УДК 519.171.1

ПОЛНЫЙ ИНВАРИАНТ ГРАФА И АЛГОРИТМ ЕГО ВЫЧИСЛЕНИЯ

Погребной Андрей Владимирович,
магистрант кафедры информатики и проектирования систем
Института кибернетики Томского политехнического университета,
Россия, 634050, г. Томск, пр. Ленина, д. 30. E-mail: avpogrebnoy@gmail.com

Актуальность *научной работы определяется тем, что в теории графов начиная со средины прошлого века все попытки найти вид полного инварианта и разработать для него эффективный алгоритм вычисления оказывались безуспешными. Предложенное в статье решение данной проблемы будет способствовать развитию методов инвариантного представления и анализа абстрактных структур графов.*

Цель исследования: *сформулировать теоретические положения метода независимой интеграции кодов структурных различий и на этой основе разработать эффективный алгоритм вычисления полного инварианта графа.*

Методы исследования *основаны на теории графов и методах свободной и зависимой интеграции кодов структурных различий для получения интегральных описателей вершин абстрактных структур графов.*

Результаты. *Предложено новое правило назначения кодов структурных различий для дифференциации вершин структуры графа. Правило отличается простотой, представляет независимую систему кодирования и гарантирует получение интегрального описателя структуры (Integral Structure Descriptor – ISD), инвариантного относительно исходной нумерации её вершин. Используя данное правило, разработан метод независимой интеграции кодов структурных различий в графе. На основе этого метода разработан эффективный алгоритм вычисления полного инварианта графа. Показано, что для самых неблагоприятных случаев предельные объёмы вычислений ограничиваются полиномиальными оценками. На языке Java разработано программное средство GraphISD и проведены экспериментальные исследования эффективности работы алгоритма. Эксперименты показали, что предложенный полный инвариант и алгоритм его вычисления способны эффективно работать с библиотеками графов, содержащих до 5000 вершин, инвариантно представлять графы в библиотеке, выделять изоморфные графы на основе сравнения полных инвариантов, формировать подстановки изоморфизма и исходные представления графов.*

Ключевые слова:
Полный инвариант графа, абстрактная структура графа, однородный граф, интегральный описатель структуры, устойчивая группа вершин, симметричный граф, изоморфизм графов.

Введение

Полным инвариантом графа G называют некоторую количественную характеристику $P(G)$, которая представляет его структуру с точностью до изоморфизма [1], т. е. равенство полных инвариантов $P(G)$ и $P(H)$ для графов G и H гарантирует их изоморфизм. В данном определении отсутствуют сведения о форме и содержании характеристики $P(G)$. Поэтому в исследованиях по проблеме получения полного инварианта приходится решать две задачи – поиск вида характеристики $P(G)$ и разработка приемлемого для практического применения алгоритма её вычисления. Проблемы изоморфизма и поиска полного инварианта были актуальны с начала становления теории графов. Эти проблемы не остаются без внимания и в настоящее время. Вместе с тем полные инварианты удалось получить лишь для отдельных видов графов, например, в работах [2, 3] – для ациклических графов, в [4] – для планарных.

Для анализа структур графов широко применяются инварианты, отражающие локальные характеристики графа. Известны исследования, в которых в качестве инвариантов выступают объединения локальных характеристик [5]. Сюда можно отнести также многие эвристические алгоритмы для определения изоморфизма графов, у которых вершины и/или рёбра помечены атрибутами [6]. Подобные исследования нашли широкое применение при анализе молекулярных структур в химии и биологии [7–11].

К настоящему времени известен один полный инвариант, названный миникодом $\mu(G)$ или максикодом $\mu^*(G)$ [12]. Для получения миникода $\mu(G)$ граф G, содержащий n вершин, представляется матрицей смежности вершин A. Элементы матрицы A в определенной последовательности формируются в одну строку, которая принимается в качестве двоичного числа и преобразуется в десятичную форму. Такие числа вычисляются для всех $n!$ вариантов матрицы A, и минимальное из них принимается в качестве $\mu(G)$. Таким образом, в данном подходе к получению полного инварианта найден только вид характеристики $P(G)$, а алгоритм её вычисления не может уйти от перебора $n!$ матриц смежности и, следовательно, не пригоден для практического применения.

В работах [13, 14] предложен другой вид полного инварианта и разработан алгоритм его вычисления. В этих работах характеристика $P(G)$ представляется в виде вектора $P(G)=\{d_i(F(d_i))\}$. Каждый элемент $d_i(F(d_i))$ соответствует определенной вершине абстрактной структуры графа G и является уникальным описателем (дескриптором) данной вершины. В записи $d_i(F(d_i))$ отражено имя вершины в виде уникального кода d_i, принимающего значения $1,2,\ldots,n$, n – число вершин в графе G и инцидентор $F(d_i)$ этой вершины в форме множества имён $d_j \in F(d_i)$ тех вершин, с которыми она связана. Например, элемент $d_i(F(d_i))=6(2\ 7,9)$ означает, что вершина с уникальным кодом $d_i=6$ в абстрактной структуре графа G связана с вершинами, имеющими уникальные коды 2, 7, 9.

Очевидно, что получение полного инварианта $P(G)$ в этом случае сводится к вычислению уникальных кодов d_i вершин абстрактной структуры графа G. Для вычисления кодов d_i используется метод интеграции кодов структурных различий в графах и получения интегральных описателей структур – Integral Structure Descriptor (ISD) [13, 14]. Метод получения ISD в [14] представлен двумя алгоритмами – свободной и зависимой интеграции, которые в последующем будем именовать ISD-F и ISD-D соответственно.

Полный инвариант $P(G)$, полученный с помощью алгоритма ISD-F, принимается в качестве эталона для поиска графов, изоморфных графу G. Поиск таких графов, например графа H, осуществляется с помощью алгоритма ISD-D путём получения полного инварианта $P_G(H)$, зависимого от эталона на основе графа G. Если для графа H инвариант $P_G(H)=P(G)$ найден, то граф H изоморфен графу G. Оба алгоритма эффективно работают, например, при решении задачи разбиения множества графов на классы изоморфных либо задачи, связанной с выбором из множества графов таких, которые изоморфны заданному графу.

Свободная интеграция кодов, реализованная в алгоритме ISD-F, сопровождается формированием системы кодирования для класса графов, представленных графом G, и необходимостью её хранения в качестве эталона для последующего применения при работе алгоритма ISD-D. Заметим, что алгоритм ISD-F в таком виде был разработан не столько для достижения высокой эффективности при решении задачи определения изоморфизма графов, сколько потому, что авторам не удалось решить проблему независимой интеграции кодов при получении полных инвариантов.

Решение этой проблемы на основе метода ISD связано с разработкой алгоритма получения полного инварианта $P(H)$ для любого графа H без использования эталона в виде $P(G)$ и соответствующей системы кодирования. При этом, если полные инварианты графов G и H, полученные независимо друг от друга оказались равными, т. е. $P(G)=P(H)$, графы G и H изоморфны, а при $P(G)\neq P(H)$ неизоморфны.

Статья посвящена разработке метода независимой интеграции кодов структурных различий и соответствующего алгоритма вычисления полного инварианта графа. Алгоритм получил название ISD-I и реализован на языке Java в составе программы GraphISD. В статье также приведены результаты экспериментальных исследований эффективности работы алгоритма ISD-I при вычислении полных инвариантов для однородных и неоднородных графов большой размерности.

Метод независимой интеграции кодов структурных различий

Рассмотрению подлежат абстрактные структуры обыкновенных графов. Если вершины абстрактной структуры пронумеровать в произвольном порядке числами от 1 до n, то соответствующий граф $G=(E,U)$ с множеством вершин $E=\{e_i\}$ и множеством рёбер $U=\{u_{ij}\}$ можно представить матрицей смежности $A=\|a_{ij}\|$. Элемент $a_{ij}=1$, если ребро $u_{ij}\in U$, $a_{ij}=0$, если ребра u_{ij} нет. Описание абстрактной структуры графа G в виде матрицы A вполне пригодно для автономного исследования его свойств. Но как только возникает потребность в сопоставлении свойств графов в составе множества, то из-за произвольной нумерации вершин графы идентифицируются неоднозначно. Например, один и тот же граф при разной нумерации вершин в исследуемом множестве будет восприниматься двумя разными графами. Широко известным примером исследуемого множества графов являются молекулярные структуры химических соединений [15–17].

Описание обыкновенного графа можно представить в виде списка $\{e_i(F(e_i))\}$ инцидентов $F(e_i)$ вершин e_i. Такое описание графа G по форме полностью соответствует представлению полного инварианта в виде $P(G)=\{d_i(F(d_i))\}$. Можно сказать, что цель метода независимой интеграции кодов заключается в нахождении алгоритма преобразования $\{e_i(F(e_i))\}\Rightarrow\{d_i(F(d_i))\}$, в котором номера вершин e_i заменяются на уникальные коды интегральных описателей вершин d_i. При этом в отличие от свободной интеграции кодов в данном методе должны быть разработаны универсальные правила вычисления уникальных кодов вершин и соответствующих полных инвариантов.

При изложении правил вычисления уникальных кодов будем придерживаться схемы процесса интеграции, приведенной на рис. 1.

Схема на рис. 1 отражает рекуррентное изменение кодов d_i^k для одной вершины e_i. Аналогичные изменения выполняются параллельно для всех других вершин графа. Слева на схеме расположено визуальное представление инцидентора для одной из вершин абстрактной структуры. После произвольной нумерации вершин структуры соответствующий инцидентор для вершины e_i записан в виде $F(e_i)=(e_{j1},e_{j2},\ldots,e_{js_i})$. Здесь e_j – вершины, инцидентные рёбрам u_{ij}; s_i – степень вершины e_i, $s_i=|F(e_i)|$.

Перед запуском процесса интеграции производится начальное назначение кодов d_i^0 вершинам графа и формирование исходного вектора кодов $D^0=\{d_i^0\}$. При этом должны соблюдаться два требования: наличие исходной дифференциации вершин и однозначность назначения кодов.

Первое требование является условием запуска процесса интеграции. Исходная дифференциация вершин может быть получена на основе легко вычисляемых характеристик, таких как степени вершин. Для однородных графов, у которых степени вершин равны, приходится использовать другие характеристики. Рассмотрим одну из таких характеристик на основе матрицы A^2. Такую характеристику назовём маршрутной. Для однородного графа в строках матрицы A содержится одинаковое число единиц, т. к. степени вершин совпадают.

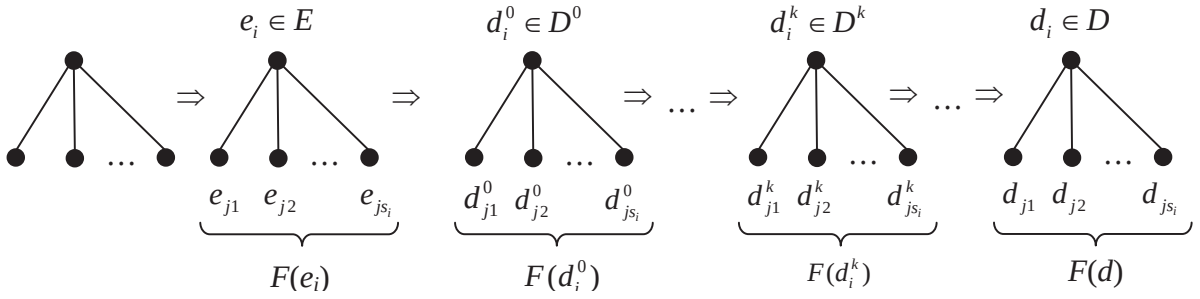

Рис. 1. *Схема процесса интеграции кодов*

Fig. 1. *Code integration*

В матрице A^2 элементы a_{ij}^2 равны числу маршрутов длины 2 между вершинами e_i и e_j. Если i-я строка матрицы A^2 с множеством элементов A_i^2 не совпадает с множеством элементов A_j^2 для j-й строки, то вершины e_i и e_j по данной маршрутной характеристике различаются.

В отношении маршрутных характеристик на основе матриц A^g, g=2,3,... можно высказать следующее предположение. Однородные графы, для которых маршрутные характеристики не приводят к дифференциации вершин, являются симметричными. Понятие симметричного графа здесь формально не вводится и подробно не рассматривается.

Пример абстрактной структуры симметричного графа показан на рис. 2 (слева), а несимметричного, содержащего структурное различие, расположен справа. Нетрудно убедиться, что для симметричного графа маршрутные характеристики не приводят к дифференциации вершин, а для несимметричного дифференциация происходит уже на основании матрицы A^2.

Наличие симметрии в структуре графа предопределяет возможность существования в нём устойчивых групп [13]. Вершины устойчивых групп образуют однородные подграфы. Поэтому однородный граф можно рассматривать также в качестве устойчивой группы и для исходной дифференциации его вершин вводить виртуальное структурное различие [13]. К данному способу будем прибегать в крайнем случае, когда маршрутные характеристики для приемлемых степеней матриц не приводят к успеху и, следовательно, граф с большой вероятностью можно отнести к симметричному.

Требования однозначности при назначении кодов должно соблюдаться как при назначении кодов для получения исходной дифференциации вершин, так и на последующих шагах интеграции по преобразованию векторов как это показано на рис. 1. Соблюдение требования однозначности является принципиальным отличием независимой интеграции от свободной и зависимой.

В алгоритмах ISD-F и ISD-D назначение кодов инцидентам вершин $F(d_i^k)$ выполнялось исходя из естественного стремления на каждом k-м шаге интеграции привязать назначение кода d_i^k к конкретному составу и значениям кодов d_i^k в инциденторе $F(d_i^k)$. При независимой интеграции такой привязки нет и хранить её в памяти в виде соответствия $F(d_i^k) \rightarrow d_i^{k+1}$ нет необходимости. Однозначность назначения кодов при независимой интеграции достигается за счёт введения новых правил при выполнении назначения.

Совокупность этих правил будет подробно рассмотрена при изложении алгоритма ISD-I в следующем разделе. Здесь лишь отметим, что при поиске таких правил потребовалось преодолеть своего рода барьер, отделявший исследователя от мысли о существовании альтернативной системы кодирования, пригодной для решения проблемы вычисления полного инварианта. В этой системе ко-

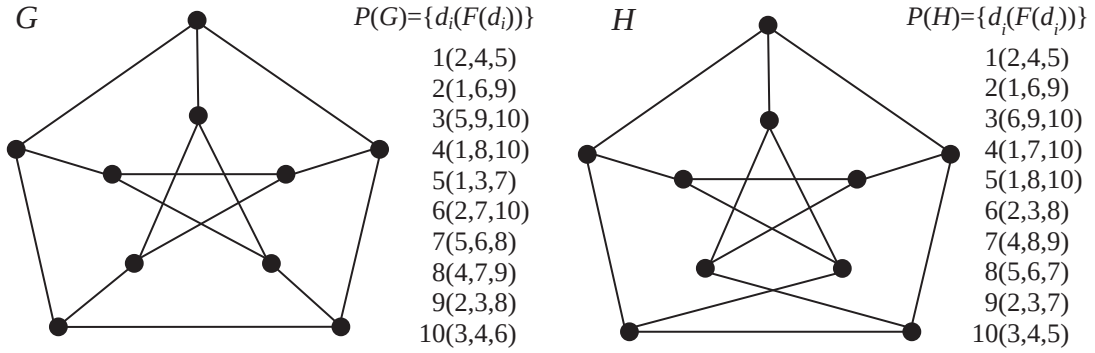

Рис. 2. *Симметричный граф (слева) и несимметричный (справа)*

Fig. 2. *Symmetric (left) and asymmetrical (right) graph*

дирование и хранение «бесконечного» числа комбинаций кодов инцидентов $F(d_i^k)$ удалось заменить назначением конечного числа кодов d_i^k, которые отражают отношения порядка (<, =, >) между инцидентами. Таким образом, в предлагаемом методе независимой интеграции на каждом k-м шаге кодируются не сами инциденторы $F(d_i^k)$, а их места, которые они занимают в упорядоченном множестве $\{d_i^k(F(d_i^k))\}$. Назначенные при этом коды принимают значения 1, 2,...,n и вместе с тем обеспечивают соблюдение требования однозначности.

Основное правило назначения кодов d_i^k, используемое в методе независимой интеграции кодов, сводится к следующему. Для каждого инцидентора $F(d_i^k)$ формируется числовой эквивалент. В итоге на k-м шаге интеграции для n инцидентов получаем n числовых эквивалентов. Минимальному числовому эквиваленту, среди тех которые относятся к минимальному d_i^k, назначается код $d_i^{k+1}=1$, следующему по величине – код $d_i^{k+1}=2$ и т. д. Равные числовые эквиваленты получают равные коды. Процесс пошаговой интеграции выполняется до тех пор, пока на очередном $(k+1)$-м шаге все числовые эквиваленты окажутся разными и, следовательно, будут использованы все коды от 1 до n. Соответствующий вектор D^{k+1} является интегральным описателем $D=\{d_i\}$ абстрактной структуры, а совокупность инцидентов $\{d_i(F(d_i))\}$ упорядоченная по возрастанию значений кодов d_i принимается в качестве полного инварианта.

Алгоритм вычисления полного инварианта

В данном разделе раскрывается содержание операций алгоритма ISD-I и их взаимодействие при вычислении полного инварианта $P(G)$ для графа G. Предполагается, что граф $G=(E,U)$ описывает некоторую абстрактную структуру с помощью произвольной нумерации вершин множества $E=\{e_i\}$, $i=1,2,...,n$ и представлен списком инцидентов $\{e_i(F(e_i))\}$. Если граф G однородный, то его представление дополняется матрицей смежности A. Полные графы не рассматриваются, т. к. для них любая нумерация вершин соответствует вектору D. Предполагается также, что однородные графы имеют степени $s \le n/2$, иначе вместо таких графов можно рассматривать их дополнения. Результатом работы алгоритма является полный инвариант $P(G)=\{d_i(F(d_i))\}$, который вычисляется с помощью следующих операций.

1. *Формирование исходного вектора D^0.* Для этого по списку $\{e_i(F(e_i))\}$ определяются степени s_i вершин e_i, $s_i=|F(e_i)|$. Инциденту $F(e_i)$ с минимальной степенью s_i назначается код $d_i^0=1$. Следующий инцидентор с более высокой степенью получает код 2 и т. д. Инциденторы с равными степенями получают равные коды.

Если в векторе D^0 все $d_i^0=1$, т. е. граф G однородный, то вычисляется матрица A^2. Множества ненулевых элементов строк A_i^2 матрицы A^2 упорядочиваются по возрастанию значений и преобразуются в числовые эквиваленты. Например, i-я строка содержит упорядоченное множество ненулевых элементов $A_i^2=(a_1,a_2,..,a_r)$. Тогда числовой эквивалент β_i принимается равным $(a_r+a_{r-1}10^{\delta}+a_{r-2}10^{2\delta}+...+a_1 10^{(r-1)\delta})$, где δ – число десятичных разрядов у наибольшего элемента матрицы A^2. Так, если $A_i^2=(2,7,15,15)$, то $\beta_i=15+15\cdot10^2+7\cdot10^4+2\cdot10^6=2071515$.

Коды d_i^0 при наличии числовых эквивалентов β_i назначаются аналогично тому, как это делалось для степеней s_i. Если для матрицы A^2 дифференциация вершин не происходит, то в зависимости от величины n процесс анализа маршрутных характеристик продолжается для матриц с более высокими степенями. При этом предельные значения степеней в алгоритме, исходя из требования однозначности, должны быть ограничены. Отсутствие дифференциации и для предельных степеней может означать, что граф G является симметричным или вычисленные маршрутные характеристики не выявили в нём структурных различий. В программной реализации алгоритма в составе GraphISD начальная дифференциация вершин в однородном графе с помощью маршрутных характеристик выполняется только на основе матрицы A^2. Если в этом случае дифференциация вершин не достигается, то граф G принимается в качестве устойчивой группы.

Начальная дифференциация вершин такого однородного графа (устойчивой группы) осуществляется с помощью введения в него виртуального структурного различия. Для этого последовательно для каждой вершины e_i назначается код $d_i^0=2$. При этом все коды вектора D^0, за исключением d_i^0, остаются равными 1. Таким образом, процесс интеграции должен циклически проработать с n исходными векторами D^0. Это неизбежная плата за столь лёгкую начальную дифференциацию вершин в однородном графе.

2. *Выполнение последовательности шагов интеграции кодов.* Процесс интеграции кодов структурных различий при переходе от D^0 к D в общем случае разбивается на интервалы, связанные с появлением устойчивых групп, и может быть представлен в виде записи:

$$D^0 \overset{\Delta k_0}{\Rightarrow} D^{k_1} \overset{\Delta k_1}{\Rightarrow} D^{k_2} \overset{\Delta k_2}{\Rightarrow} ... \overset{\Delta k_{v-1}}{\Rightarrow} D^{k_v} \overset{\Delta k_v}{\Rightarrow} ... \overset{\Delta k_V}{\Rightarrow} D. \qquad (1)$$

В записи (1) Δk_v обозначает число шагов $D^k \Rightarrow D^{k+1}$, которые потребуется выполнить на v-м интервале при переходе от D^{k_v} к $D^{k_{v+1}}$. Индекс $v=1,2,...,V$ помечает очередной k-й шаг, в котором вектор D^k содержит устойчивые группы, V – число векторов D^k в цепи шагов от D^0 к D с устойчивыми группами. Например, фрагмент цепи, относящийся к интервалу Δk_2 в записи (1) может включать 3 шага ($D^{5_2} \Rightarrow D^6 \Rightarrow D^7 \Rightarrow D^{8_3}$), т. е. $\Delta k_2=3$. Важно, что общее число шагов в цепи $D^0 \Rightarrow D$ не может превысить число вершин n. При отсутствии в графе устойчивых групп цепь $D^0 \Rightarrow D$ состоит из одного интервала с $\Delta k_0 \le n$.

Для выполнения шага интеграции $D^k \Rightarrow D^{k+1}$ в инцидентах $F(e_i)$ графа G вместо номеров вершин e_i подставляются коды d_i^k вектора D^k. В полученных

таким образом инцидентах $F(d_i^k)$ коды упорядочиваются по возрастанию значений и для них определяются числовые эквиваленты. Преобразование инцидентов $F(d_i^k)$ в числовые эквиваленты ничем не отличается от определения числовых эквивалентов для строк матрицы A^2, приведенного в операции 1. Значение δ в данном случае принимается равным числу десятичных разрядов величины n.

Назначение кодов d_i^k начинается для вершин с минимальным значением кодов d_i^k. Если их в векторе D^k несколько, то соответствующие числовые эквиваленты упорядочиваются по возрастанию значений и для минимального из них назначается код $d_i^{k+1}=1$. Для следующих по величине значений назначаются коды 2, 3 и т. д. Далее среди оставшихся кодов d_i^k вновь выбирается минимальный, и процесс назначения кодов d_i^{k+1} соответствующим вершинам повторяется. В итоге формируется вектор D^{k+1}, и на этом выполнение шага интеграции заканчивается.

В зависимости от состояния вектора D^{k+1} процесс интеграции продолжается по следующим направлениям:

а) если $\{d_i^{k+1}\}=n$, то имеет место полная дифференциация вершин и вектор D^{k+1} запоминается в качестве вектора D;

б) если $\max\{d_i^{k+1}\}=\max\{d_i^k\}$, то вектор D_i^{k+1} содержит устойчивую группу и запоминается как вектор, помеченный очередным индексом $v+1$;

в) если $\max\{d_i^{k+1}\}>\max\{d_i^k\}$, то выполняется следующий шаг интеграции.

В итоге данная операция выполняет процесс интеграции по всем цепям шагов в очередном v-м интервале. При этом каждая цепь в зависимости от ситуации а) или б) завершается пополнением списков векторов типа D либо $D^{k_{v+1}}$.

3. *Анализ списков векторов D и $D^{k_{v+1}}$.* Наличие векторов в списке D означает, что, следуя записи (1), процесс интеграции выполнялся в последнем интервале цепи $D^0 \Rightarrow D$ либо в условиях отсутствия устойчивых групп, когда данная цепь включает один интервал с числом шагов Δk_0. Если вектор D^0 был получен путём введения виртуального различия, то список D может содержать несколько векторов. Не вдаваясь в теоретические вопросы существования в списке D нескольких векторов и возможного их различия, алгоритм, следуя требованию однозначности, должен выбрать один вектор.

Сначала в списке D выбирается вектор с наибольшим числом шагов интеграции. Если оказывается, что такой вектор один, то для него формируется полный инвариант. Если векторов несколько, то для каждого из них формируется совокупность инцидентов $F(d_i)$. Коды d_i в инцидентах $F(d_i)$ упорядочиваются по возрастанию значений, и для них определяются числовые эквиваленты. В результате каждому вектору соответствует множество числовых эквивалентов его инцидентов. Среди них выбирается вектор с множеством, содержащим минимальный числовой эквивалент. Если таких векторов окажется несколько, то среди них выбор делается по следующему минимальному числовому эквиваленту. Инциденторы $F(d_i)$ в выбранном векторе упорядочиваются по возрастанию значений d_i, и полученный вектор $P(G)=\{d_i(F(d_i))\}$ принимается в качестве полного инварианта графа G.

Может оказаться, что ряд векторов в списке D по составу числовых эквивалентов также неразличимы. Тогда в качестве вектора для формирования полного инварианта выбирается любой из них. В частности, если список D получен для однородного графа и начальная дифференциация вершин достигалась введением виртуального различия, то список D может содержать n неразличимых векторов. В этом случае мы имеем дело с симметричным графом.

Анализ векторов в списке $D^{k_{v+1}}$ выполняется при условии, что список D был пуст. Это означает, что процесс интеграции, выполненный для всех цепей шагов интервала, привёл к векторам, содержащим устойчивые группы. Выбор одного вектора из списка $D^{k_{v+1}}$ осуществляется аналогично тому, как это делалось для списка D. Отличие заключается в том, что вначале выбираются векторы, содержащие наибольшее значение d_i^k. Если таких векторов несколько, то выбор среди них производится по числовым эквивалентам. Если и в этом случае будет выбрано несколько неразличимых векторов, то все они принимаются в качестве векторов для продолжения процесса интеграции и поступают на вход операции 4. В частности, если список D пуст, а в списке $D^{k_{v+1}}$ оказалось n неразличимых векторов, то здесь также можно предположить, что имеет место симметричный граф.

4. *Выбор устойчивой группы для продолжения процесса интеграции.* В каждом векторе, выбранном в списке $D^{k_{v+1}}$, содержится одна или несколько устойчивых групп. Если имеем векторы с одной группой, то эта группа принимается в качестве исходной для продолжения интеграции в очередном интервале с помощью операций 2 и 3. При наличии в векторах нескольких устойчивых групп необходимо среди них выбрать одну. Выбор такой группы должен отвечать требованию однозначности. Для этого вначале выбираются группы минимальной мощности. Если выбирается одна группа, то она принимается в качестве исходной. При наличии нескольких групп минимальной мощности в качестве исходной выбирается группа с минимальным кодом d.

Процесс интеграции на данном интервале запускается для каждого выбранного вектора столько раз, сколько вершин содержится в исходной группе. Каждый раз, перед запуском процесса, очередной вершине группы назначается код виртуального различия, равный увеличенному на единицу максимальному коду вектора. После этого алгоритм возвращается к выполнению операций 2 и 3.

Алгоритм завершает работу в операции 3 после получения полного инварианта в виде вектора

$P(G)=\{d_i(F(d_i))\}$. Для изоморфных графов G и H алгоритм гарантирует равенство $P(G)=P(H)$ и, напротив, если графы неизоморфны, то полные инварианты $P(G)$ и $P(H)$ неравны. При этом инциденторы $F(d_i)$ полного инварианта $P(G)$ представляют абстрактную структуру графа G с вершинами, поименованными кодами интегральных описателей d_i.

Оценка предельного объёма вычислений при определении полного инварианта

Анализ содержания операций алгоритма указывает на отсутствие в нём больших переборов и сложных вычислений. Поэтому уже на этом основании можно говорить о высокой эффективности алгоритма. Тем не менее важно оценить предельные объёмы вычислений, которые зависят от размерности графа и особенности его структуры.

Объём вычислений алгоритма в основном определяется числом шагов процесса интеграции, выполняемых внутри каждого интервала (операция 2) и объёмами вычислений, затрачиваемых на один шаг. Подготовительная операция 1 является разовой и только для однородного графа выполняет вычисления, связанные с возведением в степень матрицы A и получением числовых эквивалентов. Операции 3 и 4, обеспечивающие переход от одного интервала к другому, также не требуют больших вычислений, а число интервалов всегда существенно меньше числа вершин.

При определении числа шагов будем исходить из того, что длина цепи, обозначим её Δk, при переходе от D^0 к D не может превышать n, т. е. $\Delta k = \sum_{v=0}^{V} \Delta k_v < n$. Наибольшее, суммарное по всем цепям, число шагов ΔK доставляют однородные графы – симметричные и несимметричные, для которых не удалось получить начальную дифференциацию с помощью маршрутной характеристики. В этом случае виртуальное различие вводится для каждой вершины графа. При отсутствии устойчивых групп общее число шагов составит $\Delta K=\Delta k_0 n$, а при их наличии величина ΔK может существенно снизиться.

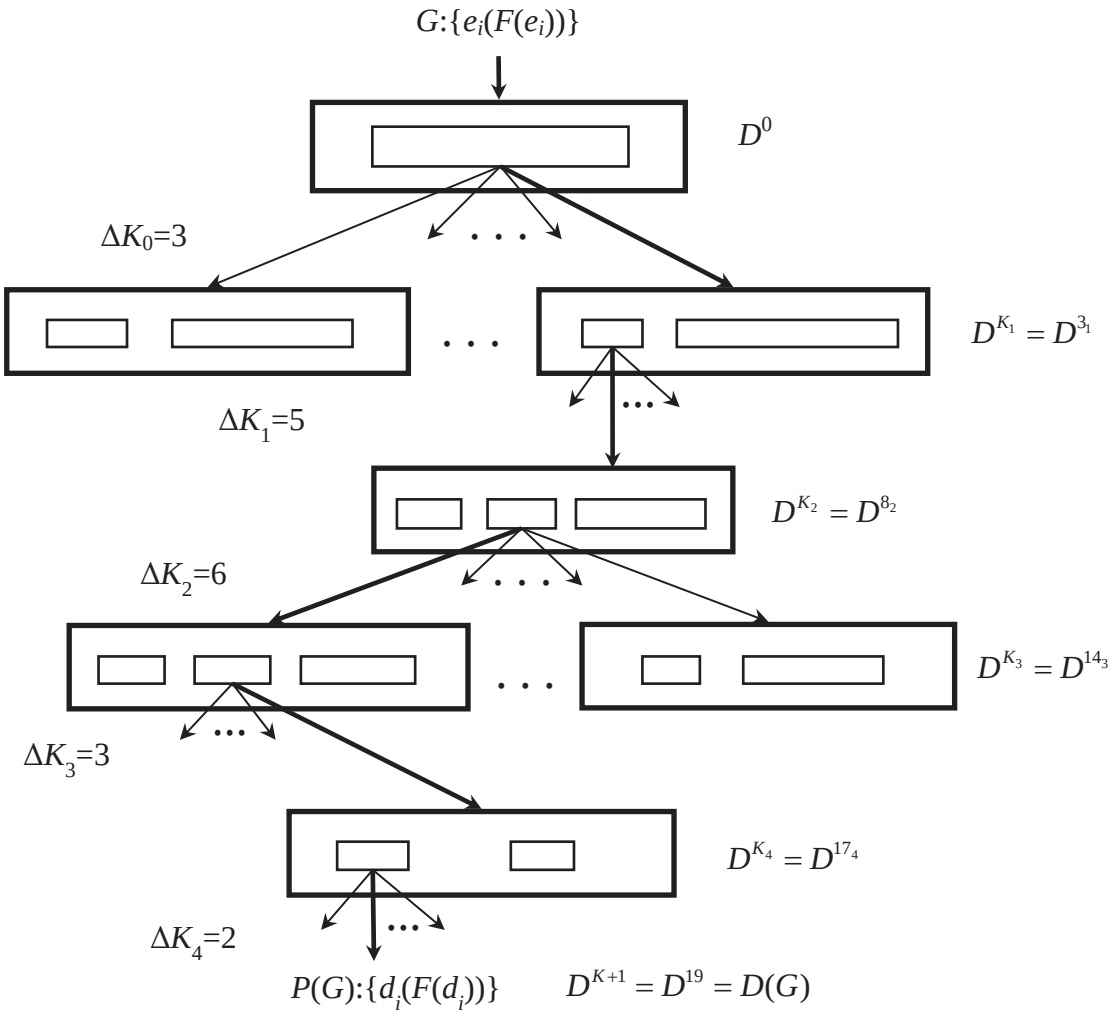

Рис. 3. *Пример схемы вычисления полного инварианта*

Fig. 3. *Complete invariant calculation pattern*

На рис. 3 приведён пример схемы вычисления полного инварианта для ситуации, когда граф G однородный и начальная дифференциация вершин достигается введением виртуального различия. Блоки на схеме соответствуют векторам D^k, а внутри них выделены устойчивые группы.

Процесс интеграции в примере на рис. 3 включает 5 интервалов. На каждом из них выбирается один вектор, и в нём выбирается одна устойчивая группа. Стрелки, выходящие из этой группы, отражают фрагменты цепей, выполняемых внутри интервала. Длинные стрелки указывают на векторы, полученные с наибольшим числом шагов. Выбор вектора среди них осуществляется по числовым эквивалентам. На рис. 3 такая ситуация показана для 1-го интервала с Δk_0=3 и для 3-го с Δk_2=6. Коротким стрелкам соответствуют векторы, которые отсеялись из-за меньшего числа шагов.

Число фрагментов цепей в каждом интервале совпадает с числом вершин в устойчивой группе, принятой в качестве исходной. Если для продолжения интеграции в списке $D^{k_{v+1}}$ было выбрано несколько неразличимых векторов, то число фрагментов цепей в интервале становится равным сумме вершин исходных устойчивых групп во всех этих векторах. Обозначим это число величиной n_v, v=1,2,...,V. В нашем примере n_0=n, т. к. граф G однородный, а начальная дифференциация вершин отсутствует. Тогда суммарная оценка предельного объёма вычислений $L(G)$ запишется в виде полинома:

$$L(G) = \tau(n,m)\sum_{v=0}^{V} n_v \cdot \Delta k_v. \qquad (2)$$

Здесь $\tau(n,m)$ – объём вычислений, затрачиваемых на выполнение одного шага интеграции для графа, содержащего n вершин и m рёбер.

Предельный объём оценки $L(G)$ по выражению (2) получается при самом неблагоприятном сценарии протекания процесса интеграции в однородном графе, когда n_0=n, V=1, а величина Δk_0 приближается к n. Но даже в этом, маловероятном, случае, оценка $L(G)<\tau(n,m)n^2$. Такой случай действительно является маловероятным, т. к. предполагает одновременное отсутствие устойчивых групп и начальной дифференциации вершин. Оба эти фактора противодействуют друг другу. Так, если не удаётся достичь начальной дифференциации, то граф характеризуется высокой симметричностью, что, в свою очередь, является непременным условием для существования устойчивых групп.

При наличии начальной дифференциации и (или) устойчивых групп суммарное число шагов интеграции, выполняемых алгоритмом, резко снижается. Как следует из рис. 3, появление устойчивых групп в векторах D^{3_1} приводит к тому, что в интервале Δk_1 число цепей $n<n/2$, т. к. выбирается группа с минимальным числом вершин. Кроме того, с ростом v размер групп не может возрастать [13]. Поэтому величины n_v для $v\geq1$ всегда будут в разы меньше числа вершин в графе.

Экспериментальные исследования алгоритма ISD-I с помощью программы GraphISD

При проведении экспериментальных исследований наряду с проверкой работоспособности алгоритма ISD-I большое внимание было уделено анализу его эффективности при вычислении полных инвариантов для однородных графов. Проверка работоспособности осуществлялась на ряде контрольных примеров графов небольшой размерности. Пример такого графа показан на рис. 4.

Начальная дифференциация вершин в данном примере выполнена на основе матрицы A^2 (рис. 4, *а*). Диагональные элементы в матрице A^2 принимаются равными нулю. В процессе независимой интеграции встретилось 3 вектора D^{k_v} (рис. 4, *б*), содержащих устойчивые группы. Число шагов в цепи Δk=5, а общее число шагов ΔK, которое потребовалось выполнить алгоритму, равно 11. На последнем интервале цепи интеграции получено 2 неразличимых вектора D (G). На рис. 4, *б* они обведены жирными линиями. Полный инвариант для данного графа после упорядочивания инциденторов в векторе $D(G)$ запишется в виде $P(G)$={1(2,3,9,10),2(1,4,5,8), 3(1,4,6,8),4(2,3,6,7),5(2,6,7,10),6(3,4,5,9), 7(4,5,9,10), 8(2,3,9,10), 9(1,6,7,8), 10(1,5,7,8)}.

На рис. 4, *в*, слева, приведён рассматриваемый граф G с исходной произвольной нумерацией вершин, а справа – абстрактная структура данного графа, восстановленная на основе полученного полного инварианта. Вершины абстрактной структуры помечены кодами d_i интегральных описателей, а рядом в скобках указаны соответствующие номера вершин e_i для первого и второго векторов $D(G)$. Приведена также подстановка двух автоморфизмов графа G, полученных на основе двух интегральных описателей неразличимых векторов $D(G)$, выделенных на рис. 4, *б* жирными линиями.

На рис. 2, рядом с симметричным графом G, приведен его полный инвариант $P(G)$. При этом начальная дифференциация вершин выполнялась на основе введения виртуального различия. Заметим, что в этом примере введение виртуального различия можно ограничить одной вершиной, т. к. заведомо известно о симметричности графа. При получении полного инварианта $P(H)$ для однородного несимметричного графа H (рис. 2) начальная дифференциация была достигнута на основе маршрутной характеристики по матрице A^2. В общем случае сведения о симметричности однородного графа отсутствуют, поэтому для начальной дифференциации вершин всегда используются маршрутные характеристики или виртуальные различия, которые вводятся последовательно для всех вершин графа.

Основные эксперименты по вычислению полных инвариантов с помощью алгоритма ISD-I проводились для графов разных мощностей и условий начальной дифференциации вершин. Все эксперименты в зависимости от вида начальной дифференциации были разделены на 3 группы:

а/a

$A=$

	1	2	3	4	5	6	7	8	9	10
1				1	1			1	1	
2							1	1	1	1
3				1		1		1	1	
4	1		1				1			1
5	1						1		1	1
6			1				1	1		1
7		1		1	1	1				
8	1	1	1			1				
9	1	1	1		1					
10		1		1	1	1				

$A^2=$

	1	2	3	4	5	6	7	8	9	10		D^0
1		2	3		1	1	2		1	2	1112223	4
2	2		2	2	3	3					22233	2
3	3	2			1	1	2	1		2	1112223	4
4		2			3	3		2	2		22233	2
5	1	3	1	3		2		1	1		1111233	3
6	1	3	1	3	2			1	1		1111233	3
7	2		2					2	2	4	22224	1
8			1	2	1	1	2		3	2	1112223	4
9	1			2	1	1	2	3		2	1112223	4
10	2		2				4	2	2		22224	1

б/b

e_i	$(F(e_i))$	D^0		D^{1_1}			D^{2_2}			D^{2_2}			D^3			D^{4_3}			D^3	
1	4589	4	2344	4	4	2344	4	4	2344	4	4	2344	4	2355	4	4	3446	5	3447
2	78910	2	1144	2	2	1445	2	2	1445	2	6*	1445	7	1556	7	2	1445	2	1446
3	4689	4	2344	4	4	2344	4	4	2344	4	4	2344	4	2355	4	4	3446	5	3447
4	13710	2	1144	2	2	1445	2	2	1445	2	2	1445	2	1446	2	6*	1445	7	1556
5	17910	3	1144	3	3	1445	3	3	1445	3	3	1445	3	1456	3	3	1445	3	1456
6	37810	3	1144	3	3	1445	3	3	1445	3	3	1445	3	1456	3	3	1445	3	1456
7	2456	1	2233	1	5*	2233	5	1	2233	1	1	2336	1	2337	1	1	2336	1	2337
8	1236	4	2344	4	4	2344	4	4	2344	4	4	3446	5	3447	5	4	2344	4	2355
9	1235	4	2344	4	4	2344	4	4	2344	4	4	3446	5	3447	5	4	2344	4	2355
10	2456	1	2233	1	1	2233	1	5*	2233	5	5	2336	6	2337	6	5	2336	6	2337

e_i	D^{4_3}			$\{d_i(F(d_i))\}$			$\{d_i(F(d_i))\}$		
1	5	5	4478	7	45910	5	3447	6	3459
2	2	2	1446	2	1458	2	1446	2	1458
3	5	5	3447	6	3459	5	4478	7	45910
4	7	7	1556	9	1678	7	1556	9	1678
5	3	8*	1456	10	1578	3	1456	3	1468
6	3	3	1456	3	1468	8*	1456	10	1578
7	1	1	2378	1	23910	1	2378	1	23910
8	4	4	2355	4	2367	4	2558	5	26710
9	4	4	2558	5	26710	4	2355	4	2367
10	6	6	2378	8	23910	6	2378	8	23910

(d_i)	e_i	e_j	(d_j)
(7)	1	1	(6)
(2)	2	2	(2)
(6)	3	3	(7)
(9)	4	4	(8)
(10)	5	5	(3)
(3)	6	6	(10)
(1)	7	7	(1)
(4)	8	8	(5)
(5)	9	9	(4)
(8)	10	10	(8)

в/c

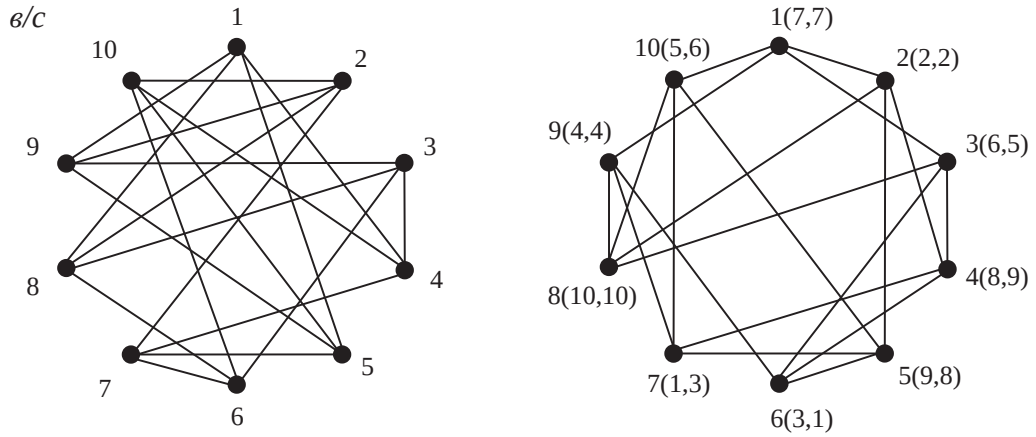

Рис. 4. *Пример работы алгоритма ISD-I*

Fig. 4. *ISD-I algorithm operation*

- однородные графы без начальной дифференциации вершин;
- однородные графы с начальной дифференциацией вершин на основе маршрутных характеристик, вычисленных по матрицам A^2;
- графы с начальной дифференциацией на основе степеней вершин.

Для проведения таких исследований программа GraphISD, реализующая алгоритм ISD-I, была дополнена генератором графов. Генератор способен формировать однородные графы с заданной мощностью n и степенью вершин s и неоднородные, когда указывается совокупность значений степеней s_i. Предусмотрена также возможность произвольной перенумерации вершин в сформированном ранее графе.

Эксперименты выполнялись для графов, содержащих от 100 до 1000 вершин, с шагом 100. Результаты экспериментов по каждой из трёх групп сведены в таблицу. Для каждого значения n=100, 200,..., 1000 в исследуемой группе генерировалось 2 серии графов – с s=0,1n и с s=0,2n. Для первых значений n каждая серия включала до 100 графов. С увеличением значений n размер серии постепенно уменьшался до 10 графов. Графы для группы 3 генерировались с двумя значениями степеней вершин s_1 и s_2. В таблице значения этих степеней указаны в скобках. При генерации графов этой группы вершины получали степень s_1 или s_2 с вероятностью 0,5, т. е. примерно в равных долях.

Таблица. *Результаты экспериментов*

Table. *Experimental results*

n	$s(s_1, s_2)$	Группа/Group						
		1		2			3	
		ΔK	$\tau(P)$	ΔK	$\tau(P)$	$\tau(A^2)$	ΔK	$\tau(P)$
100	10(10, 20)	405	38	1	6	16	14	8
	20(20, 40)	321	50	1	7	19	23	14
200	20(20, 40)	989	167	1	12	40	19	17
	40(40, 80)	632	362	1	21	42	18	28
300	30(30, 60)	933	694	1	27	72	18	25
	60(60, 120)	915	1253	1	39	65	22	47
400	40(40, 80)	1785	1665	1	48	125	10	31
	80(80, 160)	1246	2878	1	55	122	10	60
500	50(50, 100)	2082	3644	1	63	220	14	47
	100(100, 200)	1503	5865	1	76	201	10	89
600	60(60, 120)	1848	6177	1	82	350	10	68
	120(120, 240)	2434	13228	1	108	353	10	110
700	70(70, 140)	2105	11008	5	103	607	10	101
	140(140, 280)	2105	20116	9	142	606	14	160
800	80(80, 160)	2404	16202	5	117	826	18	127
	160(160, 320)	3242	32561	5	176	833	18	225
900	90(90, 180)	2703	23406	1	142	1204	16	149
	180(180, 360)	3603	48741	5	217	1200	16	285
1000	100(100, 200)	4001	31121	5	181	1699	10	182
	200(200, 400)	3018	63564	5	274	1709	18	380

Для каждого графа серии полный инвариант вычислялся с учётом начальной дифференциации: в 1-й группе – это введение виртуального различия; во 2-й – на основе маршрутной характеристики по матрице A^2; в 3-й на основе степеней s_1 и s_2. При этом фиксировались следующие величины: $\tau(P)$ – время вычисления полного инварианта $P(G)$ в миллисекундах; $\tau(A^2)$ – время на получение матрицы A^2 и начальную дифференциацию, также в миллисекундах; V – число векторов D^{k_v}, содержащих устойчивые группы; Δk – число шагов интеграции в цепи; ΔK – общее число шагов интеграции при вычислении $P(G)$.

В таблице приведены значения ΔK, $\tau(P)$ и для 2-й группы дополнительно $\tau(A^2)$. Эти значения взяты у одного из графов соответствующей серии, для которого $\tau(P)$ оказалось наибольшим. Оценки ΔK и $\tau(P)$ по таким графам дают более полное представление об эффективности алгоритма. Напротив, если оценивать работу алгоритма, например, по средним для серии значениям $\tau(P)$, то их наибольшие значения окажутся скрытыми.

Параметры V и Δk, соответствующие графам с наибольшими $\tau(P)$, в таблицу не внесены. В ходе экспериментов они принимали следующие значения: в 1-й группе V=1,2, Δk=6,...,9; во 2-й группе V=0,1,2, Δk=1,...,6; в 3-й группе V=1,...,5, Δk=4,...,13. Значения параметров V, Δk, ΔK по отношению к n и s и соответствующей величине $\tau(P)$ иногда оказывались неожиданными и трудноо бъяснимыми. Это особенно характерно для экспериментов в группах 2 и 3. Чтобы разобраться с такими ситуациями, нужно анализировать особенности графа и непосредственно процесс интеграции. Для этого требуются отдельные исследования и программные средства с соответствующими функциями. Одна из причин появления указанных ситуаций может быть обусловлена тем, что в результате генерации мы не знаем, получен связный граф или нет. Очевидно, что для графов с равными n и s параметры для связного графа будут существенно отличаться от параметров для несвязного.

Основная цель экспериментов заключалась в определении затрат времени на вычисление полных инвариантов для графов большой размерности, содержащих до 1000 вершин. Поэтому возможные особенности графов при их случайной генерации здесь не рассматривались. Не затрагивались также вопросы повышения эффективности алгоритма и его программной реализации. Здесь важно было оценить способность алгоритма вычислять полный инвариант $P(G)$ за приемлемое время $\tau(P)$ в том числе и для однородных графов. Характер зависимостей значений $\tau(P)$ от величин n и s, взятых из таблицы, представлен на рис. 5 в более наглядной форме. Здесь же (рис. 5, *г*) показана зависимость $\tau(A^2)$ от n.

Эксперименты проводились на компьютере со следующими характеристиками: Intel Core i7–4770 3,40 GHz, 16 GB RAM. Программа GraphISD реализована на языке Java. Оптимизировать работу программы можно несколькими способами: например, использованием языков более низкого уровня и распараллеливанием алгоритмов возведения матрицы в степень и вычисления полного ин-

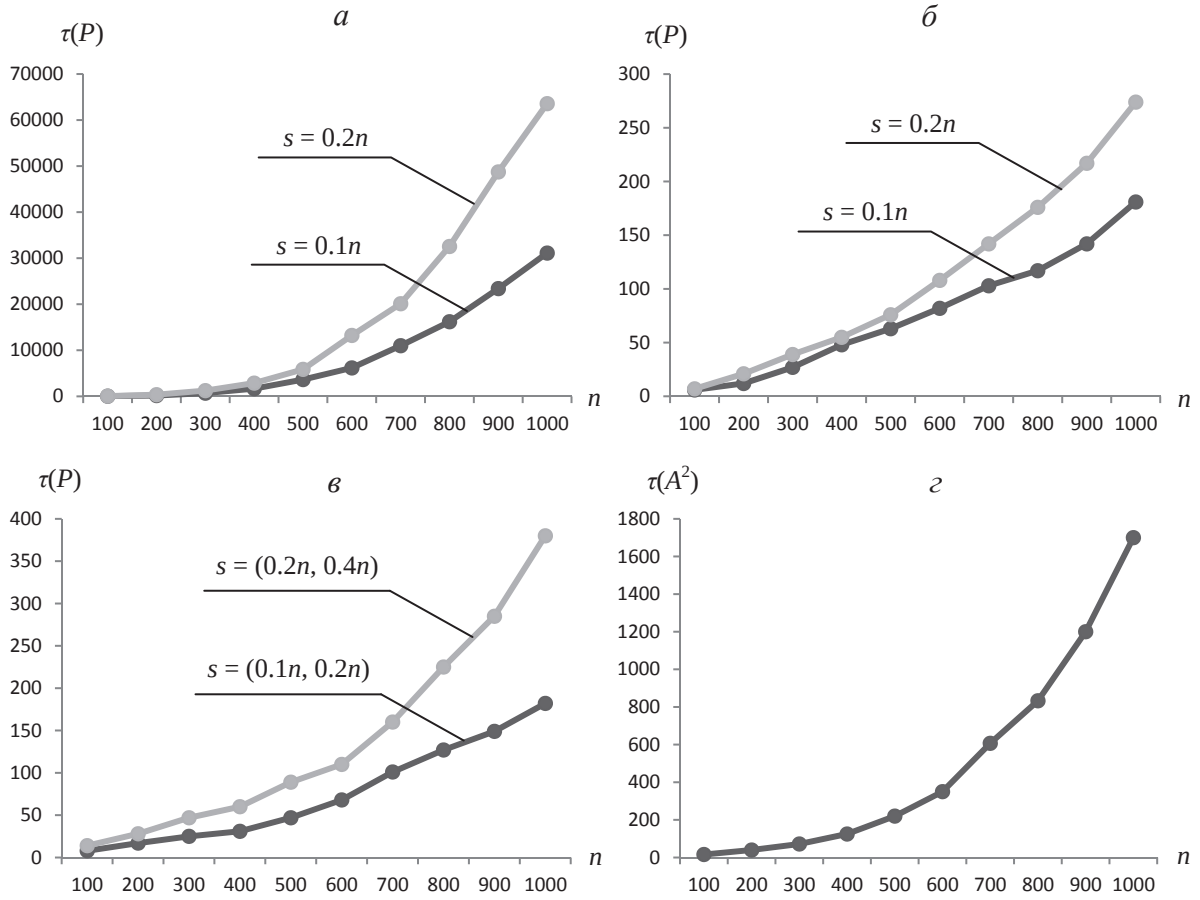

Рис. 5. *Зависимости τ(P) и τ(A²) от n*

Fig. 5. *Dependence of τ(P) and τ(A²) on n*

варианта. Для однородных графов в 1000 вершин наблюдалась предельная загрузка оперативной памяти. Поэтому в будущем для успешной работы с графами большой размерности на компьютерах меньшей мощности в программе будет предусмотрена возможность взаимодействия с внешней памятью.

Анализ результатов экспериментов, представленных в таблице и на рис. 5, даёт основание сделать следующие выводы.

1. Общее число шагов ΔK для однородных графов в условиях введения виртуального различия (группа 1 таблицы) существенно меньше предельного значения n^2, определяемого полиномом (2). Так, для n=100 величина ΔK примерно равна $0{,}04n^2$, а для n=1000 ΔK не превышает $0{,}004n^2$.
2. На кривой зависимости величины $\tau(P)$ от n по группе 1 (рис. 5, *а*) выделяются 2 интервала – до 500 вершин и от 500 до 1000. С увеличением n на 2-м интервале, в сравнении с 1-м, величина $\tau(P)$ растёт значительно быстрее. Можно предположить, что одна из причин ускорения роста $\tau(P)$ на 2-м интервале связана с дополнительными затратами времени на работу виртуальной машины Java с памятью (сборка мусора).
3. Применение маршрутной характеристики на основе матрицы A^2 для начальной дифференциации вершин (группа 2) резко снижает $\tau(P)$. Из таблицы следует, что для n=100 произошло снижение в 6–7 раз, а для n=1000 в 150–230 раз. При этом суммарное время $\tau(P)+\tau(A^2)$ по группе 2 также в разы меньше времени $\tau(P)$, полученного в условиях группы 1. Это хорошо видно и при сопоставлении кривых на рис. 5, *б*, *г* с кривой на рис. 5, *а*. Следует отметить также, что в ходе экспериментов по группе 2 не встретилось ни одного графа, для которого на основе A^2 не была бы достигнута начальная дифференциация вершин. Исходя из этого, в алгоритме принято правило, по которому для однородного графа всегда используется маршрутная характеристика на основе матрицы A^2. Если при этом начальная дифференциация не происходит, то алгоритм переходит к введению виртуальных различий, т. е. работает по условиям группы 1.
4. Эксперименты по вычислению полных инвариантов для неоднородных графов (группа 3) дали хорошие результаты, сопоставимые с результатами по группе 2 без учёта времени $\tau(A^2)$. Так, для графа с n=1000 и s_1=200, s_2=400 время $\tau(P)$ составило 380 мс. По данной группе экспери-

ментов при увеличении в 2 раза степеней s_1 и s_2 наблюдается увеличение $\tau(P)$ также примерно в 2 раза. Такая же тенденция наблюдалась и для экспериментов в группе 1. Это хорошо видно и по графикам на рис. 5, *в*, *а*, соответственно.

С помощью программы GraphISD также выполнялся ряд отдельных экспериментов, например, проверялось совпадение полных инвариантов у заведомо изоморфных графов. С этой целью граф G представлялся с разными нумерациями вершин путём случайной перестановки строк и столбцов в матрице A. Для этих нумераций вычислялись полные инварианты и проверялось их совпадение.

Генерация графов с низкими значениями степеней s, например s=2, порождала варианты графов, содержащих несколько компонент связности. При вычислении для таких графов полных инвариантов наблюдались вполне объяснимые различия значений параметров V, Δk, ΔK, $\tau(P)$.

Проводился также ряд экспериментов по вычислению полного инварианта для графов с n>1000. Например, для графа с n=2000, s_1=100, s_2=200 получены следующие результаты: V=3, Δk=9, ΔK=14, $\tau(P)$=410 мс, а для графа с n=2000, s_1=100, s_2=200, s_3=300, s_4=400: V=2, Δk=7, ΔK=10, $\tau(P)$=525 мс. В эксперименте для графа с n=10000, s_1=100, s_2=200, s_3=300, s_4=400 получены следующие результаты: V=7, Δk=17, ΔK=30, $\tau(P)$=4900 мс. При этом использовалось только 5 GB оперативной памяти. Это объясняется тем, что граф неоднородный и имеет низкие значения степеней по отношению к своей размерности.

Заключение

Возникновение идеи, положенной в основу метода независимой интеграции кодов структурных различий, стало для автора приятной неожиданностью. Действительно, трудно было уйти от вполне естественного правила кодирования, используемого при свободной и зависимой интеграции, когда каждой совокупности кодов инцидентора ставился в соответствие определенной код. При этом соответствие между инцидентором и кодом приходилось запоминать в виде системы кодирования. Такая система принималась в качестве эталона и должна была прилагаться к полному инварианту.

При независимой интеграции каждый граф рассматривается автономно. Эталонная система кодирования не формируется. Полный инвариант однозначно представляет только тот граф, для которого он вычислялся. Это стало возможным благодаря тому, что эталонную систему кодирования удалось заменить совокупностью правил выполнения независимой интеграции с соблюдением требования однозначности. Центральным в этой совокупности является правило, согласно которому кодированию подлежат не инциденторы, а места их расположения в упорядоченной последовательности. Конкретные значения кодов в инциденторах учитываются лишь на стадии формирования таких последовательностей.

Научные результаты, полученные в статье, основываются на работах [13, 14]. Метод независимой интеграции и алгоритм вычисления полного инварианта, так же как и в этих работах, излагается в статье без глубокого погружения в теоритические обоснования правил интеграции и исследования их свойств. Такие исследования еще предстоит выполнить в первую очередь для решения проблем, связанных с получением оценок сходства абстрактных структур графов.

В целом результаты экспериментальных исследований показали, что предложенный полный инвариант и алгоритм его вычисления способны работать с библиотеками графов большой размерности, инвариантно представлять графы в библиотеке в виде полных инвариантов $P(G)$, выделять изоморфные графы на основе сравнения векторов $P(G)$, формировать подстановки изоморфизма, при необходимости переходить к другим формам представления графов.

Работа выполнена в рамках государственного задания «Наука».

СПИСОК ЛИТЕРАТУРЫ

1. Зыков А.А. Основы теории графов. – М.: Вузовская книга, 2004. – 664 с.
2. Пролубников А.В. О новом полном инварианте ациклических графов // ПДМ. Приложение. – 2010. – № 3. URL: http://cyberleninka.ru/article/n/o-novom-polnom-invariante-atsiklicheskih-grafov (дата обращения: 04.02.2014).
3. Lindell S. A. Logspace Algorithm for Tree Canonization. Proc. of the 24th Annual ACM Symposium on the Theory of Computing. – New York, 1992. – P. 400–404.
4. Planar Graph Isomorphism is in Log-Space / S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, F. Wagner. 24th Annual IEEE Conference on Computational Complexity. – Paris, 15–18 July, 2009. – P. 203–214.
5. An efficient heuristic approach to detecting graph isomorphism based on combinations of highly discriminating invariants / M. Dehmer, M. Grabner, A. Mowshowitz, F. Emmert-Streib . Advances in Computational Mathematics. – 2013. – V. 39. – № 2. – P. 311–325.
6. McKay B.D., Piperno A. Practical Graph Isomorphism. II // J. Symbolic Computation. – 2014. – January. – V. 60. – P. 94–112.
7. Новый функционал информативности для анализа структуры химических графов / М. Дэмер, Ф. Эммерт-Штрайб, Ю.Р. Цой, К. Вармуза. // Известия Томского политехнического университета. – 2010. – Т. 316. – № 5. – С. 5–11.
8. Quantum Frontiers of Atoms and Molecules / M. Dehmer, F. Emmert-Streib, R.Y. Tsoy, K. Varmuza. – New York: Nova Publishing, 2011. – 673 p.
9. Dehmer M., Grabner M. The Discrimination Power of Molecular Identification Numbers Revisited // MATCH Commun. Math. Comput. Chem. – 2013. – V. 69. – № 3. – P. 785–794.
10. De Matos S., Dehmer M., Emmert-Streib F. Interfacing cellular networks of S. cerevisiae and E. coli: Connecting dynamic and genetic information // BMC Genomics. – 2013. – V. 14. – № 324. – P. 111–134.
11. Emmert-Streib F., Dehmer M. Networks for systems biology: conceptual connection of data and function // IET Systems Biology – 2011. – V. 5. – № 3. – P. 185–207.

12. Balasubramanian K., Parthasarathy K.R. In search of a complete invariant for graphs // Lect. Notes Malthem. – 1981. – V. 885. – P. 42–59.
13. Погребной В.К., Погребной Ан.В. Полиномиальный алгоритм вычисления полного инварианта графа на основе интегрального описателя структуры // Известия Томского политехнического университета. – 2013. – Т. 323. – № 5. – С. 152–159.
14. Погребной В.К., Погребной Ан.В. Исследование полиномиальности метода вычисления интегрального описателя структуры графа // Известия Томского политехнического университета. – 2013. – Т. 323. – № 5. – С. 146–151.
15. Кинг Р.Б. Химические приложения топологии и теории графов. – М.: Мир, 1987. – 560 с.
16. Baskin I., Skvortsova M. On the Basis of Invariants of Labeled Molecular Graphs // Chem. Inf. Comput. Sci. – 1995. – V. 35. – № 3. – P. 527–531.
17. Varmuza K. Chemometrics in Practical Applications. – Rijeka, Croatia: InTech, 2012. – 326 p.

Поступила 14.02.2014 г.

UDC 519.171.1

COMPLETE GRAPH INVARIANT AND ALGORITHM OF ITS COMPUTATION

Andrey V. Pogrebnoy,
Tomsk Polytechnic University, 30, Lenin Avenue,
Tomsk, 634050, Russia. E-mail: avpogrebnoy@gmail.com

The relevance *of the discussed issue is caused by the fact, that in graph theory, since the mid of the last century, all attempts to find the form of complete invariant and to develop the effective algorithm for its computation have been failed. The proposed solution of the problem will contribute to development of the methods of invariant representation and graphs structure analysis.*
The main aim of the study *is to form theoretical basis of independent integration method of codes of structural differences and to develop the effective algorithm for complete invariant computation.*
The methods *of the study are based on graph theory and the methods of free and dependent integration of codes of structural differences for obtaining integral descriptors of vertices of abstract graph structures.*
The results. *The author has proposed a new rule of assigning structural differences code to differentiate graph vertices. The rule is simple, it is represented as an independent encoding system and ensures the obtain of the integral structure descriptor (ISD), invariant with respect to the vertices original numbering. Based on this method the effective algorithm of complete graph computation was developed. It is shown, that even for the worst cases the computation complexity is limited by polynomial evaluation. Software GraphISD was written on Java language. It was used for experimental researches of the algorithm effectiveness. The experiments showed that the proposed complete graph invariant and algorithm of its computation are capable of working effectively with libraries of graph invariants, containing up to 5000 vertices, of distinguishing isomorphic graphs based on comparison of the complete invariants, of forming isomorphism substitutions and initial graph representations.*

Key words:
Complete graph invariant, abstract graph structure, homogeneous graph, integral structure descriptor, stable group of vertices, symmetric graph, graph isomorphism.

The research was carried out within the State task «Nauka».

REFERENCES

1. Zykov A.A. *Osnovy teorii grafov* [Basics of the graph theory]. Moscow, Vuzovskaya kniga, 2004. 664 p.
2. Prolubnikov A.V. *O novom polnom invariante atsiklicheskikh grafov* [On a new complete acyclic graph invariant]. PDM. Prilozhenie, 2010, no. 3. Available at: http://cyberleninka.ru/article/n/o-novom-polnom-invariante-atsiklicheskih-grafov (accessed 4 February 2014).
3. Lindell S. A. Logspace Algorithm for Tree Canonization. *Proc. of the 24th Annual ACM Symposium on the Theory of Computing*. New York, 1992. pp. 400–404.
4. Datta S., Limaye N., Nimbhorkar P., Thierauf T., Wagner F. Planar Graph Isomorphism is in Log-Space. *24th Annual IEEE Conference on Computational Complexity*. Paris, 15–18 July, 2009. pp. 203–214.
5. Dehmer M., Grabner M., Mowshowitz A., Emmert-Streib F. An efficient heuristic approach to detecting graph isomorphism based on combinations of highly discriminating invariants. *Advances in Computational Mathematics*, 2013, vol. 39, no. 2, pp. 311–325.
6. McKay B.D., Piperno A., Practical Graph Isomorphism, II. *J. Symbolic Computation*, January 2014, vol. 60, pp. 94–112.
7. Demer M., Emmert-Shtrayb F., Tsoy U.P., Varmuza K. Novy funktsional informativnosti dlya analiza strukturi khimicheskikh grafov [New informative functional for chemical graph structure analysis]. *Bulletin of the Tomsk Polytechnic University*, 2010, vol. 316, no. 5, pp. 5–11.
8. Dehmer M., Emmert-Streib F., Tsoy R.Y., Varmuza K. *Quantum Frontiers of Atoms and Molecules*. New York, Nova Publishing, 2011. 673 p.
9. Dehmer M., Grabner M. The Discrimination Power of Molecular Identification Numbers Revisited. *MATCH Commun. Math. Comput. Chem.*, 2013, vol. 69, no. 3, pp. 785–794.
10. De Matos S., Dehmer M., Emmert-Streib F. Interfacing cellular networks of S. cerevisiae and E. coli: Connecting dynamic and genetic information. *BMC Genomics*. 2013, vol. 14, no. 324, pp. 111–134.

11. Emmert-Streib F., Dehmer M. Networks for systems biology: conceptual connection of data and function. *IET Systems Biology*. 2011, vol. 5, no. 3, pp. 185–207.
12. Balasubramanian K., Parthasarathy K.R. In search of a complete invariant for graphs. *Lect. Notes Malthem.* 1981, vol. 885, pp. 42–59.
13. Pogrebnoy V.K., Pogrebnoy An.V. Polinomialny algoritm vychisleniya polnogo invarianta grafa na osnove integralnogo opisatelya structury [Polynomial algorithm of complete graph invariant computation on the basis of integral structure descriptor]. *Bulletin of the Tomsk Polytechnic University*, 2013, vol. 323, no. 5, pp. 152–159.
14. Pogrebnoy V.K., Pogrebnoy An.V. Issledovanie polinomialnostri metoda vychisleniya integralnogo opisatelya strukturi grafa [Research of polynomiality of computation method for the graph structure integral descriptor]. *Bulletin of the Tomsk Polytechnic University*, 2013, vol. 323, no. 5, pp. 146–151.
15. King R.B. *Khimicheskie prilozheniya topologii i teorii grafov* [Chemical applications of graphs topology and theory]. Moscow, Mir, 1987, 560 p.
16. Baskin I., Skvortsova M. On the Basis of Invariants of Labeled Molecular Graphs. *Chem. Inf. Comput. Sci*. 1995, vol. 35, no. 3, pp. 527–531.
17. Varmuza K. *Chemometrics in Practical Applications*. Rijeka, Croatia, InTech, 2012. 326 p.

Received: 14 February 2014.