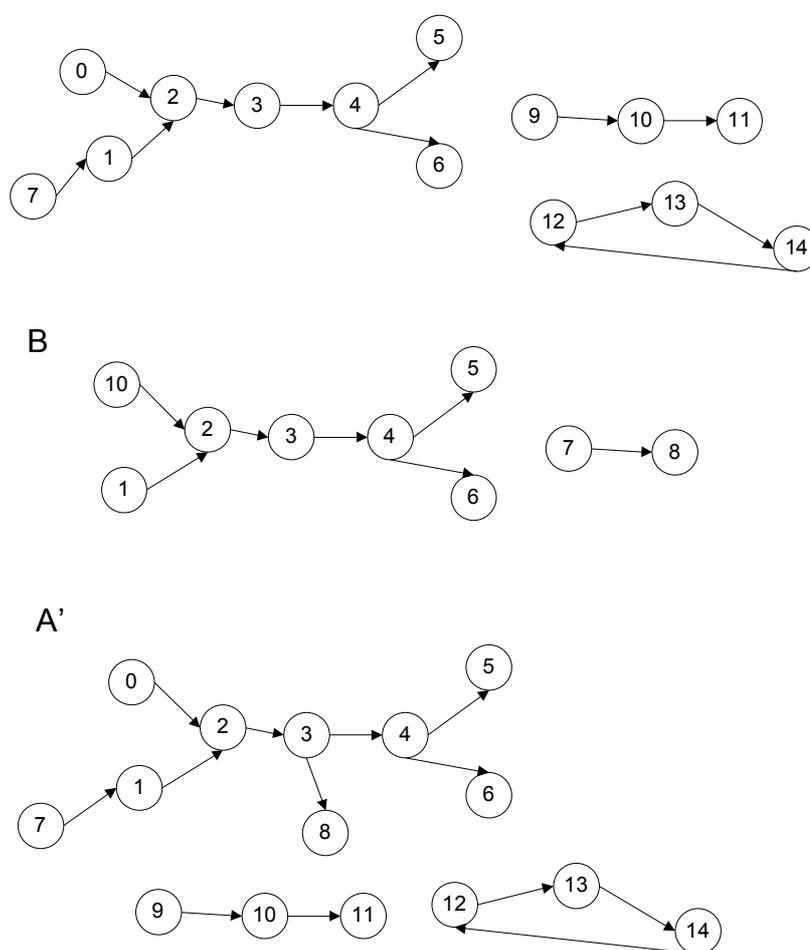


Рис. 1

2. Описание алгоритма поиска изоморфных «вписанных» подграфов

Исходя из сказанного в разделе 1, общее описание алгоритма поиска «вписанных» в граф А подграфов, изоморфных некоторому графу В, будет выглядеть следующим образом.

I. Препроцессинг

Формирование всех неразветвляющихся путей максимальной длины путей по каждому из графов, а также оценка факторов, ограничивающих пространство выбора при переборе, а именно:

1. Нахождение всех максимальных неразветвляющихся путей в графе А и запись их в динамический массив (вектор) PathsA
2. Нахождение всех максимальных неразветвляющихся путей в графе В и запись их в динамический массив (вектор) PathsB
3. Вычисление и запоминание кратности всех ребер графов А и В. Дальнейшие действия по этапам II-IV проводятся исходя из допущения, что кратности всех ребер равны 1. Однако на финальной стадии производится сравнение кратности ребер подграфа графа А-кандидата на изоморфизм с графом В и кратности ребер самого графа В: все ребра подходящего графа-кандидата должны иметь кратность не меньшую, чем сопоставляемое им ребро графа В.
4. Подсчет степеней всех вершин графов А и В (для ориентированных графов степени по входящим и исходящим ребрам учитываются отдельно).
5. Вычисление матриц кратчайших путей между каждой парой вершин в А и в В: DA и DB соответственно.
6. Формирование матрицы смежности B00 для графа В в порядке, указанном в разделе 1, исходя из очередности путей в PathsB.

II. Сопоставление

Поиск путей-кандидатов из PathsA для каждого из путей в PathsB. Отметка о каждом пути-кандидате из PathsA для каждого i-го пути PathsB[i] будет записываться в двумерный динамический массив (вектор векторов) NPaths – соответственно в i-й вектор (i-ю строку) – в виде упорядоченного триплета чисел: номер соответствующего пути в PathsA – номер стартовой позиции в нем – длина пути.

Например, PathsB[2] = {1, 0, 3, 3, 1, 3} означает, что пути PathsB[2] сопоставлено 2 пути-кандидата из А: из PathsA[1] – 3 первых элемента пути начиная с нулевого (начального), и из PathsA[3] – также 3 элемента начиная с первого (следующего за начальным).

При этом поиск путей-кандидатов проводится по 4-м направлениям:

1. Поиск путей-кандидатов для всех внутренних путей графа В из PathsB, т.е. тех, обе граничные вершины которых соединены в графе В как минимум с 2-мя другими вершинами (независимо от направления такой связи) и при этом такой путь не является простым циклом (ориентированным – для ориентированных графов).
2. Поиск путей-кандидатов для концевых путей из PathsB.
3. Поиск путей-кандидатов для простых циклов из PathsB.
4. Поиск путей-кандидатов для простых цепей из PathsB.

При отборе путей-кандидатов для каждого i-го пути из PathsB сравниваются:

- его длина и длина пути-кандидата (должны быть равны для случаев 1 и 3, а для случаев 2 и 4 путь-кандидат из PathsA также может быть длиннее),
- степени его граничных вершин и соответствующих им вершин пути-кандидата (у вершин из пути-кандидата данные значения должны быть не менее чем у пути из PathsB),
- опционально: кратности его ребер и ребер пути-кандидата (кратность ребра пути-кандидата должна быть не менее, чем у потенциально соответствующего ему ребра пути из PathsB),

– опционально: другие возможные ограничительные параметры (например, совпадения меток вершин, что может быть применимо, в частности, при задании графами химических соединений (в этом случае вершины можно маркировать соответствующими атомами), и т.д.).

Если по результатам этапа II для какого-либо из путей в PathsB не найдено ни одного пути-кандидата из PathsA, то считается, что A не содержит «вписанных» подграфов, изоморфных графу B.

Предварительное прореживание

На основании всех отобранных путей-кандидатов обновляется граф A – в нем остаются только те ребра, которые вошли в пути-кандидаты. Если хотя бы одно ребро исходного графа было удалено – осуществляется возврат на этап I “Препроцессинг”: степени вершин графа A и, соответственно, перечень путей-кандидатов может быть уменьшен и, соответственно, может быть сокращено пространство поиска.

III. Прореживание перечня путей-кандидатов

Целью настоящего этапа является максимально возможное исключение неподходящих путей кандидатов до их полного перебора. Для этого осуществляются следующие действия.

1. Исключение заведомо неподходящих путей-кандидатов исходя из минимальных расстояний между вершинами в графе B. Исключению подлежит тот путь-кандидат по каждому PathsB[i], для которого хотя бы по одному PathsB[j] не нашлось ни одного такого пути-кандидата, чтобы кратчайшие расстояния между их граничными вершинами в графе A было меньше или равно кратчайшему расстоянию между соответствующими им граничными вершинами путей PathsB[i] и PathsB[j] из графа B.

2. Исключение для всех циклов из PathsB сопоставленных им нециклических путей-кандидатов и наоборот.

3. Аналогичное п. 1 исключение путей-кандидатов, но не по критерию кратчайшего расстояния между граничными вершинами, а их (граничных вершин) взаимному равенству либо неравенству.

Например, если:

- начальный элемент пути PathsB[i] не равен начальному элементу пути PathsB[j], и
- конечный элемент пути PathsB[i] не равен конечному элементу пути PathsB[j], и
- начальный элемент пути PathsB[i] равен конечному элементу пути PathsB[j], и
- начальный элемент пути PathsB[j] не равен конечному элементу пути PathsB[i],

то исключению подлежит тот путь-кандидат по пути PathsB[i], для которого хотя бы по одному PathsB[j] не нашлось ни одного пути-кандидата, удовлетворяющего приведенному выше условию относительно равенства/ неравенства соответственно их (путей-кандидатов) начальных и конечных вершин.

Если по результатам этапа III хотя бы 1 путь-кандидат был удален из PathsA, то обновляется граф A – в нем остаются только те ребра, которые вошли в оставшиеся пути-кандидаты. Если при этом хотя бы одно ребро исходного графа A было удалено – возврат на этап I “Препроцессинг”: степени вершин графа A и, соответственно, перечень путей-кандидатов может быть уменьшен и, соответственно, может быть сокращено пространство поиска.

IV. Заключение

Каждое возможное сочетание всех оставшихся путей-кандидатов задает подграф графа А. Для каждого такого подграфа строится матрица смежности с учетом порядка следования путей кандидатов таким же образом, как была построена матрица смежности V_{00} для графа В, а затем эти матрицы смежности сравниваются между собой.

Если они равны, то сравниваются кратности ребер (см. п. 3 Этапа I Препроцессинг). Если все эти условия выполнены – найденный подграф считается изоморфным графу В и добавляется в множество найденных результатов.

При проведении этапа IV “Заключение” возможно применение различных дополнительных проверок, ускоряющих выявление и отбрасывание неподходящего подграфа.

3. Пример результатов работы алгоритма

На рис. 2 приведены графы $A = \{1 \rightarrow 2, 2 \rightarrow 5, 5 \rightarrow 15, 16 \rightarrow 15, 5 \rightarrow 5, 5 \rightarrow 5, 5 \rightarrow 4, 5 \rightarrow 6, 7 \rightarrow 6, 6 \rightarrow 8, 6 \rightarrow 6, 6 \rightarrow 9, 9 \rightarrow 10, 9 \rightarrow 11, 12 \rightarrow 0, 0 \rightarrow 12, 4 \rightarrow 13, 13 \rightarrow 14, 14 \rightarrow 4\}$ и $B = \{1 \rightarrow 1, 4 \rightarrow 5, 5 \rightarrow 1, 1 \rightarrow 3, 3 \rightarrow 3, 3 \rightarrow 2, 2 \rightarrow 7, 2 \rightarrow 8, 9 \rightarrow 10, 10 \rightarrow 9, 1 \rightarrow 6, 6 \rightarrow 11, 11 \rightarrow 12, 12 \rightarrow 6\}$. Задачей является нахождение в графе А всех «вписанных» подграфов, изоморфных графу В. Результат также отражен на рисунке: найденные вершины и ребра графа А выделены толстой линией, а номера соответствующих вершин графа В указаны в скобках (если вариантов несколько – они указываются через дробь).

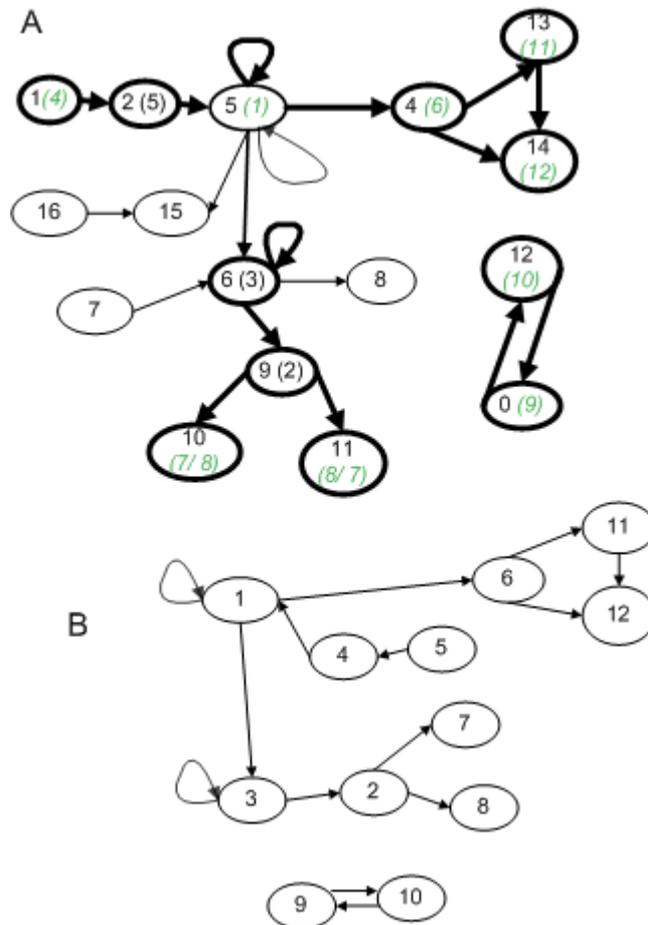


Рис. 2

При решении задачи поиска в графе А «вписанных» подграфов, изоморфных графу В, имеем следующие результаты работы алгоритма.

Этап I: Выполнены все действия и расчеты по п.п. 1-5 данного этапа, ниже приводятся полученные PathsA и PathsB.

PathsA:

1->2->5

4->13->14 4

5->4

5->5

5->6

5->15

6->6

6->8

6->9

7->6

9->10

9->11

16->15

0->12->0

PathsB:

1->1

1->3

1->6

2->7

2->8

3->2

3->3

4->5->1

6->11->12->6

9->10->9

Этапы II-III. Сопоставление показало, что для каждого пути из PathsA находится как минимум один путь-кандидат из PathsB, а по результатам предварительного прореживания PathsA был сокращен. На этапе основного прореживания (III) дальнейшего сокращения PathsA не произошло. В результате PathsA и PathsB приняли вид:

PathsA:

1->2->5

4->13->14->4

5->4

5->5

5->6

6->6
6->9
9->10
9->11
0->12->0

PathsB:

1->1
1->3
1->6
2->7
2->8
3->2
3->3
4->5->1
6->11->12->6
9->10->9

Итоговое сопоставление NPaths имеет вид:

NPaths:

i=0: 3 0 2
i=1: 4 0 2
i=2: 2 0 2
i=3: 7 0 2 8 0 2
i=4: 7 0 2 8 0 2
i=5: 6 0 2
i=6: 5 0 2
i=7: 0 0 3
i=8: 1 0 4
i=9: 9 0 3

Каждый упорядоченный триплет чисел в NPaths[i] задает соответствующий PathsB[i] путь-кандидат из A в формате: номер соответствующего пути из PathsA – стартовая позиция отрезка из данного пути – длина отрезка. Таким образом, нетрудно убедиться, что пути PathsB[0] = {1->1} (i=0: 3 0 2) соответствует единственный путь из A, а именно: отрезок из пути PathsA[3], начинающийся с его самого начала и имеющий длину 2.

Этап IV. В результате в графе A был найден единственный подграф A1, изоморфный графу B. $A1 = \{0 \rightarrow 12, 1 \rightarrow 2, 2 \rightarrow 5, 4 \rightarrow 13, 5 \rightarrow 4, 5 \rightarrow 5, 5 \rightarrow 6, 6 \rightarrow 6, 6 \rightarrow 9, 9 \rightarrow 10, 9 \rightarrow 11, 12 \rightarrow 0, 13 \rightarrow 14, 14 \rightarrow 4\}$.

4. Расширение возможностей алгоритма: поиск любых (а не только «вписанных») подграфов

Исходя из сути предложенного подхода можно сделать вывод о возможности его применения для поиска всех (а не только «вписанных») подграфов в данном графе, изоморфных заданному графу-образцу. Для этого вместо максимально протяженных

неразветвляющихся путей необходимо рассматривать все ребра указанных графов. В остальном подход не претерпевает изменений, при этом при его практической реализации возможны различные частные особенности, не меняющие его принципиально.

5. Область применения и пути дальнейшего развития алгоритма

При введении дополнительных признаков вершин графов А и В (например, при задании графами химических соединений каждой вершине может быть сопоставлен числовой код, соответствующий одному и только одному атому (изотопу)) процесс поиска может быть ускорен за счет большей точности сопоставлений по Этапу II: дополнительным условием отбора путей-кандидатов будет соответствие меток вершин.

Скорость работы представленного алгоритма очень сильно зависит от структур исходных графов и их взаимного соответствия. Алгоритм будет работать тем быстрее, чем более «уникальную», «нестандартную» и несимметричную структуру имеет граф-образец В.

Исходя из этого, одним из возможных путей использования представленного алгоритма может являться, *в том числе*, **«скрининговый» поиск решения** по любому из возможных направлений его применения, а именно:

- (1) поиск именно «вписанных» подграфов, изоморфных данному,
- (2) проверка изоморфности двух различных графов,
- (3) поиск всех (а не только «вписанных») подграфов, изоморфных данному,

«Скрининг» здесь подразумевает «предварительный экспресс-поиск решения», для чего необходимо задать некоторое предельное время его работы, по истечении которого при отсутствии результата можно считать, что данная задача подлежит решению другим алгоритмом.

Дополнительно о реализации алгоритма. Автором была предпринята попытка реализации данного алгоритма на языке C++: функция SubGraphsInscribed добавлена в свободно распространяемую библиотеку с открытым исходным кодом CBioInfC++ [6].

Примечания.

1. Данная библиотека содержит набор функций для работы со строками и графами и предназначена для решения как биоинформатических, так и других задач, где это может быть востребовано.
2. Данная библиотека целенаправленно реализована в заголовочном файле и без использования шаблонов, чтобы облегчить «порог вхождения».
3. Текущая версия реализации функции SubGraphsInscribed ориентирована более для работы с ориентированными графами, хотя работает и с неориентированными. Для неориентированных графов этап III «Прореживание» на сегодняшний день пока реализован не полностью.
4. Для хранения графов в CBioInfC++ применены ранее предложенные автором структуры данных «Вектор смежности» и «Ассоциативный массив смежности» (подробнее см. [5]). Если кратко, то для невзвешенных графов вектор смежности представляет собой упорядоченный набор четного количества целых чисел ($a[2i]$, $a[2i+1]$, ..., где i нумеруется с 0), в котором каждая пара чисел $a[2i]$, $a[2i+1]$ задает ребро графа между вершинами $a[2i]$ и $a[2i+1]$ соответственно. Для взвешенных графов с целочисленными весами ребер «Вектор смежности» представляет собой набор триплетов, где каждое третье число задает вес ребра, а для взвешенных с нецелочисленными весами ребер – пару векторов (динамических массивов) целых чисел (задает ребра) и вещественных чисел (веса ребер). В «Ассоциативном массиве смежности» ключу – паре целых чисел, задающих ребро графа, – соответствует значение его веса (целочисленное либо

вещественное) либо вектор (динамический массив) таких весов для случая множественных ребер. Также в векторах и ассоциативных массивах смежности, помимо весов ребер, можно хранить и другие их свойства, задаваемые числами.

Автор выражает надежду, что представленный алгоритм (как *полностью*, так и *его отдельные элементы*) может оказаться полезным как при решении различных практических задач, так и при *разработке новых алгоритмов решения задачи* об изоморфном подграфе. Автор будет рад дополнительно проверить представленный алгоритм на различных наборах исходных данных и поделиться результатами своих проверок.

Литература

1. Bonnici, V., Giugno, R., Pulvirenti, A. et al. A subgraph isomorphism algorithm and its application to biochemical data. BMC Bioinformatics 14, S13 (2013). <https://doi.org/10.1186/1471-2105-14-S7-S13>.
2. Cordella L, Foggia P, Sansone C, Vento M: A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004, 26 (10): 1367-1372. 10.1109/TPAMI.2004.75.
3. Ullmann Julian R.: An algorithm for Subgraph Isomorphism. Journal of the Association for Computing Machinery. 1976, 23: 31-42. 10.1145/321921.321925.
4. Ullmann Julian R.: Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. J Exp Algorithmics. 2011, 15 (1.6): 1.1-1.6. 1.64
5. Черноухов С.А. Вектор смежности и ассоциативный массив смежности как способы представления и хранения графов / S.A. Chernouhov. Adjacency vector and adjacency map as data structures to represent a graph // Сборник статей Международной научно-практической конференции «Проблемы внедрения результатов инновационных разработок и пути их решения» (Саратов, 14.09.2019 г.). – Стерлитамак: АМИ, 2019, с. 65-69.
6. Репозиторий библиотеки CBioInfCpp (включая сопутствующую документацию): <https://github.com/chernouhov/CBioInfCpp-0->