МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Санкт-Петербургский государственный технологический институт
(технический университет)

Кафедра систем автоматизированного проектирования и управления

А.Ю. Рогов, В.И. Халимон, О.В. Проститенко

ГРАФОВЫЕ МЕТОДЫ АНАЛИЗА В ДИСКРЕТНОЙ МАТЕМАТИКЕ

Учебное пособие

Санкт-Петербург
2012

УДК 651

Рогов А.Ю., Графовые методы анализа в дискретной математике: учебное пособие / А.Ю. Рогов, В.И. Халимон, О.В. Проститенко, СПб: СПбГТИ(ТУ), 2012.- 88 с.

Рассматриваются разделы дискретной математики, связанные с методами теории графов. Даются матричные и списковые способы представления графов в ЭВМ и алгоритмы анализа графов при их компьютерной обработке. Изложение учебного материала сопровождается примерами и иллюстрациями описываемых задач теории графов.Учебное пособие формирует компетенции: **ОК-1** в части –владеет культурой мышления, способен к обобщению, анализу и восприятию информации, **ОК-10** в части – применяет методы математического анализа и теоретического исследования, **ПК-4** в части – умеет разрабатывать модели компонентов информационных систем и данных.

Учебное пособие предназначено для студентов и бакалавров, обучающихся по специальностям: 230100 – Информатика и вычислительная техника, 230102 – Автоматизированные системы обработки информации и управления, 230104 – Системы автоматизированного проектирования и управления, 220100 – Системный анализ, изучающих дисциплину «Дискретная математика», а также для студентов всех факультетов и слушателей факультета переподготовки кадров по новым направлениям науки и техники.

Рисунков – 41, формул – 26, библиогр. – 12 назв.

Рецензенты:

1. Санкт-Петербургский государственный университет технологии и дизайна, кафедра математики, к.т.н., доцент, Кикец Е.В.
2. В.В. Куркина, к.т.н., доцент кафедры автоматизации процессов химической промышленности Санкт-Петербургского государственного технологического института (технического университета)

Утверждено на заседании учебно-методической комиссии факультета Информатики и Управления 25.04.2012 протокол № 8.

Рекомендовано к изданию РИСо СПбГТИ(ТУ)

С О Д Е Р Ж А Н И Е

В В Е Д Е Н И Е

Учебное пособие посвящено изучению одного из разделов дискретной математики, связанного с формализованным представлением и анализом топологических моделей и структур объектов. Данный раздел включает в себя базовые разделы теории графов, широко используемой для структурного анализа различных систем и объектов.

Теория графов является разделом дискретной математики, который позволяет формально представлять структуры систем, в частности, химико-технологических объектов, и применять формализованные процедуры анализа структур с целью определения различных характеристик системы. Методы и понятия теории графов имеют хорошую наглядную интерпретацию и чёткое формальное представление.

Математический аппарат теории графов позволяет с определенной степенью детализации формально представить структуру системы и располагает большим количеством методов и алгоритмов формального анализа структуры, отыскания в ней частей, обладающих заданными структурными свойствами, вычисления характеристик систем исходя из особенностей её структуры. Все это делает графовые методы анализа очень эффективными при анализе химико-технологических объектов.

Теория графов, как правило, не используется отдельно, а является надстройкой или разделом других теорий, которые характерны тем, что в них так или иначе отражается содержательное представление о потоках вещества, энергии, информации, существующих в рассматриваемом технологическом объекте. В ряде надстроек графовая структура интерпретируется как описание сложного объекта или системы, делая упор на внутренние связи и зависимости между элементами.

Материал учебного пособия позволяет студентам любых специальностей ознакомится с базовыми понятиями и моделями теории, а также освоить математические методы формального анализа графов применительно к структурам технологических систем.

1 Понятие графа

Хотя смысл понятия структуры представляется интуитивно ясным, дать ему удовлетворительное определение не так-то легко. Поскольку структура – это часть системы, необходимо четко указать, какая именно часть, какие свойства и признаки системы являются структурными, а какие – нет. Ответы на эти вопросы зависят от целей исследования системы, поэтому под структурой обычно понимают совокупность тех свойств системы, которые являются существенными с точки зрения проводимого анализа объекта.

Для описания структур систем используются схемы и графы. Структурные схемы имеют наглядность и позволят включать информацию о большом числе структурных свойств системы, легко поддаются уточнению и конкретизации. Однако структурная схема с трудом поддается формализации и является скорее «мостиком», облегчающим переход от содержательного описания структуры системы к математическому описанию, чем действенным инструментом анализа структур. Другим вариантом представления структуры системы являются графы, которые позволяют формализовать процесс исследования инвариантных во времени свойств системы и использовать хорошо развитый математический аппарат теории графов, позволяющий компьютеризировать анализ структуры.

Граф представляет собой двойку вида:

$$G = (V, U) , \qquad (1)$$

где V – множество вершин графа ;

U – множество дуг графа.

Множество вершин V обычно представляет собой множество элементов системы, а множество дуг U – связи между элементами системы. Формально, множество дуг U представляет собой двухэлементое подмножество декартового произведения множества вершин V на себя, т.е. $U \subseteq V \times V$.

Каждый элемент множества U представляет собой двойку из элементов множества V вида $u=(v^{н},v^{к})$. Элемент множества вершин, стоящий на первом месте в этой двойке $\mathbf{v^{н}}$ называется ***вершиной-началом*** дуги **u**, а элемент, стоящий на втором месте $\mathbf{v^{к}}$ этой двойки – ***вершиной-концом*** дуги **u**. Если

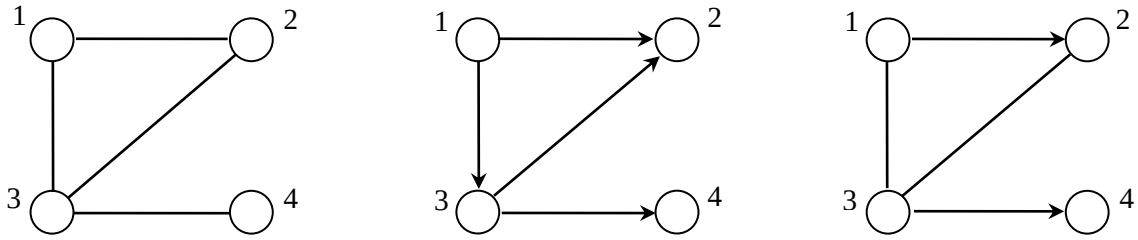
некоторые два элемента-вершины **v** и **v'** образуют некоторую двойку-дугу **u,** т.е. u=(v,v'), то говорят, дуга **u** ***инцидентна*** вершине **v** и **v'**, а вершины **v** и **v'** ***смежны***. Если некоторую двойку-дугу **u** образует одна и та же вершина **v**, т.е. на первом и втором месте находится та же самая вершина u=(v,v), то такую дугу называют ***петлей***. Если порядок следования элементов-вершин в двойке-дуге не существенен, т.е. неважно как записано (v,v') или (v',v), то говорят что ***дуга неориентированная***. Если порядок в двойках важен, т.е. (v,v') не тоже самое, что (v',v), то говорят что ***дуга ориентированная***. Если для некоторой ориентированной дуги (v,v') существует ориентированная дуга (v',v), то такие дуги называются ***противоположно направленными***. Неориентированная дуга всегда может быть представлена парой противоположно направленных дуг.

Если в множестве дуг U нет одинаковых двоек, т.е. каждая двойка встречается только один раз, то отношение вершин является единственным, в противном случае, т.е. если хотя бы одна двойка повторяется в множестве дуг U, то отношение вершин является множественным. При множественном отношении в множестве дуг U могут быть одинаковые пары u_1=(v,v'), u_2=(v,v'), u_3=(v,v'), ... , u_n=(v,v'), которые называются ***кратными дугами***. Другими словами, ***кратные дуги*** – это дуги, у которых начало, конец и направление совпадают, поэтому их также называют ***сонаправленными дугами***. Примеры показаны на рис. 1.

Рисунок 1 – Примеры дуг

В зависимости от ориентации дуг графы делятся на неориентированные, ориентированные и смешанные. ***Ориентированный граф*** – это граф, в котором все дуги ориентированные. ***Неориентированный граф*** – это граф, в котором все дуги неориентированные. ***Смешанный граф*** – это граф, в котором имеется хотя бы одна ориентированная и хотя бы одна неориентированная дуга. Примеры показаны на рис. 2.

Смешанные графы используются в основном на начальных этапах проектирования при составлении исходного описания системы. Это связано с тем, что анализ смешанных графов всегда сопряжен с определенными трудностями интерпретации направления дуг и усложняет алгоритмы анализа. Поэтому, на практике, смешанный граф преобразуют в ориентированный граф либо заданием определенной ориентации неориентированным дугам, либо заменой неориентированных дуг парой противоположно направленных ориентированных дуг.



а) Неориентированный б) Ориентированный в) Смешанный

Рисунок 2 – Примеры графов

Мощность множества вершин |V| – это количество вершин, и будет обозначаться буквой **P**. Количество вершин в графе также называется **порядком графа**. Мощность множества дуг |U| – это количество дуг, и будет обозначаться буквой **Q.** Подмножество вершин, смежных с данной вершиной ***v***, называется ***окрестностью вершины v***, и будет обозначаться буквой $\mathbf{\Gamma}_v$. В ориентированном графе для каждой вершины ***v*** выделяют ***окрестность по выходам*** $\mathbf{\Gamma}^-_v$ – это подмножество вершин смежных с данной вершиной ***v*** так, что данная вершина ***v*** является началом дуг, инцидентных этим вершинам, и ***окрестность по входам*** $\mathbf{\Gamma}^+_v$ – это подмножество вершин смежных с данной вершиной ***v*** так, что данная вершина ***v*** является концом дуг, инцидентных этим вершинам.

Например, для неориентированного графа на рис. 2-а имеем следующее: $\mathbf{\Gamma}_1=\{2,3\}$, $\mathbf{\Gamma}_2=\{1,3\}$, $\mathbf{\Gamma}_3=\{1,2,4\}$, $\mathbf{\Gamma}_4=\{3\}$; для ориентированного графа на рис. 2-б имеем следующее: $\mathbf{\Gamma}^-_1=\{2,3\}$, $\mathbf{\Gamma}^+_1=\varnothing$, $\mathbf{\Gamma}^-_2=\varnothing$, $\mathbf{\Gamma}^+_2=\{1,3\}$, $\mathbf{\Gamma}^-_3=\{2,4\}$, $\mathbf{\Gamma}^+_3=\{1\}$, $\mathbf{\Gamma}^-_4=\varnothing$, $\mathbf{\Gamma}^+_4=\{3\}$.

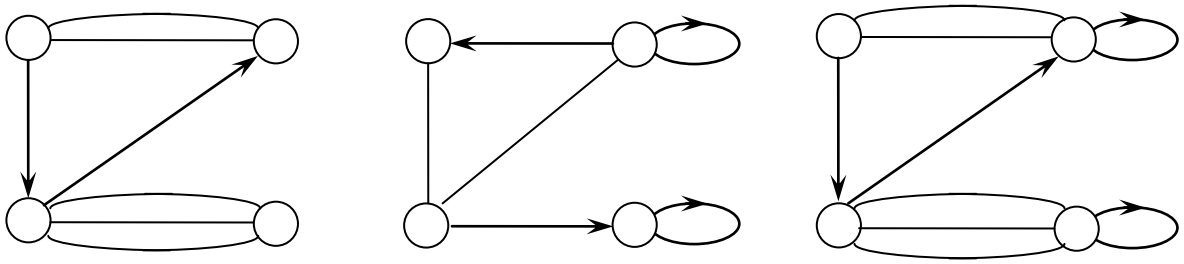
Количество дуг, инцидентных вершине, называется ***степенью вершины*** и будет обозначаться ρ. Учитывая, что дуги могут быть ориентированными, степень вершины разбивается на полустепень исхода ρ^- и полустепень захода ρ^+. ***Полустепень исхода*** – это количество дуг, исходящих из вершины, а ***полустепень захода*** – это количество дуг, заходящих в вершину. В неориентированном графе для каждой *i*-ой вершины принимается, что полустепень исхода и полустепень захода равны половине степени вершины, т.е. $\rho^-_i=\rho^+_i=\rho_i/2$. В ориентированном графе для каждой *i*-ой вершины степень вершины равна сумме полустепеней исхода и захода, т.е. $\rho^-_i+\rho^+_i=\rho_i$.

Например, для неориентированного графа на рис. 2-а имеем следующее: $\rho_1=2$, $\rho_2=2$, $\rho_3=3$, $\rho_4=1$; для ориентированного графа на рис. 2-б имеем следующее: $\rho^-_1=2$, $\rho^+_1=0$, $\rho_1=2$, $\rho^-_2=0$, $\rho^+_2=2$, $\rho_2=2$, $\rho^-_3=2$, $\rho^+_3=1$, $\rho_3=3$, $\rho^-_4=0$, $\rho^+_4=1$, $\rho_4=1$. В смешанном графе неориентированная дуга обычно трактуется как пара противоположно направленных дуг, поэтому для графа на рис. 2-в имеем следующее: $\rho^-_1=2$, $\rho^+_1=1$, $\rho_1=3$, $\rho^-_2=1$, $\rho^+_2=2$, $\rho_2=3$, $\rho^-_3=3$, $\rho^+_3=2$, $\rho_3=5$, $\rho^-_4=0$, $\rho^+_4=1$, $\rho_4=1$.

В ориентированном графе выделяют истоки и стоки. Вершина, в которую только заходят дуги, называется ***стоком*** или ***тупиком***. Вершина, из которой только исходят дуги, называется ***истоком*** или ***антитупиком***. Вершина, которая не смежна ни с одной вершиной графа называется ***изолированной***. Очевидно, что для стока полустепень исхода равна нулю, а полустепень захода ненулевая, т.е. $\rho^+_c>0$ и $\rho^-_c=0$; для истока полустепень захода равна нулю, а полустепень исхода ненулевая, т.е. $\rho^+_и=0$ и $\rho^-_и>0$; для изолированной вершины полустепени исхода и захода нулевые, т.е. $\rho^+_з=0$ и $\rho^-_з=0$.

Вершина, у которой полустепень исхода равна полустепени захода, т.е. $\rho^+=\rho^-$, называется ***регулярной***. Граф, содержащий только регулярные вершины, называется ***регулярным графом***. Граф, в котором степени всех вершин равны, т.е. $\rho_i=\rho$, называется ***однородным графом***. Граф, в котором для любой дуги (v_i,v_j) существует противоположная дуга (v_j,v_i), называется ***симметрическим графом***. Граф, в котором хотя бы для одной дуги (v_i,v_j) не существует противоположной дуги, называется ***антисимметрическим графом***.

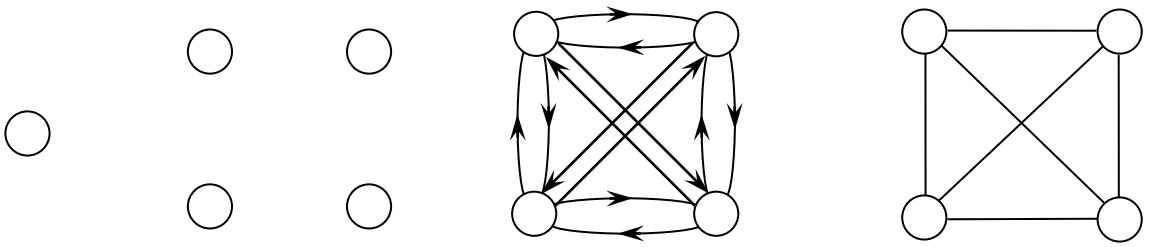
В зависимости от того, сколько вершин сопоставлено дуге и сколько раз производится это сопоставление, графы делятся на обычные графы, псевдографы, мультиграфы, мультипсевдографы. ***Обычный граф*** – это граф, в множестве дуг U которого нет петель и нет кратных дуг. ***Псевдограф*** – это граф, в множестве дуг U которого имеется хотя бы одна петля и нет кратных дуг. ***Мультиграф*** – это граф, в множестве дуг U которого имеются кратные дуги, но нет петель. ***Мультипсевдограф*** – это граф, имеющий обычные дуги, кратные дуги и петли. Все эти графы могут быть как ориентированными, так и неориентированными. Примеры показаны на рис. 3.



а) Мультиграф б) Псевдограф в) Мультипсевдограф

Рисунок 3 – Примеры графов

Существуют два особых случая. Если в графе отсутствуют дуги, т.е. U=∅, то граф называется ***пустым***, а если при этом граф содержит только одну вершину, то граф называется ***тривиальным***. Если в графе любая пара вершин связана дугой (смежна), т.е. U=V×V, то граф называется ***полным***. Примеры особых случаев показаны на рис. 4.



а) Тривиальный б) Пустой в) Полный ориентированный г) Полный неориентированный

Рисунок 4 – Примеры особых случаев графов

Подграфом G'=(V',U') исходного графа G=(V,U) называется граф, такой что множество его вершин и множество его дуг являются подмножествами исходного графа, т.е. V'⊂V и U'⊂U, и две вершины в нем G' смежны

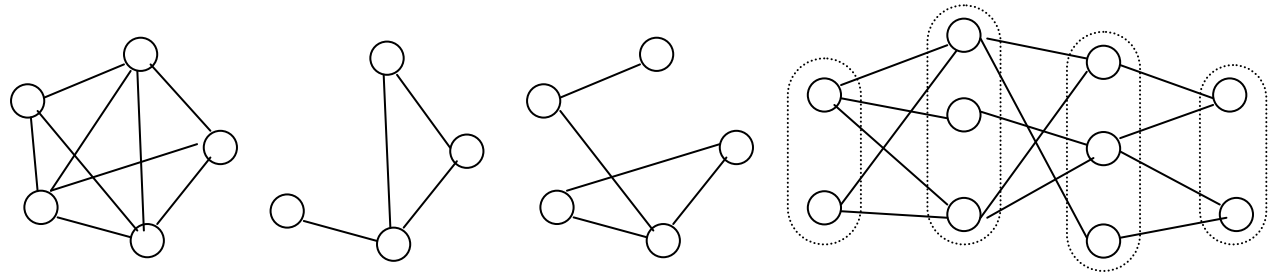
тогда и только тогда, когда они смежны в исходном графе G. Другими словами подграф получается удалением из графа некоторых вершин и дуг. Пример показан на рис. 5-б.

Суграфом G”=(V”,U”) исходного графа G=(V,U) называется граф, такой что множество его вершин совпадает с множеством вершин исходного графа, т.е. V”=V, а множество его дуг является подмножеством исходного графа, т.е. U”⊂U. Другими словами, суграф получается удалением из графа некоторых дуг. Пример показан на рис. 5-в.

Многодольным графом называется граф G=(V,U), такой что множество его вершин можно разбить на несколько непересекающихся подмножеств, т.е.

$$V_1 \subset V,\ V_2 \subset V,\ ...,\ V_n \subset V \ \text{ и } \ V_1 \cup V_2 \cup ... \cup V_n = V \ \text{ и } \ V_1 \cap V_2 \cap ... \cap V_n = \varnothing$$

таким образом, чтобы вершины каждого подмножества были не смежны между собой, а были смежны только с вершинами другого подмножества. Пример показан на рис. 5-г.



а) Исходный граф б) Подграф в) Суграф г) Многодольный граф

Рисунок 5 – Примеры графов

2 Понятие гиперграфа

До сих пор подразумевалось, что декартовое произведение множества вершин V на себя выполнено один раз. Однако, если множество вершин V перемножить несколько раз на себя, т.е.

$$U \subseteq V \times V \times V \times ... \times V ,$$

то в результате получается множество дуг состоящее не из двоек, а из троек, четверок, пятерок, и т.д., т.е. N-арных отношений. В этом случае дуга **u** будет инцидентна сразу N вершинам и N вершин будут одновременно смежны между собой. Понятие петли в этом случае сохраняется как дуги, в N-арном

отношении которой все вершины одинаковые:

$$u=(v, v, \ldots, v) \quad - \text{N-раз.}$$

Также сохраняется понятие кратности дуг, как одинаковые N-арные отношения в U. Единственное с чем возникают проблемы – это с ориентацией таких дуг. Можно полагать, что концом дуги является последняя вершина в списке, а все остальные первые, или что началом дуги является первая вершина в списке, а все остальные последними, или определить более сложное правило ориентации. Поэтому, как правило, все дуги гиперграфа считаются неориентированными. Графы, в которых используются N-арные отношения при N>2, образуют специальный класс, называемый ***гиперграфы***. Выделяют также обычные, мульти- и псевдо- гиперграфы.

Гиперграф также представляется как пара:

$$H = (A , B), \qquad (2)$$

где **A** – множество вершин A=V,

B – множество гиперребер, в котором каждое ребро $b_j \in B$ задается непосредственным перечислением вершин, ему инцидентных в N-арном отношении $B=U \subseteq V \times V \times \ldots \times V$.

Графически, множество вершин гиперграфа представляется также в виде кружочков, а вот множество дуг представляется в виде заштрихованных многоугольников, в углах которых расположены вершины. В общем случае, графическое представление гиперграфа не обладает наглядностью, поэтому для его задания используют различные матрицы.

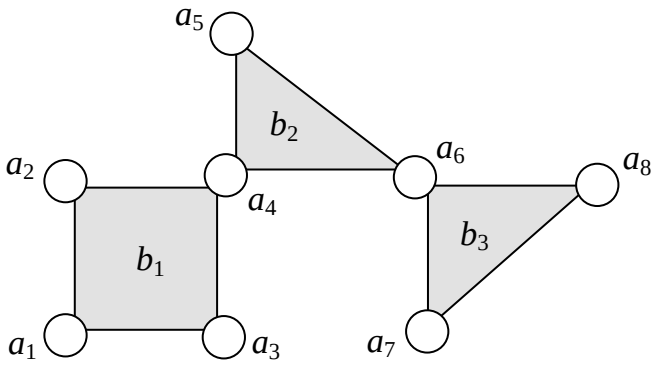


Рисунок 6 – Пример гиперграфа

Применение гиперграфов для описания структур систем весьма экономично. Особенно просто и эффективно их использование для задания

ориентированных иерархических графов без петель и контуров, в которых любые две вершины, являющиеся смежными, расположены на близлежащих уровнях иерархии. Каждый уровень такой системы представляется в виде гиперграфа, вершинами которого являются элементы системы нижнего уровня, а гиперребрами являются элементы системы текущего уровня иерархии. Таким образом, обычный граф системы непосредственно без каких-либо предварительных преобразований представляется как совокупность ***m–1*** гиперграфов, где ***m***-общее число уровней иерархии в графе.

Гиперграф на рис. 6 формально состоит из множества вершин A={a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8}, а множества гиперребер B={b_1, b_2, b_3}, где каждое ребро задается перечислением инцидентных вершин: b_1=(a_1, a_2, a_3, a_4), b_2=(a_4, a_5, a_6), b_3=(a_6, a_7, a_8).

Гиперграфы удобны для анализа структур и расчёта их топологических характеристик. Например, на рис. 6 вершина a_4 инцидентна одновременно двум гиперребрам b_1 и b_2, а вершина a_6 – гиперребрам b_2 и b_3. По этим вершинам гиперграф легко разбивается на три однореберных гиперграфа. Подобный приём позволяет провести декомпозицию структуры на отдельные мало связанные друг с другом подструктуры, что существенно облегчает исследование структуры системы в целом.

Изучение гиперграфов выходит за рамки данного курса и далее они рассматриваться не будут.

3 Способы представления графа

Графы задаются: матрицей смежности, матрицей инцидентности, списком дуг, массивом окрестностей вершин по выходам и входам графа. Пусть дан граф G = (V , U), и пусть P – число вершин графа, Q – число связей в графе. Определения матриц, которые даются ниже предполагают, что граф является ориентированным. При построении тех же матриц для неориентированного и смешанного графа каждое ребро представляется как две противоположно направленные дуги. Рассмотрим базовые представления графов на примере смешанного графа на рис. 7 (P=7, Q=12).

Неориентированное ребро (v_3,v_4) представляется как пара противоположно направленных дуг (v_3,v_4) и (v_4,v_3), поэтому Q=12+1=13.

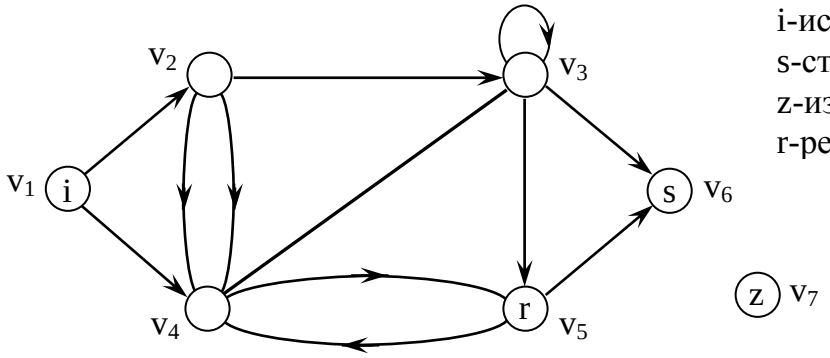


Рисунок 7 – Граф для описания представлений

Матрицей смежности $\mathbf{M^{см}}$ графа называется матрица, имеющая размерность **P**×**P**, каждый элемент $\mathbf{m^{см}_{ij}}$ которой определяется следующим образом:

$$\mathbf{m^{см}_{ij}} = \begin{cases} 1, \text{ если между i-ой и j-ой вершиной есть дуга в направлении от вершины i к вершине j;} \\ 0, \text{ если от вершины i к вершине j дуги нет.} \end{cases} \quad (3)$$

Если в графе имеется петля на вершине **i**, то на **i**-ом месте главной диагонали матрицы смежности стоит единица. Если в графе имеются кратные дуги, то вместо единицы в матрице ставится количество кратных дуг из вершины **i** в вершину **j**. Если в графе имеются противоположно направленные дуги, то элементы матрицы, расположенные симметрично относительно главной диагонали являются заполненными, т.е. $\mathbf{m^{см}_{ij}} > 0$ и $\mathbf{m^{см}_{ji}} > 0$. Строка матрицы смежности соответствует выходной окрестности $\mathbf{\Gamma^{-}}$ вершины, а столбец – входной окрестности $\mathbf{\Gamma^{+}}$ вершины. Сумма элементов по строке равна полустепени исхода ρ^{-} соответствующей вершины, а сумма элементов по столбцу равна полустепени захода ρ^{+} соответствующей вершины.

По матрице смежности легко находить изолированные, регулярные вершины, истоки и стоки графа. Вершине-истоку **i** в матрице смежности соответствует нулевой столбец **i** и не нулевая строка **i**. Вершине-стоку **s** соответствует нулевая строка **s** и не нулевой столбец **s**. Изолированной вершине **z** соответствует нулевой столбец **z** и нулевая строка **z** матрицы. Регулярной вершине **r** соответствует с строка **r** и столбец **r**, у которых сумма ячеек по строке равна сумме ячеек по столбцу.

Для графа представленного на рис. 7 матрица смежности имеет вид:

$$\mathbf{M}^{\text{см}} = \begin{array}{|ccccccc|l} 0 & 1 & 0 & 1 & 0 & 0 & 0 & \text{i} \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & \text{r} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{z} \\ \multicolumn{7}{c}{\text{i} \qquad\qquad \text{r} \quad \text{s} \quad \text{z}} & \end{array}$$

Данное представление является очень удобным и широко используется на практике. Однако, в случае, когда много вершин и мало дуг $\mathbf{Q} < \mathbf{P}\times\mathbf{P}$ это представление является не экономичным, поскольку требует много памяти для хранения нулей.

Матрицей инцидентности $\mathbf{M}^{\text{ин}}$ графа называется матрица, имеющая размерность $\mathbf{P}\times\mathbf{Q}$, каждый элемент $\mathbf{m}^{\text{ин}}_{\mathbf{ij}}$ которой определяется следующим образом:

$$\mathbf{m}^{\text{ин}}_{\mathbf{ij}} = \begin{cases} 1, & \text{если } \mathbf{i}\text{-ая вершина является концом } \mathbf{j}\text{-ой дуги ;} \\ -1, & \text{если } \mathbf{i}\text{-ая вершина является началом } \mathbf{j}\text{-ой дуги ;} \\ 0, & \text{если } \mathbf{i}\text{-ая вершина не инцидентна } \mathbf{j}\text{-ой дуге.} \end{cases} \qquad (4)$$

Как видно из определения матрицы, строки соответствуют вершинам графа, а столбцы – дугам. Если в графе имеется петля на вершине **i**, то в соответствующем столбце матрицы инцидентности в **j**-ой позиции стоит (+**1**), и отсутствует (**–1**), или наоборот. Если в графе имеются кратные дуги, то в матрице инцидентности имеются одинаковые столбцы. Если в графе имеются противоположно направленные дуги, то в матрице инцидентности имеются столбцы, в которых (+**1**) и (**–1**) переставлены местами. Количество (**–1**) в **i**-ой строке равно полустепени исхода $\rho^{-}_{\mathbf{i}}$ i-ой вершины, а количество (+**1**) в **i**-ой строке равно полустепени захода $\rho^{+}_{\mathbf{i}}$.

По матрице инцидентности изолированная вершина **z** определяется как нулевая строка **z**. Вершине-истоку **i** соответствует строка **i**, в которой имеются только (**–1**). Вершине-стоку **s** соответствует строка **s**, в которой имеются только (+**1**). Регулярной вершине **r** соответствует строка **r**, в которой количество (**–1**) равно количеству (+**1**).

Для графа представленного на рис. 7 матрица инцидентности имеет вид:

$$\mathbf{M}^{\text{см}} = \left|\begin{array}{cccccccccccc|c} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{i} \\ 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & -1 & -1 & 1 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & -1 & \text{r} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \text{s} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{z} \end{array}\right|$$

Эффективно использование данного представления для задания гиперграфов. В нём гиперребро задается столбцом матрицы, в котором ставятся 1 в тех строках, вершинам которых инцидентно данное гиперребро. Это представление также широко используется на практике. Оно не является экономичным для обычных и мульти-графов, поскольку требует много памяти для хранения нулей.

Списком дуг $\mathbf{M}^{\text{дуг}}$ графа называется матрица, имеющая размерность $\mathbf{2\times Q}$, каждый элемент $\mathbf{m}^{\text{дуг}}_{\mathbf{1k}}$ и $\mathbf{m}^{\text{дуг}}_{\mathbf{2k}}$ которой определяется следующим образом:

$$\left.\begin{array}{l} \mathbf{m}^{\text{дуг}}_{\mathbf{1k}} = \mathbf{i} \\ \mathbf{m}^{\text{дуг}}_{\mathbf{2k}} = \mathbf{j} \end{array}\right\} \text{, если в графе } \mathbf{k}\text{-ая дуга направлена из вершины } \mathbf{i} \text{ в вершину } \mathbf{j}. \qquad (5)$$

Как видно из определения матрицы, столбцы соответствуют дугам графа, первая строка – вершинам-начала дуг, вторая строка – вершинам-конца дуг. Если в графе имеется петля на вершине **t**, то в соответствующем столбце матрицы первый и второй элементы столбца равны, т.е. $\mathbf{m}^{\text{дуг}}_{\mathbf{1k}}=\mathbf{m}^{\text{дуг}}_{\mathbf{2k}}= \mathbf{t}$. Если в графе имеются кратные дуги, то в матрице имеются одинаковые столбцы. Если в графе имеются противоположно направленные дуги, то в матрице имеются столбцы, в которых номера вершины-начала **i** и вершины-конца **j** переставлены местами. Полустепень исхода $\rho^{-}_{\mathbf{t}}$ вершины **t** определяется как количество повторений номера **t** в первой строке, а полустепень исхода $\rho^{+}_{\mathbf{t}}$ – как количество повторений номера **t** во второй строке.

По списку дуг изолированная вершина **z** определяется как номер, не встречающийся ни в первой, ни во второй строке. Вершина-исток **i** определяется как номер **i**, встречающийся в первой и не встречающийся во второй строке. Вершина-сток **s** определяется как номер **s**, встречающийся во второй и не встречающийся в первой строке. Регулярная вершина **r** определяется как номер **r**, встречающийся одинаковое количество раз в обоих строках.

Для графа представленного на рис. 7 матрица списка дуг имеет вид:

$$\mathbf{M}^{\text{дуг}} = \left| \begin{array}{cccccccccccccc} 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 5 & 5 \\ 2 & 4 & 3 & 4 & 4 & 3 & 4 & 5 & 6 & 3 & 5 & 4 & 6 \end{array} \right|$$

Окрестность вершин по выходам (входам)

Задание графа окрестностью вершин включает массивы FO (FI) и KAO. Массив FO (FI) имеет размерность Q. В нём хранятся номера вершин, являющихся концами (началами) дуг. Массив KAO имеет размерность P+1. В нём хранятся номера элементов массива FO (FI), с которых начинается новая окрестность. Таким образом, i-ый элемент массива KAO хранит номер индекса элемента массива FO (FI), с которого начинается окрестность i-ой вершины графа.

Для графа представленного на рис. 7 представление массивом окрестностей по выходам имеет вид:

Массив окрестностей по входам строится аналогично и приведен ниже. Задание массивом окрестностей наиболее экономично для обычных графов при Q < P*P. Оно эффективно во многих алгоритмах анализа, поскольку анализ окрестности производится быстрее за счёт отсутствия нулей.

Массив окрестностей по входам:

По массивам окрестностей изолированная вершина **z** определяется как номер, не встречающийся в FO (FI) и разность элементов KAO равна нулю. Вершина-исток **i** определяется как номер **i**, встречающийся в FI, но разность элементов KAO равна нулю для **i**. Вершина-сток **s** определяется как номер **s**, встречающийся в FO, но разность элементов KAO равна нулю для **s**. Регулярная вершина **r** определяется как число повторений номера равное разности KAO.

4 Изолированные компоненты

Несвязанные элементы могут быть одной из ошибок проектирования системы потому, что любая система – это целостный организм. Элементы включаются в систему для того, чтобы они несли какую-то функцию в ней, взаимодействовали с другими элементами путем передачи потоков вещества, энергии, информации.

Выявление несвязанных элементов графовым методом состоит в поиске в графе системы изолированных вершин. Алгоритм поиска вычисляет степень каждой вершины графа и если она равна нулю, т.е. $\rho_v=0$, то найдена изолированная вершина. Для графа, представленного на рис. 8, вершина v_{15} – изолированная.

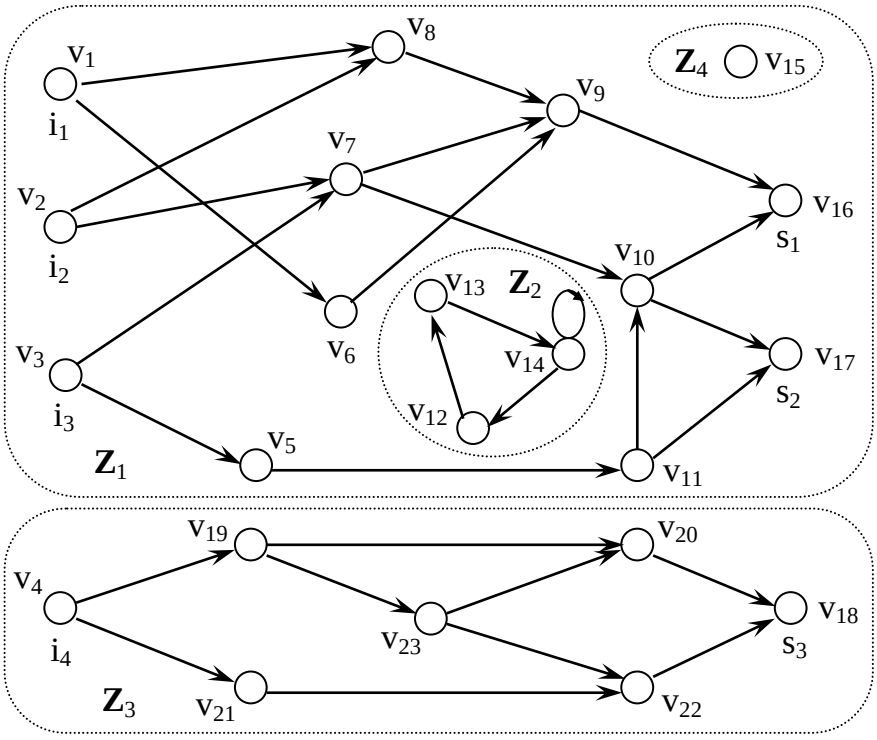


Рисунок 8 – Изолированные компоненты на графе

В системах могут встречаться части, состоящие из нескольких связанных между собой элементов, но при этом несвязанные с остальной частью системы. Такая часть, например, либо не обменивается информацией, либо логически не зависит от остальной части системы. На языке теории графов такие части системы называются изолированными подграфами или ***изолированными компонентами***. Частным случаем изолированной компоненты является изолированная вершина как одиночный элемент.

Если изолированная компонента преобразует входной поток системы в выходной поток или влияет на систему, то такая часть системы называется ***самостоятельной подсистемой***. Если же изолированная компонента не влияет на систему или не преобразует входной поток в выходной поток, то структура такой системы содержит ошибки в соединении элементов. Таким образом, выделение изолированных компонент является важным этапом структурного анализа.

Для отыскания изолированных компонент графа применяется алгоритм, использующий то свойство, что если две вершины связаны по входу или выходу, то они принадлежат одной компоненте. Этот алгоритм последовательно строит изолированные компоненты $\mathbf{Z}_1,...,\mathbf{Z}_k$. Для построения текущей изолированной компоненты $\mathbf{Z}_i$ из графа системы G=(V,U) выбирается вершина $\mathbf{v}\in V$, не принадлежащая ни к одной из уже найденных компонент и она помещается в текущую компоненту. Далее для этой вершины берется её окрестность $\mathbf{\Gamma}_v$ и все вершины окрестности помещаются в подмножество связанности **K**, за исключением тех вершин, которые уже отнесены к текущей компоненте $\mathbf{Z_i}$. Если подмножество связанности **K** не пусто, т.е. связанные вершины нашлись, то из него произвольно выбирается новая вершина **v**, она помещается в текущую компоненту $\mathbf{Z_i}$, выбирается её окрестность и вершины окрестности помещаются в подмножество **K.** Поиск связанных вершин повторяется до тех пор, пока подмножество **K** не окажется пустым, тогда текущая компонента $\mathbf{Z_i}$ будет сформирована.

Алгоритм поиска изолированных компонент следующий:

1. Положить N равному количеству вершин в графе N=|V| и i=1. Взять произвольную вершину $v\in V$ из графа.
2. Построить изолированную компоненту Z_i. Для этого:
   2.1. Поместить вершину v в текущую компоненту, т.е. $Z_i\cup\{v\}$.
   2.2. Найти все вершины ее окрестности Γ^+v и Γ^-v.
   2.3. Включить найденные вершины в подмножество связанности K, предварительно отбросив вершины, уже помещенные в текущую компоненту Z_i, т.е.

$$K=K\cup((\Gamma^+v\cup\Gamma^-v)-Z_i)$$

2.4. Если K=∅, т.е. вершин не найдено, то перейти к п.3, иначе выбрать из подмножества связанности вершину $v \in K$ и $K=K-\{v\}$, и перейти к п.2.1.

3. Уменьшить N на количество вершин в компоненте Z_i, т.е. $N=N-|Z_i|$.
4. Если все вершины отнесены к своим компонентам, т.е. N=0, то перейти к п.6, иначе перейти к п.5.
5. Выбрать вершину из графа, не принадлежащую ни к одной из изолированных компонент, т.е. $v \in (V - \cup^{i}_{j=1} Z_j)$, увеличить i=i+1.
   Перейти к п.2.
6. Прекратить поиск. Последовательность подмножеств $Z_1,...,Z_i$ являются искомыми изолированными компонентами системы.

Для графа, представленного на рис. 8, изолированными компонентами будут $Z_1=\{v_1, v_2, v_3, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{16}, v_{17}\}$, $Z_2=\{v_{12}, v_{13}, v_{14}\}$, $Z_3=\{v_4, v_{18}, v_{19}, v_{20}, v_{21}, v_{22}, v_{23}\}$, $Z_4=\{v_{15}\}$.

Рассмотрим на этом графе пример работы алгоритма отыскания изолированных компонент. Граф имеет 23 вершины, N=23, в качестве начальной вершины выберем v_1. Проиллюстрируем работу п.2 – ядра алгоритма – для нахождения первой компоненты Z_1, K=∅.

шаг 1. $Z_1=\{v_1\}$

Берем окрестности v_1: $\Gamma^{+}v_1=\{\}$, $\Gamma^{-}v_1=\{v_8,v_6\}$

Вычисляем: $K=\{v_8,v_6\}-\{v_1\}=\{\sout{v_8},v_6\}$, -выбираем v_8.

шаг 2. $Z_1=Z_1\cup\{v_8\}=\{v_1,v_8\}$

Берем окрестности v_8: $\Gamma^{+}v_8=\{v_1,v_2\}$, $\Gamma^{-}v_8=\{v_9\}$

Вычисляем: $K=\{v_6\}\cup\{\sout{v_1},v_2\}\cup\{v_9\}-\{\underline{v_1},v_8\}=\{\sout{v_6},v_2,v_9\}$, -выбираем v_6.

шаг 3. $Z_1=Z_1\cup\{v_6\}=\{v_1,v_6,v_8\}$

Берем окрестности v_6: $\Gamma^{+}v_6=\{v_1\}$, $\Gamma^{-}v_6=\{v_9\}$

Вычисляем: $K=\{v_2,v_9\}\cup\{\sout{v_1}\}\cup\{v_9\}-\{\underline{v_1},v_8,v_6\}=\{\sout{v_2},v_9\}$, -выбираем v_2.

шаг 4. $Z_1=Z_1\cup\{v_2\}=\{v_1,v_2,v_6,v_8\}$

Берем окрестности v_2: $\Gamma^{+}v_2=\{\}$, $\Gamma^{-}v_2=\{v_7,v_8\}$

Вычисляем: $K=\{v_9\}\cup\{v_7,\sout{v_8}\}-\{v_1,v_2,v_6,\underline{v_8}\}=\{\sout{v_9},v_7\}$, -выбираем v_9.

шаг 5. $Z_1=Z_1\cup\{v_9\}=\{v_1,v_2,v_6,v_8,v_9\}$

Берем окрестности v_9: $\Gamma^{+}v_9=\{v_6,v_7,v_8\}$, $\Gamma^{-}v_9=\{v_{16}\}$

Вычисляем: $K=\{v_7\}\cup\{\sout{v_6},v_7,\sout{v_8}\}\cup\{v_{16}\}-\{v_1,v_2,\underline{v_6},\underline{v_8},v_9\}=\{\sout{v_7},v_{16}\}$, -выбираем v_7.

шаг 6. $Z_1=Z_1\cup\{v_7\}=\{v_1,v_2,v_6,v_7,v_8,v_9\}$

Берем окрестности v_7: $\Gamma^+v_7=\{v_2,v_3\}$, $\Gamma^-v_7=\{v_9,v_{10}\}$

Вычисляем: $K=\{v_{16}\}\cup\{\sout{v_2},v_3\}\cup\{\sout{v_9},v_{10}\}-\{v_1,\underline{v_2},v_6,v_7,v_8,\underline{v_9}\}=\{\sout{v_{10}},\sout{v_3},v_{16}\}$,

-выберем сразу v_3 и v_{10}.

шаг 7. $Z_1=Z_1\cup\{v_3,v_{10}\}=\{v_1,v_2,v_3,v_6,v_7,v_8,v_9,v_{10}\}$

Берем окрестности: $\Gamma^+v_3=\{\}$, $\Gamma^-v_3=\{v_5\}$, $\Gamma^+v_{10}=\{v_7,v_{11}\}$, $\Gamma^-v_3=\{v_{16},v_{17}\}$

Вычисляем: $K=\{v_{16}\}\cup\{v_5\}\cup\{\sout{v_7},v_{11}\}\cup\{v_{16},v_{17}\}-\{v_1,v_2,v_3,v_6,\underline{v_7},v_8,v_9,v_{10}\}=$

$=\{v_5, v_{11},v_{16},v_{17}\}$.

Дальнейшие несложные вычисления показывают, что в $Z_1\cup K$ уже сформировалась искомая компонента, вопрос только в проверке оставшихся в множестве К вершин, которые все равно перекачуют в Z_1.

Анализируя результат видно, что изолированные компоненты Z_1 и Z_3 преобразуют входные потоки в выходные, и следовательно являются самостоятельными подсистемами. Компонента Z_2 не связывает входы с выходами, а компонента Z_4 состоит из изолированной вершины.

С задачей поиска изолированных компонент тесно связана задача проверки количества входов и выходов системы. Графовым методом эта задача сводится к поиску антитупиковых вершин (истоков) и тупиковых вершин (стоков). При нахождении истоков и стоков на графе используется то свойство, что для истоков полустепень захода нулевая, а для стоков полустепень исхода нулевая. Алгоритм поиска истоков и стоков вычисляет полустепени исхода и захода для всех вершин графа и проверяет условие $\rho^+_с>0$ и $\rho^-_с=0$ для вершин-стоков и условие $\rho^+_и=0$ и $\rho^-_и>0$ для вершин-истоков. Например, для графа, представленного на рис. 8, вершинами-истоками будут вершины с номерами v_1, v_2, v_3, v_4, а вершинами-стоками – v_{16}, v_{17}, v_{18}.

При анализе также важно, чтобы потоки проходили от истоков к стокам. Например, выходная вершина v_{16} достижима из входных вершин v_1, v_2, v_3, выходная вершина v_{17} достижима из входов v_2, v_3 и недостижима из входа v_1, выходная вершина v_{18} достижима из входа v_4.

5 Маршрут, классификация маршрутов, метод перечисления маршрутов

Анализ связей структуры является одним из основных этапов анализа любой системы. Исследование особенностей связей между элементами структуры направлено прежде всего на выявление в графе структуры путей и циклов, замкнутых подсистем, уровней системы, проверки достижимости одних элементов системы из других. Определения этих понятий опирается на понятие маршрута. Для демонстрации вводимых понятий и определений используем смешанный граф, представленный на рис. 9.

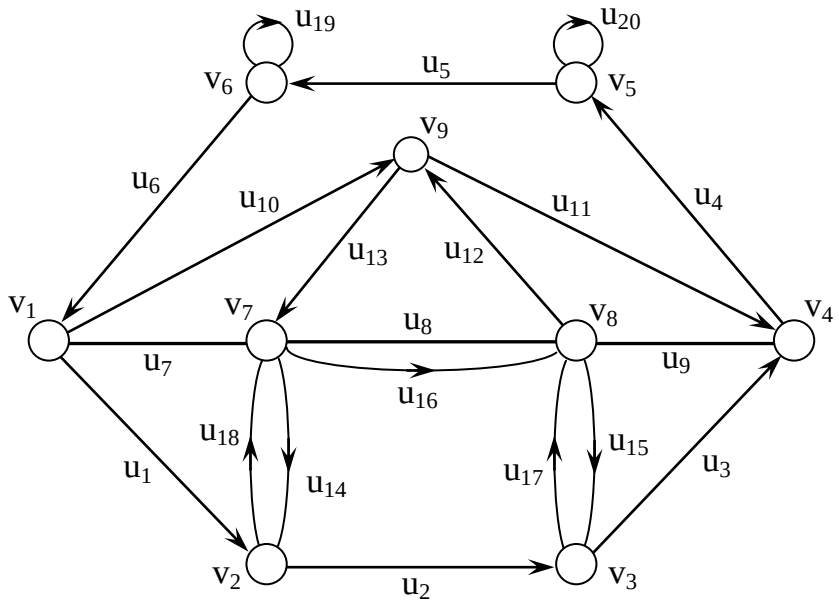


Рисунок 9 – Пример графа для маршрутов, путей, циклов

Маршрут – это чередующаяся последовательность вершин и дуг v_1,u_1 ; ... ; v_i,u_i ; ... ; v_k,u_k, обладающая тем свойством, что любая пара соседних элементов инцидентна. Вершина, стоящая на первом месте этой последовательности, называется ***начальной*** или началом маршрута. Вершина, стоящая на последнем месте этой последовательности, называется ***конечной*** или концом маршрута. ***Длиной маршрута*** называется количество входящих в него пар вершина-дуга. Например, для графа на рис. 9 последовательность $\{v_1, u_{10}, v_9, u_{13}, v_7, u_8, v_8, u_9, v_4\}$ будет произвольным маршрутом длиной 4, а последовательность $\{v_1, u_7, v_8, u_{12}, v_3, u_3, v_4\}$ маршрутом не является, поскольку не все рядом стоящие элементы в ней инцидентны.

Маршрут называется ***элементарным***, если в нём все вершины различны, т.е. никакая вершина не повторяется, кроме быть может начальной вершины, повторяющейся один раз в конце маршрута. В противном случае, маршрут называется ***неэлементарным***. Например, для графа на рис. 9 последовательность $\{v_1, u_{10}, v_9, u_{13}, v_7, u_{16}, v_8, u_{15}, v_3, u_3, v_4\}$ является элементарным маршрутом, а последовательность $\{v_1, u_{10}, v_9, u_{13}, v_7, u_{16}, v_8, u_{12}, v_9, u_{11}, v_4\}$ является неэлементарным маршрутом, поскольку повторяется вершина v_9.

Маршрут называется ***простым***, если в нём все дуги различны, т.е. никакая дуга не повторяется. В противном случае, маршрут называется ***составным***. Например, для графа на рис. 9 последовательность $\{v_1, u_1, v_2, u_{18}, v_7, u_{16}, v_8\}$ является простым маршрутом, а последовательность $\{v_1, u_1, v_2, u_{18}, v_7, u_{14}, v_2, u_{18}, v_7, u_{16}, v_8\}$ является составным маршрутом, поскольку в нём повторяется дуга u_{18}. Очевидно, что для обычных графов если маршрут является составным, то он также является неэлементарным.

Маршрут называется ***ориентированным***, если в нём все дуги ориентированны и ориентация дуг совпадает, т.е. вершина-конец текущей дуги является вершиной-началом следующей дуги в последовательности. В противном случае, маршрут называется ***неориентированным***. Например, для графа на рис. 9 последовательность $\{v_1, u_7, v_7, u_8, v_8, u_9, v_4\}$ является неориентированным маршрутом, а последовательность $\{v_1, u_1, v_2, u_2, v_3, u_3, v_4\}$ является ориентированным маршрутом.

В теории графов вводится понятие ***пустой маршрут*** – это пустая последовательность, не содержащая вообще пар вершина-дуга. Пустой маршрут будет обозначаться $\varnothing s=\{\}$ и его длина $|\varnothing s|=0$. Пустого маршрута на графе нет, но он является абстракцией, используемой в алгоритмах анализа.

Граф, в котором любая пара вершин соединена непустым маршрутом, называется ***связанным***, в противном случае граф называется ***несвязанным***.

Очевидно, что если первый маршрут заканчивается на вершине, с которой начинается второй маршрут, то второй маршрут можно присоединить к первому маршруту и тем сам получить новый маршрут, который по длине равен сумме длин обоих маршрутов. Новый маршрут сначала повторяет первый маршрут, а затем переходит во второй маршрут. Для формализации

этого действия в теории графов вводится операция композиции маршрутов.

Пусть $s_i=\{v_{i1},u_{i1}, v_{i2},u_{i2}, \dots ,v_{in},u_{in},v_k,\}$ маршрут длины n и $s_j=\{v_l, u_{j1},v_{j1}, u_{j2},v_{j2}, \dots , u_{jm},v_{jm},\}$ маршрут длины m. Введем бинарную **операцию композиции** над двумя маршрутами $(*,s_i,s_j)$ следующим образом:

$$s_i*s_j = \begin{cases} \{v_{i1},u_{i1}, \dots,v_{in},u_{in}, v_k, u_{j1},v_{j1}, \dots, u_{jm},v_{jm},\}, \text{ если } v_k=v_l, \text{ т.е.} \\ \text{конец первого маршрута совпадает с началом второго;} \\ \varnothing s \text{ – в противном случае (пустой маршрут).} \end{cases} \quad (6)$$

В результате композиции получается новый маршрут длины $n+m$. Операция композиции обладает следующими свойствами:

1. Умножение на ноль $\varnothing s * s = s * \varnothing s = \varnothing s * \varnothing s = \varnothing s$
2. Ассоциативна $s_1*(s_2*s_3) = (s_1*s_2)*s_3$
3. Не коммутативна $s_1*s_2 \neq s_2*s_1$

Например, для графа на рис. 9 композицией маршрута $\{v_1,u_{10}, v_9,u_{13}, v_7\}$ длины 2 с маршрутом $\{v_7, u_{16},v_8, u_{15},v_3, u_3,v_4\}$ длины 3 будет маршрут $\{v_1,u_{10}, v_9,u_{13}, v_7,u_{16}, v_8,u_{15}, v_3,u_3, v_4\}$ длины 5.

Вообще, в произвольном графе из одной вершины i в другую вершину k может существовать несколько маршрутов одинаковой длины n. Например, в графе на рис. 9 из вершины v_1 в вершину v_8 можно провести такие маршруты длины 3: $\{v_1,u_1,v_2,u_2,v_3,u_{17},v_8\}$, $\{v_1,u_1,v_2,u_{18},v_7,u_{16},v_8\}$, $\{v_1,u_{10},v_9,u_{13}, v_7,u_{16},v_8\}$. С другой стороны, из вершины v_8 в вершину v_4 можно провести такие маршруты длины 2: $\{v_8,u_{12},v_9,u_{11},v_4\}$ и $\{v_8,u_{15},v_3,u_3,v_4\}$. Далее, из этих маршрутов, используя операцию композиции *, можно построить 6 различных маршрутов из вершины v_1 в вершину v_4 длины 5. Таким образом, возникает необходимость формальной работы с множествами маршрутов одинаковой длины.

Маршрут длины n из вершины i в вершину k обозначим через $s^{n,i\to k}$, а маршрут длины m из вершины l в вершину j – через $s^{m,l\to j}$, и объединим их в множества. Пусть даны два множества: $S^n_{ik}=\{s_1^{n,i\to k},\dots, s_h^{n,i\to k},\dots, s_H^{n,i\to k}\}$ – множество, состоящее из H маршрутов длины n из вершины i в вершину k и $S^m_{lj}=\{s_1^{m,l\to j},\dots, s_t^{m,l\to j},\dots, s_T^{m,l\to j}\}$ – множество, состоящее из T маршрутов длины m из вершины l в вершину j. Введем бинарную операцию композиции двух множеств маршрутов $(\circ,S^n_{ik},S^m_{lj})$ следующим образом:

$S^{n}_{ik} \circ S^{m}_{lj} = \{(s_h^{n,i\to k} * s_t^{m,l\to j}) \mid s_h^{n,i\to k} \in S^{n}_{ik}$ и $s_t^{m,l\to j} \in S^{m}_{lj}$ и $k=l$, где $h=1,..,H$ и $t=1,..,T\}$ (7)

Другими словами, это множество маршрутов, полученных композицией всех пар маршрутов первого и второго множеств таким образом, что новый маршрут начинается с маршрута из первого множества, и заканчивается маршрутом из второго множества. Получающееся в результате этой операции $\circ$ множество маршрутов будет состоять из $H*T$ маршрутов длины $n+m$ из вершины i в вершину j, т.е. множество:

$$S^{n+m}_{ij} = \{s_1^{n+m,i\to j}, ..., s_z^{n+m,i\to j}, ..., s_Z^{n+m,i\to j}\}, \text{ где } z=1,...,Z=H*T. \quad (8)$$

Надо отметить, что наличие общей вершины у маршрутов первого множества S^{n}_{ik} на конце и маршрутов второго множества S^{m}_{lj} в начале обязательно, т.е. $k=l$, в противном случае в результате композиции по каждой паре маршрутов получится пустой маршрут: $s_h^{n,i\to k} * s_t^{m,l\to j} = \varnothing s$ при $k\neq l$, в следствии чего результирующее множество маршрутов окажется пустым $S^{n+m}_{ij}=\varnothing$. Операция композиции обладает следующими свойствами:

1. Умножение на ноль $\varnothing \circ S^{n}_{ij} = S^{n}_{ij} \circ \varnothing = \varnothing$
2. Ассоциативна $(S^{n}_{ik} \circ S^{r}_{kl}) \circ S^{m}_{lj} = S^{n}_{ik} \circ (S^{r}_{kl} \circ S^{m}_{lj})$
3. Не коммутативна $S^{n}_{ij} \circ S^{m}_{ij} \neq S^{m}_{ij} \circ S^{n}_{ij}$
4. Мощность множества $|S^{n+m}_{ij}| = |S^{n}_{ik}| * |S^{m}_{kj}|$

Так, для графа на рис. 9, имеем:

$S^{3}_{v1,v8} = \{ s_1^{3,v1\to v8} = \{v_1,u_1,v_2,u_2,v_3,u_{17},v_8\},\ s_2^{3,v1\to v8} = \{v_1,u_1,v_2,u_{18},v_7,u_{16},v_8\},$
$s_3^{3,v1\to v8} = \{v_1,u_{10},v_9,u_{13},v_7,u_{16},v_8\} \},\quad H=3$

$S^{2}_{v8,v4} = \{ s_1^{2,v8\to v4} = \{v_8,u_{12},v_9,u_{11},v_4\},\ s_2^{2,v8\to v4} = \{v_8,u_{15},v_3,u_3,v_4\} \},\ T=2$

В результате применения операции композиции множеств маршрутов получаем следующие множество $S^{5}_{v1,v4}$ маршрутов из вершины v_1 в вершину v_4 длины 5:

$S^{5}_{v1,v4} = S^{3}_{v1,v8} \circ S^{2}_{v8,v4} = \{$
$(s_1^{3,v1\to v8} * s_1^{2,v8\to v4}) = \{v_1,u_1, v_2,u_2, v_3,u_{17}, v_8,u_{12}, v_9,u_{11}, v_4\},$
$(s_2^{3,v1\to v8} * s_1^{2,v8\to v4}) = \{v_1,u_1, v_2,u_{18}, v_7,u_{16}, v_8,u_{12}, v_9,u_{11}, v_4\},$
$(s_3^{3,v1\to v8} * s_1^{2,v8\to v4}) = \{v_1,u_{10}, v_9,u_{13}, v_7,u_{16}, v_8,u_{12}, v_9,u_{11}, v_4\},$
$(s_1^{3,v1\to v8} * s_2^{2,v8\to v4}) = \{v_1,u_1, v_2,u_2, v_3,u_{17}, v_8,u_{15}, v_3,u_3, v_4\},$
$(s_2^{3,v1\to v8} * s_2^{2,v8\to v4}) = \{v_1,u_1, v_2,u_{18}, v_7,u_{16}, v_8,u_{15}, v_3,u_3, v_4\},$
$(s_3^{3,v1\to v8} * s_2^{2,v8\to v4}) = \{v_1,u_{10}, v_9,u_{13}, v_7,u_{16}, v_8,u_{15}, v_3,u_3, v_4\} \quad \}.$

Это множество состоит из $Z=H*T=3*2=6$ маршрутов.

Теперь, представим, что в некотором графе G=(V,U) порядка P>1 найдены все возможные маршруты длины n из вершины v_i в вершину v_j и они помещены в множество маршрутов, обозначим его через C^n_{ij}. Таким образом, в графе любое множество S^n_{ij} есть подмножество C^n_{ij}, т.е. $S^n_{ij} \subset C^n_{ij}$. Имея два множества всех возможных маршрутов: C^n_{ik} – длины *n* из вершины *i* в вершину *k* и C^m_{kj} – длины *m* из вершины *k* в вершину *j*, используя операцию композиции ° этих двух множеств, т.е. $C^n_{ik} \circ C^m_{kj}$, тем не менее невозможно получить C^{n+m}_{ij} – множество всех возможных маршрутов длины *n+m* из вершины *i* в вершину *j*. Это связано с тем, что выполняя операцию композиции даже множеств всех возможных маршрутов, в результате получается только множество тех маршрутов из вершины *i* в вершину *j* длиной *n+m*, которые проходят через общую вершину *k*, т.е. $S^{n+m}_{ij}=C^n_{ik} \circ C^m_{kj}$, а в графе могут существовать еще и другие пути из вершины *i* в вершину *j*, проходящие через другие вершины. Например, на вершине *l*, *l≠k*, могут заканчиваться маршруты длины *n* из вершины *i* множества C^n_{il} и начинаться маршруты длины *m* в вершину *j* множества C^m_{lj}, а операция композиции даст множество маршрутов длины *n+m* из вершины *i* в вершину *j*, т.е. $S^{n+m}_{ij}=C^n_{il} \circ C^m_{lj}$, которое в прочем тоже не будет содержать всех возможных маршрутов.

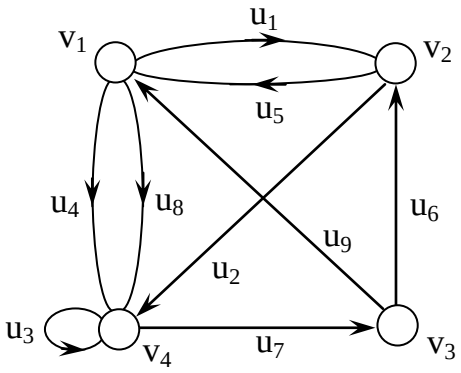
Чтобы получить множество всех возможных маршрутов C^{n+m}_{ij} длины *n+m* из вершины *i* в вершину *j*, нужно выполнить операцию композиции над всеми множествами всех возможных маршрутов, начинающихся с вершины *i* и всеми множествами всех возможных маршрутов, заканчивающихся на вершине *j*, по всем вершинами графа, а затем объединить все множества, полученные в результате применения операции композиции °, т.е.

$$C^{n+m}_{ij} = \cup^{p}_{k=1} C^n_{ik} \circ C^m_{kj}, \text{ где } k=1,...,P, \text{ и } P - \text{количество вершин} \qquad (9)$$

Теперь пойдем дальше. Для графа G=(V,U) порядка P объединим в одно целое все множества всех возможных маршрутов C^n_{ij} длины *n* по всем парам вершин i=1,...,P и j=1,...,P, т.е. вычислим $\cup^{p}_{i=1} \cup^{p}_{j=1} C^n_{ij}$. В результате получится квадратная матрица размерностью P×P, аналогичная матрице смежности, но в

(*i*,*j*)-ячейке которой стоят все маршруты длины *n* из вершины *i* в вершину *j*. Такая матрица $\|C^n_{ij}\|$ будет называться ***матрицей маршрутов***. Обозначим её через $\mathbf{M_c^n}$, где **n** – длина маршрутов матрицы, будет называться ***степенью матрицы***.

Матрицу 1-ой степени можно получить непосредственно из графа потому, что маршруты единичной длины – это дуги графа. Например, для графа, представленного на рис. 10, матрица маршрутов длины = 1 – это $\mathbf{M_c^1}$ приведена справа.



	v1	v2	v3	v4
v1		v1u1v2		v1u4v4 v1u8v4
v2	v2u5v1			v2u2v4
v3	v3u9v1	v3u6v2		
v4			v4u7v3	v4u3v4

$= \mathbf{M_c^1}$

Рисунок 10 – Пример графа для поиска маршрутов

Матрица маршрутов определенной степени задает перечисление всех маршрутов между всеми парами вершин соответствующей длины в графе G. Последовательность матриц $\mathbf{M_c^1}$, $\mathbf{M_c^2}$, ..., $\mathbf{M_c^n}$, ..., $\mathbf{M_c^P}$, n=1,...P, где P – порядок графа, задает все маршруты от длины 1 до длины P, причем в них возможны только простые и элементарные маршруты. При n>P в матрицах получаются маршруты большей длины, но все маршруты будут содержать циклическое повторение фрагментов матриц меньших степеней, т.е. маршруты будут составными и неэлементарными.

Для того, чтобы из матрицы меньшей степени можно получить матрицу большей степени, по аналогии с алгеброй, введем операцию перемножения матриц маршрутов • следующим образом:

$$\mathbf{M_c^{n+m}} = \mathbf{M_c^n} \bullet \mathbf{M_c^m} \text{ , где}$$

каждый (*i*,*j*)-элемент результирующей матрицы определяется на основе операции композиции множеств маршрутов соответствующей длины:

$$\mathbf{M_c^{n+m}} = \|C^{n+m}_{ij}\| = \| \cup^{p}_{k=1} C^{n}_{ik} \circ C^{m}_{kj} \| \qquad (10)$$

Интерпретация этой операции для (3,4)-ячейки матрицы $\mathbf{M_c^2}=\mathbf{M_c^1}\bullet\mathbf{M_c^1}$ графа на рис.10 приведена на рис.11. Как видно, матрицы перемножаются как обычно в алгебре, *i*-ая строка на *j*-ый столбец, но при этом выполняется операция композиции маршрутов.

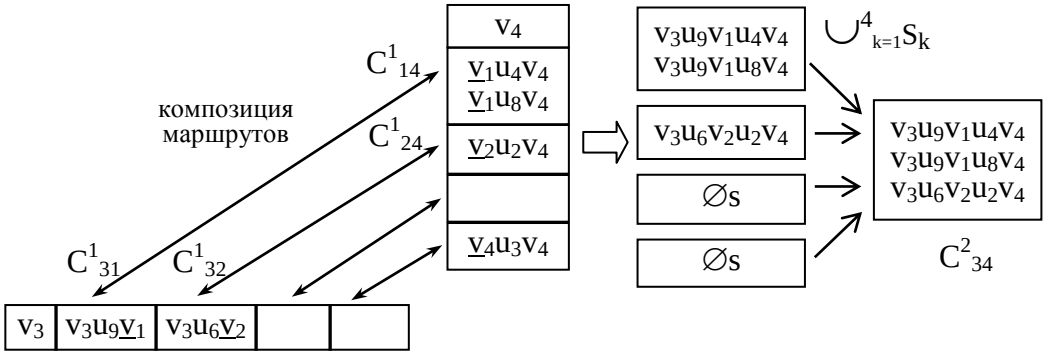


Рисунок 11 – Интерпретация перемножения матриц маршрутов

Используя операцию перемножения матриц маршрутов можно решить задачу перечисления всех маршрутов в графе заданной длины. Как было отмечено выше, такое перечисление имеет смысл от 1 до P – количество вершин графа. Таким образом, метод перечисления всех маршрутов состоит в последовательном построении матриц маршрутов, начиная с первой степени до заданной степени, т.е.

$$\mathbf{M_c^2}=\mathbf{M_c^1}\bullet\mathbf{M_c^1},\ \mathbf{M_c^3}=\mathbf{M_c^2}\bullet\mathbf{M_c^1},\ \mathbf{M_c^4}=\mathbf{M_c^3}\bullet\mathbf{M_c^1},...,\ \mathbf{M_c^n}=\mathbf{M_c^{n-1}}\bullet\mathbf{M_c^1},\ \text{и т.д.}$$

где исходная матрица $\mathbf{M_c^1}$ строится непосредственно по графу. В принципе, благодаря общности введенной операции перемножения матриц, матрицу более высокой степени можно получить из матриц меньших степеней, минуя вычисление промежуточных матриц. Например, матрицу степени 4 можно получить из матрицы степени 2, т.е. $\mathbf{M_c^4}=\mathbf{M_c^2}\bullet\mathbf{M_c^2}$, не вычисляя матрицу степени 3, а матрицу степени 8 можно получить из матрицы степени 4, т.е. $\mathbf{M_c^8}=\mathbf{M_c^4}\bullet\mathbf{M_c^4}$, не вычисляя матриц степеней 3, 5, 6, 7.

Рассмотрим пример графа на рис. 10, для нахождения всех маршрутов длины от 1 до P=4. Матрица маршрутов степени 1 уже построена и приведена на том же рисунке, $\mathbf{M_c^1}$. Построим матрицу степени 2, $\mathbf{M_c^2}=\mathbf{M_c^1}\bullet\mathbf{M_c^1}$, результат следующий:

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	$v_4u_3v_4$

$\bullet$

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	$v_4u_3v_4$

$=$

$=$

	v_1	v_2	v_3	v_4
v_1	$v_1u_1v_2u_5v_1$		$v_1u_4v_4u_7v_3$ $v_1u_8v_4u_7v_3$	$v_1u_1v_2u_2v_4$ $v_1u_4v_4u_3v_4$ $v_1u_8v_4u_3v_4$
v_2		$v_2u_5v_1u_1v_2$	$v_2u_2v_4u_7v_3$	$v_2u_5v_1u_4v_4$ $v_2u_5v_1u_8v_4$ $v_2u_2v_4u_3v_4$
v_3	$v_3u_6v_2u_5v_1$	$v_3u_9v_1u_1v_2$		$v_3u_9v_1u_4v_4$ $v_3u_9v_1u_8v_4$ $v_3u_6v_2u_2v_4$
v_4	$v_4u_7v_3u_9v_1$	$v_4u_7v_3u_6v_2$	$v_4u_3v_4u_7v_3$	$v_4u_3v_4u_3v_4$

$= \mathbf{M_c^2}$

(Серым выделен пример операции композиции для (3,4)-ячейки из рисунка 11)

Для получения матрицы степени 3, перемножим полученную матрицу степени 2 на матрицу степени 1, т.е. $\mathbf{Mc^3}=\mathbf{Mc^2} \bullet \mathbf{Mc^1}$, получаем следующий результат:

	v_1	v_2	v_3	v_4
v_1	$v_1u_1v_2u_5v_1$		$v_1u_4v_4u_7v_3$ $v_1u_8v_4u_7v_3$	$v_1u_1v_2u_2v_4$ $v_1u_4v_4u_3v_4$ $v_1u_8v_4u_3v_4$
v_2		$v_2u_5v_1u_1v_2$	$v_2u_2v_4u_7v_3$	$v_2u_5v_1u_4v_4$ $v_2u_5v_1u_8v_4$ $v_2u_2v_4u_3v_4$
v_3	$v_3u_6v_2u_5v_1$	$v_3u_9v_1u_1v_2$		$v_3u_9v_1u_4v_4$ $v_3u_9v_1u_8v_4$ $v_3u_6v_2u_2v_4$
v_4	$v_4u_7v_3u_9v_1$	$v_4u_7v_3u_6v_2$	$v_4u_3v_4u_7v_3$	$v_4u_3v_4u_3v_4$

$\bullet$

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	$v_4u_3v_4$

$=$

	v_1	v_2	v_3	v_4
v_1	$v_1u_4v_4u_7v_3u_9v_1$ $v_1u_8v_4u_7v_3u_9v_1$	$v_1u_1v_2u_5v_1u_1v_2$ $v_1u_4v_4u_7v_3u_6v_2$ $v_1u_8v_4u_7v_3u_6v_2$	$v_1u_1v_2u_2v_4u_7v_3$ $v_1u_4v_4u_3v_4u_7v_3$ $v_1u_8v_4u_3v_4u_7v_3$	$v_1u_1v_2u_5v_1u_4v_4$ $v_1u_1v_2u_5v_1u_8v_4$ $v_1u_1v_2u_2v_4u_3v_4$ $v_1u_4v_4u_3v_4u_3v_4$ $v_1u_8v_4u_3v_4u_3v_4$
v_2	$v_2u_5v_1u_1v_2u_5v_1$ $v_2u_2v_4u_7v_3u_9v_1$	$v_2u_2v_4u_7v_3u_6v_2$	$v_2u_5v_1u_4v_4u_7v_3$ $v_2u_5v_1u_8v_4u_7v_3$ $v_2u_2v_4u_3v_4u_7v_3$	$v_2u_5v_1u_1v_2u_2v_4$ $v_2u_5v_1u_4v_4u_3v_4$ $v_2u_5v_1u_8v_4u_3v_4$ $v_2u_2v_4u_3v_4u_3v_4$
v_3	$v_3u_9v_1u_1v_2u_5v_1$	$v_3u_6v_2u_5v_1u_1v_2$	$v_3u_6v_2u_2v_4u_7v_3$ $v_3u_9v_1u_4v_4u_7v_3$ $v_3u_9v_1u_8v_4u_7v_3$	$v_3u_6v_2u_5v_1u_4v_4$ $v_3u_6v_2u_5v_1u_8v_4$ $v_3u_9v_1u_1v_2u_2v_4$ $v_3u_9v_1u_4v_4u_3v_4$ $v_3u_9v_1u_8v_4u_3v_4$ $v_3u_6v_2u_2v_4u_3v_4$
v_4	$v_4u_7v_3u_6v_2u_5v_1$ $v_4u_3v_4u_7v_3u_9v_1$	$v_4u_7v_3u_9v_1u_1v_2$ $v_4u_3v_4u_7v_3u_6v_2$	$v_4u_3v_4u_3v_4u_7v_3$	$v_4u_7v_3u_9v_1u_4v_4$ $v_4u_7v_3u_9v_1u_8v_4$ $v_4u_7v_3u_6v_2u_2v_4$ $v_4u_3v_4u_3v_4u_3v_4$

$= \mathbf{M_c^3}$

Для получения матрицы маршрутов степени 4, можно перемножить полученную матрицу степени 3 на матрицу степени 1, т.е. $\mathbf{Mc^4=Mc^3 \bullet Mc^1}$, либо перемножить матрицу степени 2 на матрицу степени 2, т.е. $\mathbf{Mc^4=Mc^2 \bullet Mc^2}$, получаем результат, приведенный ниже.

Как видно из приведенных матриц, на главной диагонали всегда образуются циклические маршруты. В ячейках матриц появляются как элементарные, так и неэлементарные маршруты, причем матрица степени 4, равная порядку графа, содержит неэлементарные маршруты. Поскольку в большинстве задач поиска маршрутов, интересуются в основном элементарными маршрутами, то перемножение матриц имеет смысл проводить до порядок графа минус один, так как в графе для соединения P вершин в маршрут требуется как минимум P–1 дуг. С другой стороны, в матрице степени 4, имеются как составные, так и простые маршруты. Наличие простых маршрутов обусловлено избыточностью графа по связям. Возможно существование простых маршрутов в матрицах и больших степеней: P, P+1, P+2, ..., Q, где Q – количество дуг графа.

Вообще, прямое перечисление всех маршрутов графа используется редко из-за громоздкости вычислений, потребности в больших объёмах памяти под результирующие матрицы, поэтому на практике данный метод

используется как основа для алгоритмов поиска маршрутов, обладающих определенными заданными характеристиками, например элементарных и простых, для которых вводятся специальные определения.

Матрица маршрутов степени 4 (**Mc^4**).

	v_1	v_2	v_3	v_4
v_1	$v_1u_1v_2u_5v_1u_1v_2u_5v_1$ $v_1u_4v_4u_7v_3u_6v_2u_5v_1$ $v_1u_8v_4u_7v_3u_6v_2u_5v_1$ $v_1u_1v_2u_2v_4u_7v_3u_9v_1$ $v_1u_4v_4u_3v_4u_7v_3u_9v_1$ $v_1u_8v_4u_3v_4u_7v_3u_9v_1$	$v_1u_4v_4u_7v_3u_9v_1u_1v_2$ $v_1u_8v_4u_7v_3u_9v_1u_1v_2$ $v_1u_1v_2u_2v_4u_7v_3u_6v_2$ $v_1u_4v_4u_3v_4u_7v_3u_6v_2$ $v_1u_8v_4u_3v_4u_7v_3u_6v_2$	$v_1u_1v_2u_5v_1u_4v_4u_7v_3$ $v_1u_1v_2u_5v_1u_8v_4u_7v_3$ $v_1u_1v_2u_2v_4u_3v_4u_7v_3$ $v_1u_4v_4u_3v_4u_3v_4u_7v_3$ $v_1u_8v_4u_3v_4u_3v_4u_7v_3$	$v_1u_4v_4u_7v_3u_9v_1u_4v_4$ $v_1u_8v_4u_7v_3u_9v_1u_4v_4$ $v_1u_4v_4u_7v_3u_9v_1u_8v_4$ $v_1u_8v_4u_7v_3u_9v_1u_8v_4$ $v_1u_1v_2u_5v_1u_1v_2u_2v_4$ $v_1u_4v_4u_7v_3u_6v_2u_2v_4$ $v_1u_8v_4u_7v_3u_6v_2u_2v_4$ $v_1u_1v_2u_5v_1u_4v_4u_3v_4$ $v_1u_1v_2u_5v_1u_8v_4u_3v_4$ $v_1u_1v_2u_2v_4u_3v_4u_3v_4$ $v_1u_4v_4u_3v_4u_3v_4u_3v_4$ $v_1u_8v_4u_3v_4u_3v_4u_3v_4$
v_2	$v_2u_2v_4u_7v_3u_6v_2u_5v_1$ $v_2u_5v_1u_4v_4u_7v_3u_9v_1$ $v_2u_5v_1u_8v_4u_7v_3u_9v_1$ $v_2u_2v_4u_3v_4u_7v_3u_9v_1$	$v_2u_5v_1u_1v_2u_5v_1u_1v_2$ $v_2u_2v_4u_7v_3u_9v_1u_1v_2$ $v_2u_5v_1u_4v_4u_7v_3u_6v_2$ $v_2u_5v_1u_8v_4u_7v_3u_6v_2$ $v_2u_2v_4u_3v_4u_7v_3u_6v_2$	$v_2u_5v_1u_1v_2u_2v_4u_7v_3$ $v_2u_5v_1u_4v_4u_3v_4u_7v_3$ $v_2u_5v_1u_8v_4u_3v_4u_7v_3$ $v_2u_2v_4u_3v_4u_3v_4u_7v_3$	$v_2u_5v_1u_1v_2u_5v_1u_4v_4$ $v_2u_2v_4u_7v_3u_9v_1u_4v_4$ $v_2u_5v_1u_1v_2u_5v_1u_8v_4$ $v_2u_2v_4u_7v_3u_9v_1u_8v_4$ $v_2u_2v_4u_7v_3u_6v_2u_2v_4$ $v_2u_5v_1u_1v_2u_2v_4u_3v_4$ $v_2u_5v_1u_4v_4u_3v_4u_3v_4$ $v_2u_5v_1u_8v_4u_3v_4u_3v_4$ $v_2u_2v_4u_3v_4u_3v_4u_3v_4$
v_3	$v_3u_6v_2u_5v_1u_1v_2u_5v_1$ $v_3u_6v_2u_2v_4u_7v_3u_9v_1$ $v_3u_9v_1u_4v_4u_7v_3u_9v_1$ $v_3u_9v_1u_8v_4u_7v_3u_9v_1$	$v_3u_9v_1u_1v_2u_5v_1u_1v_2$ $v_3u_6v_2u_2v_4u_7v_3u_6v_2$ $v_3u_9v_1u_4v_4u_7v_3u_6v_2$ $v_3u_9v_1u_8v_4u_7v_3u_6v_2$	$v_3u_6v_2u_5v_1u_4v_4u_7v_3$ $v_3u_6v_2u_5v_1u_8v_4u_7v_3$ $v_3u_9v_1u_1v_2u_2v_4u_7v_3$ $v_3u_9v_1u_4v_4u_3v_4u_7v_3$ $v_3u_9v_1u_8v_4u_3v_4u_7v_3$ $v_3u_6v_2u_2v_4u_3v_4u_7v_3$	$v_3u_9v_1u_1v_2u_5v_1u_4v_4$ $v_3u_9v_1u_1v_2u_5v_1u_8v_4$ $v_3u_6v_2u_5v_1u_1v_2u_2v_4$ $v_3u_6v_2u_5v_1u_4v_4u_3v_4$ $v_3u_6v_2u_5v_1u_8v_4u_3v_4$ $v_3u_9v_1u_1v_2u_2v_4u_3v_4$ $v_3u_9v_1u_4v_4u_3v_4u_3v_4$ $v_3u_9v_1u_8v_4u_3v_4u_3v_4$ $v_3u_6v_2u_2v_4u_3v_4u_3v_4$
v_4	$v_4u_7v_3u_9v_1u_1v_2u_5v_1$ $v_4u_3v_4u_7v_3u_6v_2u_5v_1$ $v_4u_3v_4u_3v_4u_7v_3u_9v_1$	$v_4u_7v_3u_6v_2u_5v_1u_1v_2$ $v_4u_3v_4u_7v_3u_9v_1u_1v_2$ $v_4u_3v_4u_3v_4u_7v_3u_6v_2$	$v_4u_7v_3u_9v_1u_4v_4u_7v_3$ $v_4u_7v_3u_9v_1u_8v_4u_7v_3$ $v_4u_7v_3u_6v_2u_2v_4u_7v_3$ $v_4u_3v_4u_3v_4u_3v_4u_7v_3$	$v_4u_7v_3u_6v_2u_5v_1u_4v_4$ $v_4u_3v_4u_7v_3u_9v_1u_4v_4$ $v_4u_7v_3u_6v_2u_5v_1u_8v_4$ $v_4u_3v_4u_7v_3u_9v_1u_8v_4$ $v_4u_7v_3u_9v_1u_1v_2u_2v_4$ $v_4u_3v_4u_7v_3u_6v_2u_2v_4$ $v_4u_7v_3u_9v_1u_4v_4u_3v_4$ $v_4u_7v_3u_9v_1u_8v_4u_3v_4$ $v_4u_7v_3u_6v_2u_2v_4u_3v_4$ $v_4u_3v_4u_3v_4u_3v_4u_3v_4$

Метод нахождения всех маршрутов заданной длины между всеми парами вершин графа является очень мощным средством решения графовых задач. Однако в ряде задач анализа систем, например оценка степени

значимости отдельного элемента в системе или определение веса элемента в системе, нет необходимости искать все маршруты, а нужно знать только количество маршрутов заданной длины из одной вершины в другую. Задача определения количества маршрутов между всеми парами вершин также решается с помощью описанного матричного метода, но этот метод несколько модифицируется.

Прежде всего, нужно учесть свойство 4 операции композиции множеств маршрутов: $|S^m_{ik} \circ S^n_{kj}| = |S^n_{ik}| * |S^m_{kj}|$, т.е. количество маршрутов в множестве, полученном композицией двух множеств маршрутов, равно произведению количества маршрутов в первом множестве на количество маршрутов во втором множестве. Выше, множество всех возможных маршрутов длины $n+m$ из вершины i в вершину j было обозначено как C^{n+m}_{ij} и было указано, что композиция двух таких множеств: C^n_{ik} – длины n, заканчивающихся в вершине k, и C^n_{kj} – длины m, начинающихся в вершине k, не дает множества всех возможных маршрутов длины $n+m$, т.е. $S^{n+m}_{ij} = C^n_{ik} \circ C^m_{kj} \neq C^{n+m}_{ij}$. Чтобы получить множество всех возможных маршрутов C^{n+m}_{ij} нужно было взять объединение композиций множеств всех возможных маршрутов длины n, заканчивающихся в вершине k, и длины m, начинающихся в вершине k, по всем вершинам графа, т.е. $C^{n+m}_{ij} = \cup_{k \in V} (C^n_{ik} \circ C^m_{kj})$. Следовательно, для решения задачи о количестве всех возможных маршрутов длины $n+m$ между двумя вершинами, нужно просуммировать количества маршрутов, полученных в результате композиции множеств длины n и длины m общими вершинами k по всем вершинам графа $k \in V$, т.е. $|C^{n+m}_{ij}| = \Sigma_{k \in V} |C^n_{ik} \circ C^m_{kj}|$. Принимая во внимание свойство 4 операции композиции получаем:

$$|C^{n+m}_{ij}| = \Sigma_{k \in V} |C^n_{ik}| * |C^m_{kj}| \qquad (11)$$

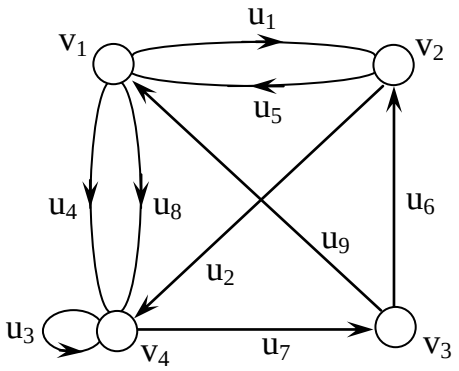
т.е. для вычисления количества всех возможных маршрутов длины $n+m$ из вершины i в вершину j нужно вычислить сумму всех произведений количества маршрутов длины n из вершины i на количество маршрутов длины m в вершину j с общими вершинами k.

По аналогии с изложенным выше, зная количество маршрутов длины n между всеми парами вершин, можно построить матрицу количества

маршрутов длины *n*, т.е. $||C^n_{ij}||$ i=1,...,P, j=1,...,P, которую обозначим через $\mathbf{M_k^n}$. Зная матрицы количеств маршрутов степени *n* и степени *m*, путем перемножения этих матриц с учетом суммирования произведений количеств маршрутов этих матриц, можно получить матрицу количества маршрутов степени *n*+*m*, т.е. $\mathbf{M_k^{n+m}} = \mathbf{M_k^n} \bullet \mathbf{M_k^m}$ или подробно:

$$\mathbf{M_k^{n+m}} = ||(|C^{n+m}_{ij}|)|| = || (\Sigma_{k=1}^{P} |C^n_{ik}| * |C^m_{kj}|) || \qquad (12)$$

Матрицу количества маршрутов длины 1, т.е. $\mathbf{M_k^1}$, можно построить непосредственно из графа, каждая её (*i*,*j*) ячейка – это количество дуг из вершины *i* в вершину *j*. Например, для графа на рис.12 матрица количества маршрутов длины 1 – $\mathbf{M_k^1}$ имеет вид:



	v1	v2	v3	v4
v1	0	1	0	2
v2	1	0	0	1
v3	1	1	0	0
v4	0	0	1	1

$= \mathbf{M_k^1}$

Рисунок 12 – Пример графа для построения матрицы количества маршрутов

Чтобы вычислить количество маршрутов длины 2 в этом графе, нужно перемножить эту матрицу первой степени на себя, т.е. $\mathbf{M_k^2} = \mathbf{M_k^1} \bullet \mathbf{M_k^1}$, в результате получаем:

	v1	v2	v3	v4
v1	0	1	0	2
v2	1	0	0	1
v3	1	1	0	0
v4	0	0	1	1

$\bullet$

	v1	v2	v3	v4
v1	0	1	0	2
v2	1	0	0	1
v3	1	1	0	0
v4	0	0	1	1

$=$

	v1	v2	v3	v4
v1	1	0	2	3
v2	0	1	1	3
v3	1	1	0	3
v4	1	1	1	1

$= \mathbf{M_k^2}$

Для демонстрации, ячейка (v_1,v_4) матрицы получается так:

$|C^2_{14}| = (\Sigma_{k=1}^{4} |C^1_{1k}| * |C^1_{k4}|) = (0*2+1*1+0*0+2*1) = 1+2 = 3$

Чтобы вычислить количество маршрутов длины 3 в этом графе, нужно перемножить полученную матрицу второй степени на матрицу первой степени, т.е. $\mathbf{M_k^3} = \mathbf{M_k^2} \bullet \mathbf{M_k^1}$, в результате получаем:

	v_1	v_2	v_3	v_4
v_1	1	0	2	3
v_2	0	1	1	3
v_3	1	1	0	3
v_4	1	1	1	1

$\bullet$

	v_1	v_2	v_3	v_4
v_1	0	1	0	2
v_2	1	0	0	1
v_3	1	1	0	0
v_4	0	0	1	1

$=$

	v_1	v_2	V_3	v_4
v_1	2	3	3	5
v_2	3	1	3	4
v_3	1	1	3	6
v_4	2	2	1	4

$= \mathbf{M_k^3}$

Для демонстрации, ячейка (v_1,v_4) матрицы получается так:

$|C^3_{14}| = (\Sigma_{k=1}^{4} |C^2_{1k}|*|C^1_{k4}|) = (1*2+0*1+2*0+3*1) = 2+3 = 5$

Чтобы вычислить количество маршрутов длины 4 в этом графе, нужно перемножить полученную матрицу третьей степени на матрицу первой степени, т.е. $\mathbf{M_k^4} = \mathbf{M_k^3} \bullet \mathbf{M_k^1}$, в результате получаем:

	v_1	v_2	v_3	v_4
v_1	2	3	3	5
v_2	3	1	3	4
v_3	1	1	3	6
v_4	2	2	1	4

$\bullet$

	v_1	v_2	v_3	v_4
v_1	0	1	0	2
v_2	1	0	0	1
v_3	1	1	0	0
v_4	0	0	1	1

$=$

	v_1	v_2	V_3	v_4
v_1	6	5	5	12
v_2	4	5	4	9
v_3	4	4	6	9
v_4	3	3	4	10

$= \mathbf{M_k^4}$

Для демонстрации, ячейка (v_1,v_4) матрицы получается так:

$|C^4_{14}| = (\Sigma_{k=1}^{4} |C^3_{1k}|*|C^1_{k4}|) = (2*2+3*1+3*0+5*1) = 4+3+5 = 12$

Результаты вычислений матриц количества маршрутов $\mathbf{M_k^n}$, n=1,..,4, не трудно проверить по матрицам маршрутов соответствующей степени $\mathbf{M_c^n}$, n=1,..,4, которые уже были приведены выше, подсчитав количество маршрутов в ячейках этих матриц.

6 Пути и циклы, метод перечисления путей и циклов

Путь – это простой и элементарный маршрут. Пути бывают ориентированные, в которых ориентация всех дуг совпадает, и неориентированными, в противном случае. Например, для графа на рис. 9, маршрут {v_1, u_{10}, v_9, u_{11}, v_4} является ориентированным путем, маршрут {v_1, u_7, v_7, u_8, v_8, u_9, v_4} является неориентированным путем, а маршрут {v_1, u_{10}, v_9, u_{13}, v_7, u_{16}, v_8, u_{12}, v_9, u_{11}, v_4} не является путем. Длина пути – это количество дуг в пути.

Цикл – это простой и элементарный маршрут, у которого начальная и конечная вершины совпадают. Циклы также бывают ориентированными и неориентированными. Например, для графа на рис. 9, маршрут $\{v_9, u_{13}, v_7, u_{14}, v_2, u_2, v_3, u_{17}, v_8, u_{12}, v_9\}$ является циклом. Длина цикла – это количество дуг в цикле.

Петлю можно интерпретировать как цикл единичной длины, а дугу – как путь единичной длины. Цикл, образованный парой противоположно направленных дуг, также называется единичным. Цикл, образованный тремя дугами, называется угольником. Например, на рис. 9 петли u_{19} и u_{20} можно рассматривать как циклы, циклы $\{v_7, u_{14}, v_2, u_{18}, v_7\}$ и $\{v_3, u_{17}, v_8, u_{15}, v_3\}$ являются единичными, а цикл $\{v_9, u_{13}, v_7, u_{16}, v_8, u_{12}, v_9\}$ является угольником.

Циклы, которые образованы одними и теми же дугами и вершинами, расположенные в маршруте в одном и том же порядке, и отличающиеся один от другого лишь циклическим сдвигом элементов маршрута, называются ***эквивалентными***. Например, для графа на рис. 9 следующие циклы $\{v_9, u_{13}, v_7, u_{16}, v_8, u_{12}, v_9\}$, $\{v_7, u_{16}, v_8, u_{12}, v_9, u_{13}, v_7\}$, $\{v_8, u_{12}, v_9, u_{13}, v_7, u_{16}, v_8\}$ являются эквивалентными.

В некоторых задачах анализа систем, например при оценке степени влияния элементов системы на проходящие потоки или анализе прохождения потоков вещества, энергии, информации через систему, возникает необходимость нахождения всех путей и циклов в графе системы. Метод отыскания всех путей и циклов полностью аналогичен методу отыскания всех маршрутов, который просто немножко модифицируется следующим образом.

Во-первых переопределяется операция композиции маршрутов *, введенная выше, так, чтобы с её помощью получались элементарные и простые маршруты. Для этого накладывается дополнительное ограничение, чтобы два пути, которые соединяются композицией не имели одинаковых дуг и одинаковых вершин, кроме последней и первой вершины, иначе результатом операции является пустой маршрут. Итак, пусть даны пути: $p_i=\{v_{i1},u_{i1}, ...,v_{in},u_{in}, v_k\}$ длины n и $p_j=\{v_l, u_{j1},v_{j1}, ..., u_{jm},v_{jm}\}$ длины m. ***Композицией путей*** является путь, полученный следующей операцией:

$$p_i * p_j = \begin{cases} \{v_{i1},u_{i1}, ...,v_{in},u_{in}, v_k, u_{j1},v_{j1}, ..., u_{jm},v_{jm}\}, \text{ если } v_k=v_l, \text{ т.е. конец первого пути} \\ \text{совпадает с началом второго, и если } v_{ix}\neq v_{jy} \text{ и } u_{ix}\neq u_{jy}, \ x=1,..,n, \ y=1,..,m, \\ \text{кроме случая } v_{i1}=v_{jm}, \text{ т.е. результат является тоже путем (или циклом) ;} \\ \varnothing s \text{ – в противном случае (пустой маршрут).} \end{cases} \quad (13)$$

Что касается, выкладок с композицией множеств, и объединением множеств в матрицу, перемножение матриц – все это остается без изменений, только относится к путям. Обозначим матрицу путей степени **n** через $\mathbf{M_p^n}$.

Во-вторых, немножко изменяются матрицы перед перемножением. Дело в том, что перемножая матрицу, содержащую циклы, т.е. заполненные ячейки на главной диагонали, в результате получается матрица, содержащая составные маршруты, ведь если к любой вершине цикла добавить путь, то получится неэлементарный маршрут, как показано на рис. 13, что противоречит требованию получать простые и элементарные маршруты.

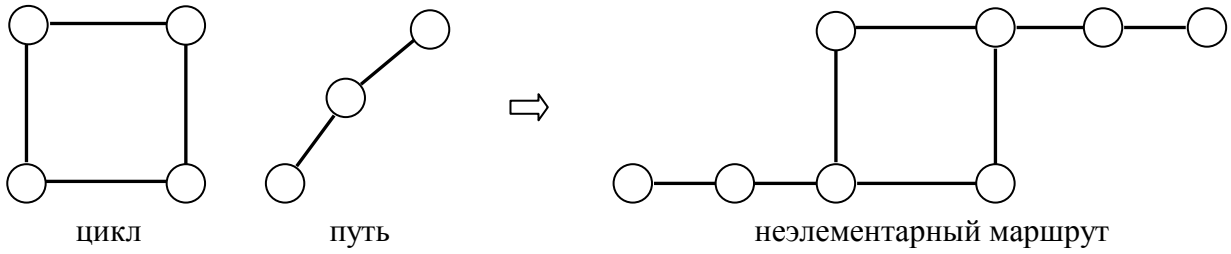


Рисунок 13 – Пояснение к операции композиции путей

Для этого перед перемножением матриц из них удаляют все циклы, т.е. вычеркивают главную диагональ. Обозначим матрицу путей с вычеркнутой главной диагональю через $\boldsymbol{M'}_{\mathbf{p}}^{\mathbf{n}}$. Матрица первой степени по прежнему содержит все дуги графа и строится непосредственно по графу, ведь дуги одновременно являются путями единичной длины.

В-третьих, для графа G порядка P, матрицы перемножают до степени, не больше P–1, так как для соединения P вершин в простой и элементарный маршрут (путь) требуется как минимум P–1 дуг. Большее количество дуг уже даст либо составной (дуги повторяются), либо неэлементарный (вершины повторяются) маршрут.

Таким образом, **метод отыскания всех путей** состоит следующем:

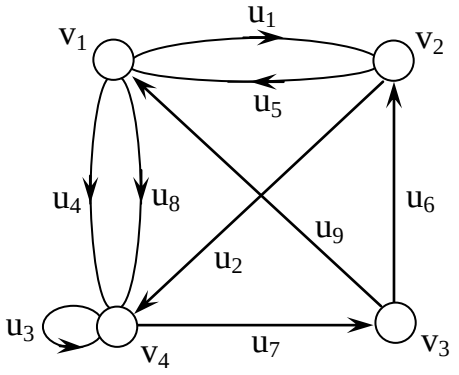
1. Строится по графу матрица путей длины 1, т.е. $\mathbf{M_p^1}$.
2. Вычеркнуть из главной диагонали все петли, т.е. получить $\boldsymbol{M'}_{\mathbf{p}}^{\mathbf{1}}$.
3. Положить n=2.

4. Для получения матрицы большей степени **n** перемножить преобразованную матрицу предыдущей степени **n–1** без циклов на матрицу степени 1 без петель, т.е.

$$\mathbf{M_p^n} = \boldsymbol{M'^{n-1}_p} \bullet \boldsymbol{M'^{1}_p}$$

5. Из матрицы степени **n** вычеркнуть главную диагональ, т.е. преобразовать матрицу $\mathbf{M_p^n}$ в матрицу $\boldsymbol{M'^{n}_p}$.
6. Проверить, если **n** = P–1, то прекратить вычисления, иначе перейти п.4

Рассмотрим применение метода для графа на рис. 14, для которого уже построена матрица маршрутов степени 1. Очевидно, что маршруты длины 1 не могут быть неэлементарными и тем более сложными, поскольку состоят из единственной дуги. Таким образом, матрица маршрутов длины 1 одновременно является и матрицей путей длины 1, т.е. $\mathbf{M_p^1}=\mathbf{M_c^1}$. Вычеркнем из неё главную диагональ, т.е. удалим петлю ($v_4u_3v_4$), и получаем матрицу $\boldsymbol{M'^{1}_p}$, которая и будет использоваться при перемножениях.



	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	$v_4u_3v_4$

$=\mathbf{M_p^1}$

Рисунок 14 – Пример графа для построения матрицы путей

Для получения матрицы степени 2 перемножим матрицу степени 1 без циклов на себя, т.е. $\mathbf{M_p^2} = \boldsymbol{M'^{1}_p} \bullet \boldsymbol{M'^{1}_p}$, учитывая что к каждой ячейке матрицы теперь применяется переопределенная операция композиции, т.е. в результате композиции должен получаться простой и элементарный маршрут. Получаем:

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	

$\bullet$

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	

=

$=$

	v_1	v_2	v_3	v_4
v_1	$v_1u_1v_2u_5v_1$		$v_1u_4v_4u_7v_3$ $v_1u_8v_4u_7v_3$	$v_1u_1v_2u_2v_4$
v_2		$v_2u_5v_1u_1v_2$	$v_2u_2v_4u_7v_3$	$v_2u_5v_1u_4v_4$ $v_2u_5v_1u_8v_4$
v_3	$v_3u_6v_2u_5v_1$	$v_3u_9v_1u_1v_2$		$v_3u_9v_1u_4v_4$ $v_3u_9v_1u_8v_4$ $v_3u_6v_2u_2v_4$
v_4	$v_4u_7v_3u_9v_1$	$v_4u_7v_3u_6v_2$		

$= \mathbf{M_p^2}$

Как видно, полученная матрица, содержит два цикла на главной диагонали, которые впрочем являются эквивалентными. Для получения матрицы степени 3 сначала вычеркнем главную диагональ из матрицы степени 2, т.е. $\mathbf{M_p^2} \rightarrow \boldsymbol{M'^2_p}$, а затем перемножим полученную матрицу с матрицей степени 1 без петель, т.е. $\mathbf{M_p^3} = \boldsymbol{M'^2_p} \bullet \boldsymbol{M'^1_p}$, получаем следующее:

	v_1	v_2	v_3	v_4
v_1			$v_1u_4v_4u_7v_3$ $v_1u_8v_4u_7v_3$	$v_1u_1v_2u_2v_4$
v_2			$v_2u_2v_4u_7v_3$	$v_2u_5v_1u_4v_4$ $v_2u_5v_1u_8v_4$
v_3	$v_3u_6v_2u_5v_1$	$v_3u_9v_1u_1v_2$		$v_3u_9v_1u_4v_4$ $v_3u_9v_1u_8v_4$ $v_3u_6v_2u_2v_4$
v_4	$v_4u_7v_3u_9v_1$	$v_4u_7v_3u_6v_2$		

$\bullet$

	v_1	v_2	v_3	v_4
v_1		$v_1u_1v_2$		$v_1u_4v_4$ $v_1u_8v_4$
v_2	$v_2u_5v_1$			$v_2u_2v_4$
v_3	$v_3u_9v_1$	$v_3u_6v_2$		
v_4			$v_4u_7v_3$	

$=$

$=$

	v_1	v_2	v_3	v_4
v_1	$v_1u_4v_4u_7v_3u_9v_1$ $v_1u_8v_4u_7v_3u_9v_1$	$v_1u_4v_4u_7v_3u_6v_2$ $v_1u_8v_4u_7v_3u_6v_2$	$v_1u_1v_2u_2v_4u_7v_3$	
v_2	$v_2u_2v_4u_7v_3u_9v_1$	$v_2u_2v_4u_7v_3u_6v_2$	$v_2u_5v_1u_4v_4u_7v_3$ $v_2u_5v_1u_8v_4u_7v_3$	
v_3	$v_3u_9v_1u_1v_2u_5v_1$	$v_3u_6v_2u_5v_1u_1v_2$	$v_3u_6v_2u_2v_4u_7v_3$ $v_3u_9v_1u_4v_4u_7v_3$ $v_3u_9v_1u_8v_4u_7v_3$	$v_3u_6v_2u_5v_1u_4v_4$ $v_3u_6v_2u_5v_1u_8v_4$ $v_3u_9v_1u_1v_2u_2v_4$
v_4	$v_4u_7v_3u_6v_2u_5v_1$	$v_4u_7v_3u_9v_1u_1v_2$		$v_4u_7v_3u_9v_1u_4v_4$ $v_4u_7v_3u_9v_1u_8v_4$ $v_4u_7v_3u_6v_2u_2v_4$

$= \mathbf{M_p^3}$

Поскольку граф содержит 4 вершины, т.е. P=4, то условие n=3 < P–1 перестало выполняться, и дальнейшее перемножение матриц больше не имеет смысла, расчет останавливается. Последовательность матриц $\mathbf{M_p}^1$, $\mathbf{M_p}^2$, $\mathbf{M_p}^3$ задает перечисление всех путей и циклов заданного графа. Это не трудно проверить простым перечислением по рисунку 14.

Матрицы путей более высоких степеней можно получать перемножением матриц путей меньших степеней, пропуская некоторые матрицы, также как это делалось с матрицами маршрутов.

Изложенный метод перечисления путей и циклов графа совместно с методом перечисления маршрутов демонстрируют общую **методику отыскания маршрутов**, обладающих каким-то заданным свойством $\mathfrak{I}$. Общая методика состоит в следующем:

1. Переопределяется операция композиции * маршрутов таким образом, чтобы из маршрутов, обладающих заданным свойством $\mathfrak{I}$, можно было получить новый маршрут, обладающий тем же свойством $\mathfrak{I}$.

$$s_i * s_j = \begin{cases} \{v_{i1},u_{i1}, ...,v_{in},u_{in}, v_k, u_{j1},v_{j1}, ..., u_{jm},v_{jm},\}, \text{ если } v_k=v_l, \text{ т.е. конец первого маршрута совпадает с началом второго маршрута и маршрут обладает заданным свойством } \mathfrak{I}; \\ \varnothing s – \text{в противном случае (пустой маршрут).} \end{cases} \quad (14)$$

2. Определяется правило построения непосредственно по графу начальной матрицы $\mathbf{M_\mathfrak{I}}^1$ маршрутов, обладающих заданным свойством $\mathfrak{I}$.
3. Определяется правило преобразования некоторой матрицы маршрутов к требуемому виду $\boldsymbol{M'_\mathfrak{I}}^1$, в котором она может использоваться при перемножениях.
4. Задается условие прекращения поиска ξ^{stop} новых матриц $\mathbf{M_\mathfrak{I}}^n$ больших степеней.
5. Осуществляется последовательное перемножение матриц от степени 1 до заданной степени, пока выполняется условие прекращения поиска ξ.

Свойством $\mathfrak{I}$ может быть, например: путь, в котором вершины чередуются между долями графа; путь, в котором дуги имеют заданную раскраску; путь, проходящий через заданное подмножество вершин, путь, проходящий по разрешенным дугам, а также возможны другие постановки задачи.

Задача поиска путей и циклов очень распространена при исследованиях систем, имеет много различных постановок и часто выступает как подзадача в более сложных задачах. Например, при проверке правильности соединения элементов возникает задача нахождения пути, проходящего через заданное множество элементов или нахождения пути между указанными элементами. Если во время анализа предположить, что некоторые элементы вышли из строя, то возникает задача нахождения путей между заданными элементами, не проходящих через множество вышедших из строя элементов.

7 Эйлеровы и гамильтоновы маршруты

При анализе структур представляет особый интерес нахождение маршрутов, обладающих определенным максимальным свойством, например, обходящие все вершины графа или проходящие по каждой дуге графа.

Маршрут, проходящий через все дуги графа в точности один раз, называется ***эйлеровым циклом***. Например, для графа на рис. 9 цикл $\{v_4, u_4, v_5, u_{20}, v_5, u_5, v_6, u_{19}, v_6, u_6, v_1, u_{10}, v_9, u_{11}, v_4, u_9, v_8, u_{12}, v_9, u_{13}, v_7, u_7, v_1, u_1, v_2, u_{14}, v_7, u_{18}, v_2, u_2, v_3, u_{17}, v_8, u_8, v_7, u_{16}, v_8, u_{15}, v_3, u_3, v_4\}$ является эйлеровым циклом. Очевидно, что все эйлеровы циклы в графе отличаются друг от друга только циклическим порядком дуг и вершин в маршруте и являются, таким образом, эквивалентными. Поэтому говорят просто об эйлеровом цикле в графе.

Связанный граф, содержащий эйлеровый цикл, называется ***эйлеровым графом***. Такой граф можно нарисовать не отрывая карандаша от бумаги. Эйлер доказал, что связанный граф, содержит цикл, проходящий через все дуги графа в точности по одному разу, тогда и только тогда, когда степени всех его вершин чётны. Эта теорема Эйлера используется как критерий проверки эйлеровости графа.

Задача поиска эйлерового цикла часто встречается в головоломках и играх. В системах управления эта задача встречается при анализе эффективности того или иного порядка выполнения операций. Например, имеется последовательность операций по обработке изделий с перерывами между ними. Каждая операция имеет некоторые числовые характеристики и затраты

времени на выполнение. Также имеется функция эффективности, зависящая от времени. Тогда перерывы представляются вершинами графа, а операции – дугами графа. Поскольку за обработкой одного изделия следует другое, то все операции выполняются циклически и нужно найти такой порядок выполнения всех операций, чтобы обеспечить наилучшее значение заданной функции эффективности с учётом перерывов между операциями.

Маршрут, проходящий через все вершины графа в точности один раз называется ***гамильтоновым***. Пути и циклы, образованные гамильтоновым маршрутом, тоже называются гамильтоновыми. Например, для графа на рис. 9 путь $\{v_4, u_4, v_5, u_5, v_6, u_6, v_1, u_1, v_2, u_2, v_3, u_{15}, v_8, u_{12}, v_9, u_{13}, v_7\}$ и цикл $\{v_9, u_{13}, v_7, u_{14}, v_2, u_2, v_3, u_{15}, v_8, u_9, v_4, u_4, v_5, u_5, v_6, u_6, v_1, u_{10}, v_9\}$ являются гамильтоновыми. Очевидно, что гамильтоновых путей и циклов в графе может быть несколько.

Связанный граф, содержащий гамильтонов цикл, называется ***гамильтоновым графом***. Распознать, является ли данный граф гамильтоновым, как правило, довольно трудно. Вообще, изучение условий гамильтоновости графов – одно из популярных направлений в теории графов. Большинство теорем утверждает, что при выполнении определенных условий, граф является гамильтоновым, т.е. в нем можно найти цикл или путь через все вершины графа, но это только достаточные условия для определенного типа графов. Например, можно привести следующие теоремы:

1. Если в графе G порядка $P \geq 3$ для любой пары несмежных вершин v_i и v_j сумма степеней этих вершин не меньше порядка графа, т.е. $\rho_{vi} + \rho_{vj} \geq P$, то граф G является гамильтоновым. (О. Оре)
2. Если в графе G порядка $P \geq 3$ для любой вершины **v** выполняется неравенство $\rho_v \geq P/2$, то граф G является гамильтоновым. (Г. Дирак)
3. Всякий 4-связанный планарный граф G является гамильтоновым. (У. Татт)
4. Если в плоском графе G каждый 3-цикл является границей некоторой грани, то граф является гамильтоновым. (Х. Уитни)

Практическое применение задачи поиска гамильтоновых путей и циклов связано прежде всего с задачей коммивояжера. Она состоит в следующем: коммивояжер должен посетить каждый из заданных городов по одному разу,

выехав из некоторого начального города и закончив свое путешествие в другом городе (поиск гамильтонова пути) или в начальном городе (поиск гамильтонова цикла). Между городами имеются разные дороги, причем в какие-то города можно приехать только посетив другие города. Расстояния между городами различны и перемещение из одного города в другой требует разных затрат времени и денег. На графе города представляются вершинами, а дороги дугами. Требуется построить такой план объезда городов, чтобы затраты были минимальны, т.е. выбрать из всех гамильтоновых путей наилучший.

Задача поиска гамильтонова пути и цикла широко встречается в системах управления. Например, имеется матрица, на которой расположены детали, причем матрица заполнялась в произвольном порядке, и имеется устройство обработки этих деталей, исполнительный механизм которого способен перемещаться по матрице. Требуется минимизировать время перемещения исполнительного устройства по матрице для обработки всех деталей. Если детали представить вершинами графа, а допустимые пути перемещения устройства дугами графа, то нужно найти гамильтонов путь на этом графе, дающий минимальное время. Задача нахождения эйлерова цикла, возникает при проверке замкнутости исследуемой системы.

8 Достижимость и ранги вершин графа

При анализе потоков вещества, энергии, информации системы, говорят, что элемент *i* ***влияет*** на элемент *j* в системе или что элемент *j* ***зависит*** от элемента *i*, если вещество, энергия, информация, из элемента *i*, преобразуясь промежуточными элементами, попадает в элемент *j*. На графе структуры это выражается существованием пути из вершины *i* в вершину *j*. В теории графов существование пути означает, что вершина *j* ***достижима*** из вершины *i* или, что вершины *i* и *j* ***связаны***. Подмножество вершин, зависящих от данной $v \in V$, будет обозначаться $\Gamma^{\leftarrow} v$, а подмножество вершин, влияющих на данную,– $\Gamma^{\rightarrow} v$. Два элемента ***непосредственно связаны*** между собой, если вещество,

энергия, информация с выхода одного элемента сразу попадает на вход другого элемента, т.е. в графе это смежные вершины.

Достижимость выходов системы из входов означает, что потоки вещества, энергии, информации в системе поступают от входов системы проходят через промежуточные элементы, преобразовываясь и изменяясь в них, и подаются на выходы системы. Отсутствие достижимости выходов системы из входов системы обычно свидетельствует о том, что в ней нарушается прохождение потоков от входов к выходам. Если поток не может пройти от определенного входа к выходу системы, а должен проходить, то структура содержит ошибки. Причиной такой ошибки может быть, например, неправильное соединение элементов. Эта задача в терминах теории графов формулируется как задача проверки достижимости одного подмножества вершин из другого подмножества вершин, при условии, что эти подмножества не пересекаются.

В принципе, решать задачу проверки достижимости можно с помощью прямого перечисления всех путей и циклов в графе, но если требуется проверить достижимость одной или нескольких вершин или построить для них множество влияющих или зависящих вершин, то есть более простой алгоритм, не требующий составления громоздких матриц. Этот алгоритм использует то свойство, что если вершины непосредственно связаны, то они образуют одно множество достижимости.

Пусть v – заданная вершина в графе системы G=(V,U). Алгоритмы отыскания подмножеств зависящих $\Gamma^{\leftarrow}v$ или влияющих $\Gamma^{\rightarrow}v$ вершин аналогичны и осуществляют последовательное построение множества достижимых вершин D, проверяя окрестности вершин по выходам или по входам соответственно. При этом найденные окрестности помещаются в специальное множество K, из которого потом выбирается новая вершина для поиска окрестностей. **Алгоритм отыскания подмножества зависящих вершин $\Gamma^{\leftarrow}v$** следующий:

1. Положить, что начальное множество D содержит только заданную вершину v, т.е. $D=\{v\}$. Текущая вершина $w=v$. Положить $K=\varnothing$.
2. Найти окрестность по выходам для текущей вершины $\Gamma^{-}w$.

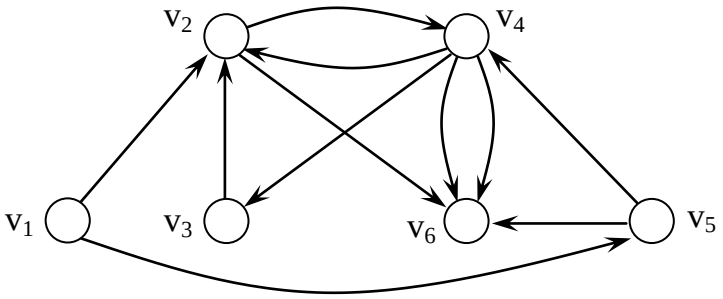
3. В множество K включаются такие вершины из окрестности $\Gamma^{-}w$, которых нет в множестве D, т.е.

$$K = K \cup \Gamma^{-}w - D$$

4. Проверить, если K=∅, т.е. новых вершин не нашлось, то перейти к п.7, иначе перейти к п.5.
5. Извлечь из K вершину и сделать ее текущей, т.е. $w \in K$, $K=K-\{w\}$.
6. Поместить текущую вершину в множество достижимости, т.е. $D=D\cup\{w\}$, и перейти к п.2.
7. Удалить из D заданную вершину, т.е. $\Gamma^{\leftarrow}v=D-\{v\}$.

Для отыскания подмножества вершин $\Gamma^{\rightarrow}v$, влияющих на заданную вершину v, используется аналогичный алгоритм, с той лишь разницей, что нужно использовать окрестность по входам $\Gamma^{+}w$.

Рассмотрим работу алгоритма нахождения подмножества зависящих вершин для вершины v_1 графа, представленного на рис.15.



$$A^{д} = \begin{vmatrix} 0&1&1&1&1&1 \\ 0&0&1&1&0&1 \\ 0&1&0&1&0&1 \\ 0&1&1&0&0&1 \\ 0&1&1&1&0&1 \\ 0&0&0&0&0&0 \end{vmatrix}$$

Рисунок 15 – Пример графа для проверки достижимости

Положить $D=\{v_1\}$, K=∅, текущая вершина $w=v_1$.

шаг 1. Нашли окрестность w: $\Gamma^{-}v_1=\{v_2,v_5\}$

Вычислить: $K=\{v_2,v_5\}-\{v_1\}=\{\cancel{v_2},v_5\}$

Извлекаем и делаем текущей $w=v_2$, $K=K-\{v_2\}=\{v_5\}$.

$D=D\cup\{v_2\}=\{v_1,v_2\}$

шаг 2. Нашли окрестность w: $\Gamma^{-}v_2=\{v_4,v_6\}$

Вычислить: $K=\{v_5\}\cup\{v_4,v_6\}-\{v_1,v_2\}=\{\cancel{v_4},v_5,v_6\}$ - $w=v_4$.

$D=D\cup\{v_4\}=\{v_1,v_2,v_4\}$

шаг 3. Нашли окрестность w: $\Gamma^{-}v_4=\{v_2,v_3,v_6\}$

Вычислить: $K=\{v_5,v_6\}\cup\{\cancel{v_2},v_3,v_6\}-\{v_1,\underline{v_2},v_4\}=\{\cancel{v_3},v_5,v_6\}$ - $w=v_3$,

$D=D\cup\{v_3\}=\{v_1,v_2,v_3,v_4\}$

шаг 4. Нашли окрестность *w*: $\Gamma^{-}v_3=\{v_2\}$

Вычислить: $K=\{v_5,v_6\}\cup\{\cancel{v_2}\}-\{v_1,\underline{v_2},v_3,v_4\}=\{\cancel{v_5},v_6\}$ - $w=v_5$,

$D=D\cup\{v_5\}=\{v_1,v_2,v_3,v_4,v_5\}$

шаг 5. Нашли окрестность *w*: $\Gamma^{-}v_5=\{v_4,v_6\}$

Вычислить: $K=\{v_6\}\cup\{\cancel{v_4},v_6\}-\{v_1,v_2,v_3,\underline{v_4},v_5\}=\{\cancel{v_6}\}$ - $w=v_6$,

$D=D\cup\{v_6\}=\{v_1,v_2,v_3,v_4,v_5,v_6\}$

шаг 6. Нашли окрестность *w*: $\Gamma^{-}v_6=\{\}$ – вершин нет, $K=\varnothing$.

Поскольку в множестве К больше нет вершин, алгоритм останавливается, $D-\{v_1\}$. Множество зависящих вершин $\Gamma^{\leftarrow}v_1=\{v_2,v_3,v_4,v_5,v_6\}$.

Результаты проверки достижимости удобно представлять в виде **матрицы достижимости** $\mathbf{A}^{д}$ графа. Элементы матрицы достижимости строятся следующим образом:

$$a_{ij}=\begin{cases}1\text{ , если из вершины }i\text{ к вершине }j\text{ имеется путь ;}\\0\text{ , в противном случае.}\end{cases}\qquad(15)$$

По матрице видно, что если $a_{ij}=1$, то вершина *j* достижима из вершины *i*. Для графа на рис. 15, матрица достижимости представлена рядом. По матрице видно, например, что вершина v_4 достижима из вершин v_1, v_2, v_3, v_5 и недостижима из вершины v_6, а вершина v_5 достижима из вершины v_1 и недостижима из вершин v_2, v_3, v_4, v_6. Вершина v_1 влияет на все вершины и не зависит ни от какой вершины графа (вход в системе), а вершина v_6, зависит от всех вершин, но ни на какую вершину не влияет (выход в системе).

Если на графе существует путь из вершины *i* в вершину *j* и существует путь из вершины *j* в вершину *i*, причем эти два пути необязательно должны проходить через одни и те же вершины, то такие вершины называются ***взаимодостижимыми***. Например, в графе на рис. 15 вершины v_2,v_3,v_4 являются взаимодостижимыми. Поскольку путь из вершины *i* заканчивающийся в вершине *j* можно соединить с путем, начинающимся в вершине *j* и заканчивающимся в вершине *i*, то, очевидно, взаимодостижимые вершины всегда образуют некоторый цикл, которой проходит через них. Таким образом, используя свойство взаимной достижимости, можно осуществлять поиск замкнутых подсистем в графе системы.

Ранг вершины – это отношение количества достижимых путей от данной вершины до всех остальных вершин графа к общему количеству достижимых путей в графе. Ранг позволяет ранжировать элементы системы в порядке их значимости. Вопрос о том, какой элемент более значимый, а какой – менее значимый встает тогда, когда необходимо распределить усилия по обеспечению надежности отдельных элементов в системе. Иначе говоря, необходимо знать, какое влияние оказывает на общую надежность системы выход из строя или ошибочная работа того или иного элемента. Чем выше ранг элемента, тем более сильно он связан с другими элементами системы и тем более ощутимыми будут последствия при выходе из строя или при ошибочной работе этого элемента. Ранги элементов вычисляются по следующей формуле:

$$\mathbf{R} = \mathbf{A} + \mathbf{A} * \mathbf{A} \qquad (16)$$

где **R** – матрица результата,

A – матрица достижимости.

Ранг *i*-ой вершины определяется как отношение суммы элементов *i*-ой строки матрицы **R** к общей сумме элементов матрицы **R.**

Например, для графа, представленного на рис. 15, ранги вычисляются следующим образом:

$$R = A+A*A = \begin{vmatrix} 0&1&1&1&1&1\\ 0&0&1&1&0&1\\ 0&1&0&1&0&1\\ 0&1&1&0&0&1\\ 0&1&1&1&0&1\\ 0&0&0&0&0&0 \end{vmatrix} + \begin{vmatrix} 0&1&1&1&1&1\\ 0&0&1&1&0&1\\ 0&1&0&1&0&1\\ 0&1&1&0&0&1\\ 0&1&1&1&0&1\\ 0&0&0&0&0&0 \end{vmatrix} * \begin{vmatrix} 0&1&1&1&1&1\\ 0&0&1&1&0&1\\ 0&1&0&1&0&1\\ 0&1&1&0&0&1\\ 0&1&1&1&0&1\\ 0&0&0&0&0&0 \end{vmatrix} =$$

$$= \begin{vmatrix} 0&1&1&1&1&1\\ 0&0&1&1&0&1\\ 0&1&0&1&0&1\\ 0&1&1&0&0&1\\ 0&1&1&1&0&1\\ 0&0&0&0&0&0 \end{vmatrix} + \begin{vmatrix} 0&3&3&3&0&4\\ 0&2&1&1&0&2\\ 0&1&2&1&0&2\\ 0&1&1&2&0&2\\ 0&2&2&2&0&3\\ 0&0&0&0&0&0 \end{vmatrix} = \begin{vmatrix} 0&4&4&4&1&5\\ 0&2&2&2&0&3\\ 0&2&2&2&0&3\\ 0&2&2&2&0&3\\ 0&3&3&3&0&4\\ 0&0&0&0&0&0 \end{vmatrix} \begin{vmatrix} 18\\ 9\\ 9\\ 9\\ 13\\ 0 \end{vmatrix}$$

Получаем: R’={18,9,9,9,13,0}, ΣR’=58 и ранги вершин равны соответственно r_1=18/58=0.310, r_2=9/58=0.155, r_3=9/58=0.155, r_4=9/58=0.155, r_5=13/58=0.225, r_6=0. Как видно из полученных результатов, наибольшим рангом обладает вершина v_1, а наименьшим – вершина v_6.

Из графа видно, что вершина v_1 имеет пути ко всем остальным вершинам графа и она является входом системы, поэтому её влияние на другие вершины наибольшее и ранг максимальный. Напротив, вершина v_6 не имеет путей к остальным вершинам графа и она является выходом системы. У данного элемента отсутствует влияние на остальные элементы, поэтому ранг равен нулю. Элементы v_2, v_3, v_4, имеют одинаковые ранги, что свидетельствует о их одинаковой значимости в системе. Элемент v_5 имеет немного большую значимость, чем элементы v_2, v_3, v_4, поскольку непосредственно связан с входом и выходом.

9 Сильные компоненты связанности

Граф, содержащий хотя бы один цикл, называется ***циклическим***. В циклическом графе анализ прохождения потоков вещества, энергии, информации через систему затруднителен. В большинстве случаев, структура системы, являющаяся циклическим графом состоит из замкнутых подсистем, каждая из которых является циклическим подграфом меньшей размерности, чем граф всей системы. Значительно проще исследовать каждую из замкнутых подсистем в отдельности, а затем исследовать влияние этих подсистем друг на друга. Поэтому особый интерес для анализа циклических графов представляет задача отыскания в структуре системы всех замкнутых подсистем, для решения которой эффективно применяются графовые методы.

В теории графов существует понятие ***компоненты связанности*** – это такой максимальный подграф, каждая вершина которого связана с другой вершиной этого подграфа. Максимальность подграфа означает, что никакая другая вершина графа не может быть включена в подграф без нарушения свойства связанности. Компоненты связанности делятся на сильные и слабые.

Сильная компонента связанности – это максимальный подграф, в котором любая пара вершин взаимодостижима. Взаимная достижимость вершин означает, что существует путь из вершины *i* в вершину *j* и существует путь из вершины *j* в вершину *i*, причем эти два пути не обязательно должны проходить через одни и те же вершины. Такие подграфы также называют циклическими.

Слабая компонента связанности – это максимальный подграф, который после дезориентации дуг становится сильно связанным. В слабой компоненте, в отличие от сильной, имеются пары вершин достижимые только в одном направлении, но если в путях между такими парами вершин дезориентировать некоторые дуги, то вершины станут взаимодостижимыми.

Практическое значение для нахождения замкнутых подсистем в графе системы имеет только поиск сильных компонент связанности. Сильная компонента связанности соответствует подсистеме, в которой все элементы оказывают взаимное влияние друг на друга, что возможно только при наличии обратных связей между элементами, которые образуют циклы в графе системы. Благодаря свойству взаимной достижимости, сильная компонента является объединением циклов входящих в нее вершин. Каждый цикл в графе системы принадлежит к одной из её сильных компонент связанности или, другими словами, циклы сложной системы концентрируются в сильных компонентах связанности.

Сильную компоненту связанности могут образовывать не только циклически связанные вершины. Изолированные вершины, истоки, стоки, подграфы из двух вершин, связанных противоположно направленными дугами, каждая вершина в ациклических цепочках, образуют самостоятельные компоненты связанности. Естественно, такого рода компоненты связанности не являются замкнутыми подсистемами и их не рассматривают при анализе циклов. В рассмотрение как замкнутая подсистема берутся сильные компоненты с числом вершин больше трёх.

Для отыскания сильных компонент связанности используется алгоритм, который последовательно разбивает граф системы G=(V,U). На текущем шаге выбирается вершина $v \in V$, для которой строится подмножество зависящих

вершин $\Gamma^{\leftarrow}v$ и подмножество влияющих вершин $\Gamma^{\rightarrow}v$. На пересечении подмножеств $\Gamma^{\leftarrow}v \cap \Gamma^{\rightarrow}v$ получается подмножество вершин, которые достижимы из верщины и одновременно из которых вершина достигается, т.е. подмножество взаимодостижимых вершин. Затем выбирается новая вершина v из оставшихся вершин графа и вычисления повторяются для новой компоненты. Пусть K_i – сильная компонента на i-ом шаге.

Алгоритм поиска сильных компонент связанности следующий:

1. Положить M равному количеству вершин в графе M=|V| и i=1.
2. Выбрать в графе вершину v, не отнесенную ни к одной из сильных компонент, т.е. $v \in (V - \cup_{j=1}^{i} K_i)$.
3. Применить описанный метод для нахождения сильной компоненты для заданной вершины v. Для этого нужно:
   3.1. Найти подмножество зависящих вершин, т.е. $\Gamma^{\leftarrow}v$.
   3.2. Найти подмножество влияющих вершин, т.е. $\Gamma^{\rightarrow}v$.
   3.3. Найти пересечение этих подмножеств и включить в него заданную вершину v, т.е. $K_i = (\Gamma^{\leftarrow}v \cap \Gamma^{\rightarrow}v) \cup \{v\}$.
4. Уменьшить M на количество вершин в компоненте K_i, т.е. $M = M - |K_i|$.
5. Если все вершины уже отнесены к своим компонентам, т.е. M=0, то остановить вычисления, иначе i=i+1 и перейти к п.2.

Последовательность подмножеств $K_1,...,K_i$, являются искомыми сильными компонентами связанности. Далее, из найденных сильных компонент выбираются те, у которых мощность больше трёх вершин.

Рассмотрим работу алгоритма на примере графа, представленного на рис. 16. Выкладки при нахождении подмножества зависящих $\Gamma^{\leftarrow}v$ и подмножества влияющих $\Gamma^{\rightarrow}v$ вершин будем проводить сокращенно и одновременно, поскольку метод их нахождения был подробно рассмотрен в предыдущем разделе.

Прежде всего, как видно из графа, вершина v_1 является истоком, а вершина v_8 является стоком. Как отмечалось выше, истоки и стоки являются отдельными сильными компонентами связанности, проверим это с помощью п.3 алгоритма. Итак, поскольку у вершины-истока v_1 нет входящих дуг, то

очевидно, подмножество влияющих вершин будет пустым $\Gamma^{\rightarrow}v_1=\{\}$, хотя подмножество зависящих вершин не пусто и содержит все вершины графа $\Gamma^{\leftarrow}v_1=V$, тем не менее пересечение этих подмножеств – $\Gamma^{\leftarrow}v_1\cap\Gamma^{\rightarrow}v_1=\{\}\cap V=\{\}$ – будет пустым, согласно алгоритму в сильную компоненту включается только вершина v_1, т.е. $K_1=(\Gamma^{\leftarrow}v_1\cap\Gamma^{\rightarrow}v_1)\cup\{v_1\}=\{\}\cup\{v_1\}=\{v_1\}$. Аналогично, поскольку у вершины-стока v_8 нет исходящих дуг, то подмножество зависящих вершин будет пустым $\Gamma^{\leftarrow}v_8=\{\}$, хотя подмножество влияющих вершин содержит все вершины графа $\Gamma^{\rightarrow}v_8=V$, их пересечение будет пустым, и в сильную компоненту связанности включается только вершина v_8, т.е. $K_2=(\Gamma^{\leftarrow}v_8\cap\Gamma^{\rightarrow}v_8)\cup\{v_8\}=\{v_8\}$.

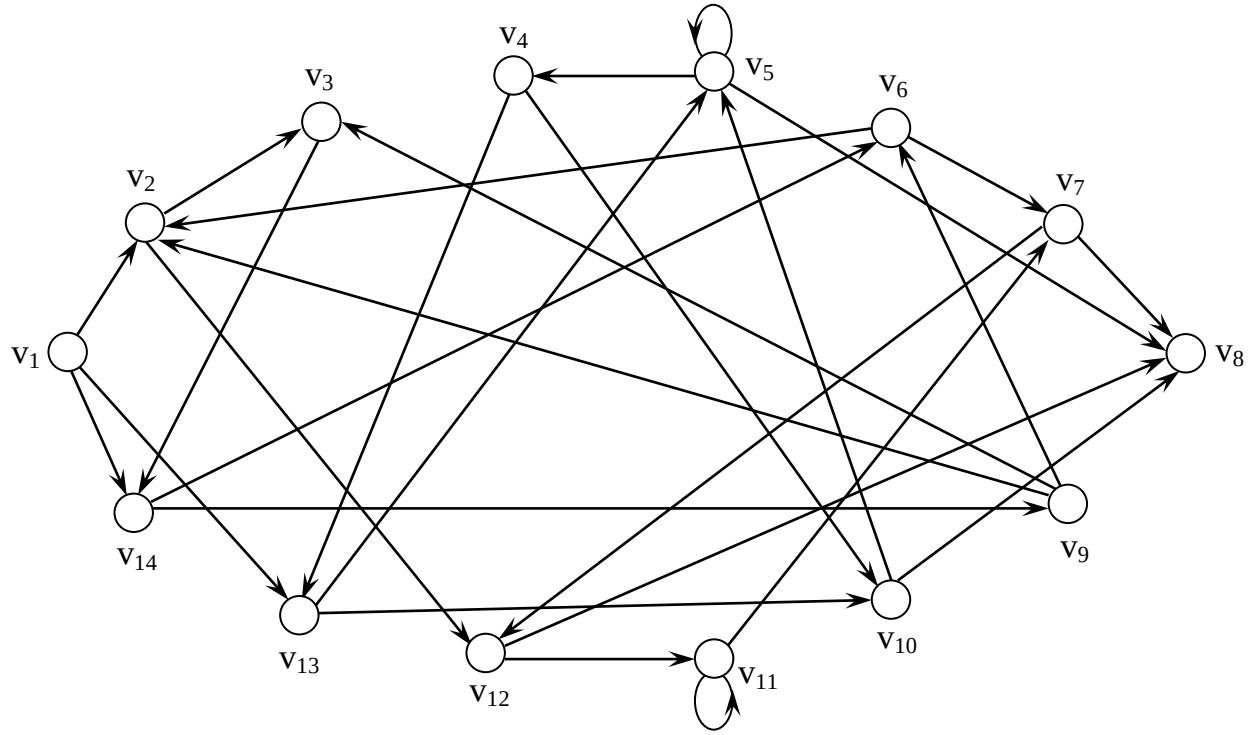


Рисунок 16 – Пример графа для поиска сильных компонент

Итак, имеем $K_1=\{v_1\}$ и $K_2=\{v_8\}$. Теперь, продемонстрируем поиск более сложных сильных компонент, для этого полагаем i=3, $M=|V|-|K_1|-|K_2|=14-1-1=12$, и согласно п.2 алгоритма произвольно выбираем вершину, не отнесенную ни к одной из компонент, например вершину v_2.

Согласно п. 3.1 и 3.2, для вершины v_2 находим подмножества зависящих $\Gamma^{\leftarrow}v_2$ и влияющих $\Gamma^{\rightarrow}v_2$ вершин. Для этого полагаем множества достижимости $D^{\leftarrow}=\{v_2\}$ и $D^{\rightarrow}=\{v_2\}$.

шаг 1. берём окрестности по выходам и по входам для вершины v_2 и помещаем все вершины в соответствующее множество достижимости:

$D^{\leftarrow}=D^{\leftarrow}\cup\Gamma^{-}v_2=\{v_2,v_3,v_{12}\}$, | $D^{\rightarrow}=D^{\rightarrow}\cup\Gamma^{+}v_2=\{v_2,v_1,v_6,v_9\}$.

шаг 2. берём окрестности по выходам для вершин v_3, v_{12} и окрестности по входам для вершин v_6, v_9:

$\Gamma^{-}v_3=\{v_{14}\}$, $\Gamma^{-}v_{12}=\{v_8,v_{11}\}$ | $\Gamma^{+}v_6=\{v_9,v_{14}\}$, $\Gamma^{+}v_9=\{v_{14}\}$, $\Gamma^{+}v_1=\{\}$

и помещаем все найденные вершины в соответствующее множество достижимости:

$D^{\leftarrow}=D^{\leftarrow}\cup\Gamma^{-}v_3\cup\Gamma^{-}v_{12}=$
$=\{v_2,v_3,v_{12}\}\cup\{v_{14}\}\cup\{v_8,v_{11}\}=$
$=\{v_2,v_3,v_{12},\underline{v_8},\underline{v_{11}},\underline{v_{14}}\}$

$D^{\rightarrow}=D^{\rightarrow}\cup\Gamma^{+}v_1\cup\Gamma^{+}v_6\cup\Gamma^{+}v_9=$
$=\{v_2,v_1,v_6,v_9\}\cup\{\}\cup\{\sout{v_9},v_{14}\}\cup\{\sout{v_{14}}\}=$
$=\{v_1,v_2,v_6,v_9,\underline{v_{14}}\}$

шаг 3. для новых вершин, найденных на предыдущем шаге, берём окрестности по выходам для вершин v_8, v_{11}, v_{14} и окрестность по входам для вершины v_{14}:

$\Gamma^{-}v_8=\{\}$, $\Gamma^{-}v_{11}=\{v_7\}$, $\Gamma^{-}v_{14}=\{v_6,v_9\}$ | $\Gamma^{+}v_{14}=\{v_1,v_3\}$,

и помещаем их в соответствующее множество достижимости:

$D^{\leftarrow}=D^{\leftarrow}\cup\Gamma^{-}v_8\cup\Gamma^{-}v_{11}\cup\Gamma^{-}v_{14}=$
$=\{v_2,v_3,v_8,v_{11},v_{12},v_{14}\}\cup\{\}\cup\{v_7\}$
$\cup\{v_6,v_9\}=\{v_2,v_3,v_8,v_{11},v_{12},v_{14},\underline{v_6},\underline{v_7},\underline{v_9}\}$

$D^{\rightarrow}=D^{\rightarrow}\cup\Gamma^{+}v_{14}=\{v_1,v_2,v_6,v_9,v_{14}\}\cup$
$\{\sout{v_1},v_3\}=\{v_1,v_2,v_6,v_9,v_{14},\underline{v_3}\}$

шаг 4. для новых вершин, найденных на предыдущем шаге, берём окрестности по выходам для вершин v_6, v_7, v_9 и окрестность по входам для вершины v_3:

$\Gamma^{-}v_6=\{v_2,v_7\}$, $\Gamma^{-}v_7=\{v_8,v_{12}\}$, $\Gamma^{-}v_9=\{v_2,v_3,v_6\}$, $\Gamma^{+}v_3=\{v_2,v_9\}$,

и помещаем их в соответствующее множество достижимости:

$D^{\leftarrow}=D^{\leftarrow}\cup\Gamma^{-}v_6\cup\Gamma^{-}v_7\cup\Gamma^{-}v_9=$
$=\{v_2,v_3,v_8,v_{11},v_{12},v_{14},v_6,v_7,v_9\}\cup$
$\{\sout{v_2},\sout{v_7}\}\cup\{\sout{v_8},\sout{v_{12}}\}\cup\{\sout{v_2},\sout{v_3},\sout{v_6}\}=$
$=\{v_2,v_3,v_6,v_7,v_8,v_9,v_{11},v_{12},v_{14}\}$

$D^{\rightarrow}=D^{\rightarrow}\cup\Gamma^{+}v_3=\{v_1,v_2,v_3,v_6,v_9,v_{14}\}$
$\cup\{\sout{v_2},\sout{v_9}\}=\{v_1,v_2,v_3,v_6,v_9,v_{14}\}$

На этом шаге новых вершин не появилось, значит подмножества зависящих и влияющих вершин сформировались:

$\Gamma^{\leftarrow}v_2=D^{\leftarrow}-\{v_2\}=\{v_3,v_6,v_7,v_8,v_9,v_{11},v_{12},v_{14}\}$,

$\Gamma^{\rightarrow}v_2=D^{\rightarrow}-\{v_2\}=\{v_1,v_3,v_6,v_9,v_{14}\}$.

Согласно п. 3.3 алгоритма, новая сильная компонента связанности находится как пересечение подмножеств зависящих и влияющих вершин:

$K_3=\Gamma^{\leftarrow}v_2\cap\Gamma^{\rightarrow}v_2\cup\{v_2\}=\{\underline{v_3},\underline{v_6},v_7,v_8,\underline{v_9},v_{11},v_{12},\underline{v_{14}}\}\cap\{v_1,\underline{v_3},\underline{v_6},\underline{v_9},\underline{v_{14}}\}\cup\{v_2\}=$
$=\{v_2,v_3,v_6,v_9,v_{14}\}$

Согласно п.4 алгоритма, $M=M-|K_3|=12-5=7$, значит в графе ещё есть сильные компоненты связанности, продолжаем поиск, $i=i+1=4$.

Согласно п.2 алгоритма, произвольно выбираем новую вершину, например v_5. Полагаем $D^{\leftarrow}=\{v_5\}$ и $D^{\rightarrow}=\{v_5\}$. Выкладки следующие:

шаг 1. берём окрестности по выходам и по входам для вершины v_5 и помещаем все вершины в соответствующее множество достижимости:

$D^{\leftarrow}=D^{\leftarrow}\cup\Gamma^{-}v_5=\{v_5,v_4,v_8\}$ | $D^{\rightarrow}=D^{\rightarrow}\cup\Gamma^{+}v_5=\{v_5,v_{10},v_{13}\}$.

шаг 2. берём соответствующие окрестности новых найденных вершин:

$\Gamma^{-}v_4=\{v_{10},v_{13}\},\ \Gamma^{-}v_8=\{\}$ | $\Gamma^{+}v_{10}=\{v_4,v_{13}\},\ \Gamma^{+}v_{13}=\{v_1\}$

и помещаем их в соответствующее множество достижимости:

$D^{\leftarrow}=\{v_5,v_4,v_8\}\cup\{v_{10},v_{13}\}\cup\{\}=$
$=\{v_5,v_4,v_8,\underline{v_{10}},\underline{v_{13}}\}$ | $D^{\rightarrow}=\{v_5,v_{10},v_{13}\}\cup\{v_4,\sout{v_{13}}\}\cup\{v_1\}=$
$=\{v_5,v_{10},v_{13},\underline{v_4},\underline{v_1}\}$

шаг 3. берём соответствующие окрестности новых найденных вершин:

$\Gamma^{-}v_{10}=\{v_5,v_8\},\ \Gamma^{-}v_{13}=\{v_5,v_{10}\}$ | $\Gamma^{+}v_1=\{\},\ \Gamma^{+}v_4=\{v_5\}$

и помещаем их в соответствующее множество достижимости:

$D^{\leftarrow}=\{v_4,v_5,v_8,v_{10},v_{13}\}\cup$
$\{\sout{v_5},\sout{v_8}\}\cup\{\sout{v_5},\sout{v_{10}}\}=\{v_4,v_5,v_8,v_{10},v_{13}\}$ | $D^{\rightarrow}=\{v_5,v_{10},v_{13},v_4,v_1\}\cup\{\}\cup\{\sout{v_5}\}=$
$=\{v_1,v_4,v_5,v_{10},v_{13}\}$

Согласно п. 3.3 новая сильная компонента связанности находится как пересечение подмножеств зависящих и влияющих вершин:

$K_4=\Gamma^{\leftarrow}v_5\cap\Gamma^{\rightarrow}v_5\cup\{v_5\}=\{\underline{v_4},v_8,\underline{v_{10}},\underline{v_{13}}\}\cap\{v_1,\underline{v_4},\underline{v_{10}},\underline{v_{13}}\}\cup\{v_5\}=\{v_4,v_5,v_{10},v_{13}\}$

Согласно п.4, $M=M-|K_4|=7-4=3$, значит в графе ещё есть сильные компоненты связанности, продолжаем поиск, $i=i+1=5$.

Нетрудно видеть по рис. 16, что оставшиеся вершины v_7, v_{11}, v_{12} образуют цикл – угольник, и следовательно образуют последнюю сильную компоненту $K_5=\{v_7,v_{11},v_{12}\}$, хотя можно было бы проделать все выкладки и найти её как было описано выше. Теперь $M=3-3=0$ и все сильные компоненты найдены, остановка алгоритма.

В результате получены следующие сильные компоненты связанности: $K_1=\{v_1\}$, $K_2=\{v_6\}$, $K_3=\{v_2,v_3,v_6,v_9,v_{14}\}$, $K_4=\{v_4,v_5,v_{10},v_{13}\}$, $K_5=\{v_7,v_{11},v_{12}\}$.

Результат разбиения исходного графа на сильные компоненты связанности представлен на рис. 17.

На последующих этапах проектирования каждую замкнутую подсистему можно исследовать отдельно. Однако часто возникает необходимость исследовать влияние замкнутых подсистем друг на друга. Для этого каждая из замкнутых подсистем представляется в виде одной обобщенной вершины, а несколько дуг, связывающих две подсистемы между собой, представляются одной обобщенной дугой. В результате, получается граф, показывающий взаимосвязи между замкнутыми подсистемами и влияние процессов в них друг на друга, который называется ***графом конденсации*** циклов. Для графа на рис. 17 граф конденсации представлен на рис. 18.

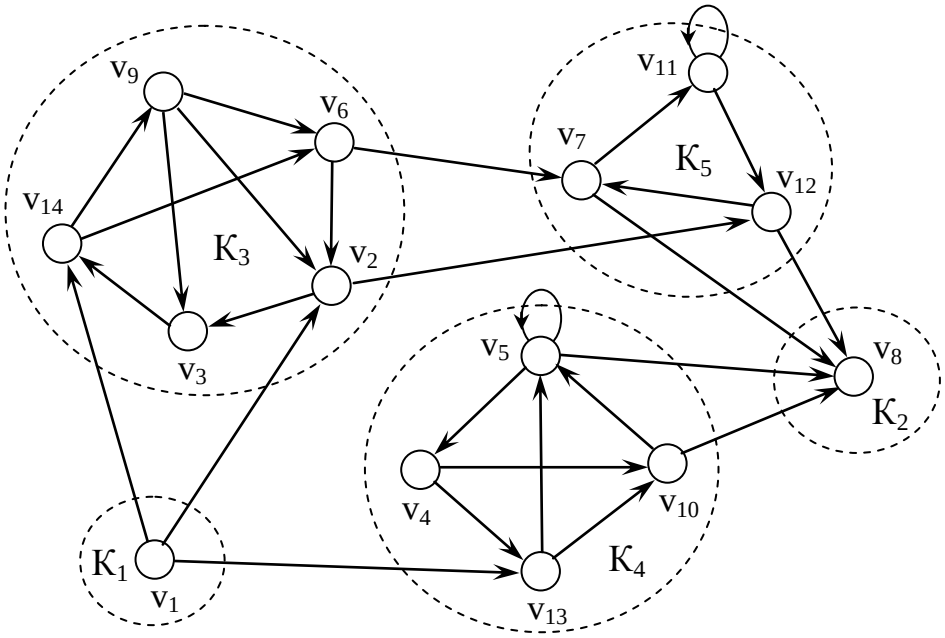


Рисунок 17 – Результат разбиения графа на рисунке 16.

Граф конденсации не содержит циклов, так как, если бы в нём хотя бы пара обобщенных вершин была бы связана циклом, то такие вершины были бы взаимодостижимыми, а следовательно были бы взаимодостижимыми и все вершины исходного графа, которые вошли в эти обобщенные вершины графа конденсации, но поскольку в результате операции разбиения графа на сильные компоненты связанности все взаимодостижимые вершины уже помещены в свои компоненты, то это противоречит существованию циклов между вершинами разных сильных компонент, следовательно ***граф конденсации ациклический***.

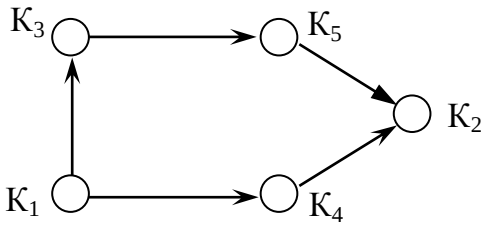


Рисунок 18 – Граф конденсации графа на рисунке 17.

Имея в виду интерпретацию графа системы, можно сказать, что сильные компоненты связанности разбивают общий процесс в системе на совокупность процессов, развивающихся в каждой из замкнутых подсистем. Одностороннее влияние процессов друг на друга означает, что процесс замкнутой подсистемы А воздействует на процесс замкнутой подсистемы В, но процесс В не воздействует на процесс А.

10 Порядковые уровни

Граф, не содержащий циклов, называется ***ациклическим*** или ***деревом***. Система, являющаяся ациклическим графом обладает очень важным свойством – в ней потоки вещества, энергии, информации передаются от входа к выходу последовательно проходя через промежуточные элементы. Анализ системы значительно упрощается, если упорядочить элементы системы от входа к выходу, при этом, система как бы разбивается на уровни.

Элементы системы на основе ациклического графа можно разбить на уровни по порядку прохождения через них потоков так, чтобы на каждом уровне элементы не были связаны между собой, при этом связь по выходным потокам уровня осуществлялась только с элементами более высоких уровней, а по входным потокам – только с элементами менее низких уровней. Внутри уровня элементы не взаимодействуют между собой по потокам. Таким образом, на каждом уровне элементы системы функционируют независимо друг от друга, входные потоки на уровень поступают только с нижних уровней, выходные обработанные потоки передаются только на верхние уровни. Самый нижний уровень составляют входы системы, т.е. истоки вещества, энергии, информации, а самый верхний уровень – выходы системы, т.е. стоки.

Порядковыми уровнями графа G=(V,U) называется такое разбиение множества вершин графа на непересекающиеся упорядоченные по возрастанию номера подмножества, т.е. $V_1 \subset V$, $V_2 \subset V$, ..., $V_N \subset V$ и $V_1 \cup V_2 \cup ... \cup V_N = V$ и $V_1 \cap V_2 \cap ... \cap V_N = \varnothing$, что внутри этих подмножеств вершины не смежны между собой, т.е. $V_1 \times V_1 = \varnothing$, $V_2 \times V_2 = \varnothing$, ..., $V_N \times V_N = \varnothing$, а смежны только с вершинами других подмножеств так, что дуги графа направлены только от вершин подмножеств с меньшим номером к вершинам подмножеств с большим номером, т.е. разрешены отношения: $V_1 \times V_2$, $V_1 \times V_3$, ..., $V_1 \times V_N$; $V_2 \times V_3$, $V_2 \times V_4$, ..., $V_2 \times V_N$; ..., $V_{N-2} \times V_{N-1}$, $V_{N-2} \times V_N$; $V_{N-1} \times V_N$.

Уровни называются ***последовательными***, если вершины каждого уровня смежны по выходам только с вершинами следующего уровня и смежны по входам только с вершинами предыдущего уровня.

Алгоритм разбиения графа на порядковые уровни последовательно строит подмножества вершин очередного уровня, которые смежны с вершинами нижних уровней и не смежны друг с другом. Для этого он на каждом шаге вычисляет полустепени ρ^+_i захода всех вершин. Вершины, у которых полустепень захода нулевая $\rho^+_i=0$, очевидно, не имеют вершин, смежных с ними по входу, а также не могут быть смежны друг с другом, поскольку если хотя бы одна вершина оказалась бы смежной, то её полустепень захода, была бы отлична от нуля $\rho^+_i \neq 0$, следовательно вершины с нулевой полустепенью захода образуют один порядковый уровень. Далее, алгоритм удаляет вершины найденного порядкового уровня вместе с их дугами из графа системы. После удаления снова пересчитываются полустепени захода оставшихся вершин и всё повторяется. Пусть G=(V,U) – граф системы. Обозначим L_n – подмножество вершин n-ого уровня.

Алгоритм разбиения графа на порядковые уровни следующий:

1. Положить M равному количеству вершин в графе M=|V| и n=1.
2. Вычислить полустепени захода всех вершин графа ρ^+_i, i=1,..,M
3. Включить в подмножество L_n такие вершины $v \in V$, у которых полустепень захода нулевая, т.е.

$$L_n = \{v \mid v \in V \text{ и } \rho^+_v = 0\}$$

4. Проверить, если в уровне вершин нет, т.е. $L_n=\varnothing$, то прекратить поиск уровней, поскольку граф содержит циклы, иначе п.5.
5. Удалить из графа вершины найденного уровня со всеми их дугами, т.е. $V=V-L_n$.
6. Уменьшить M на количество вершин в уровне L_n, т.е. $M=M-|L_n|$.
7. Если все вершины отнесены к своим уровням, т.е. M=0, то остановить вычисления, иначе новый уровень n=n+1 и перейти к п.2.

Последовательность подмножеств $L_1,...,L_n$ являются искомыми порядковыми уровнями системы.

Рассмотрим работу алгоритма на примере графа, представленного на рис. 19. Для условности, вершины, которые удаляются из графа будут обозначаться ±. Положим M=|V|=14, n=1.

шаг 1. Определяем L_1. Рассчитываем полустепени захода всех вершин графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	1	3	4	2	0	3	3	3	2	0	2	1	2	2

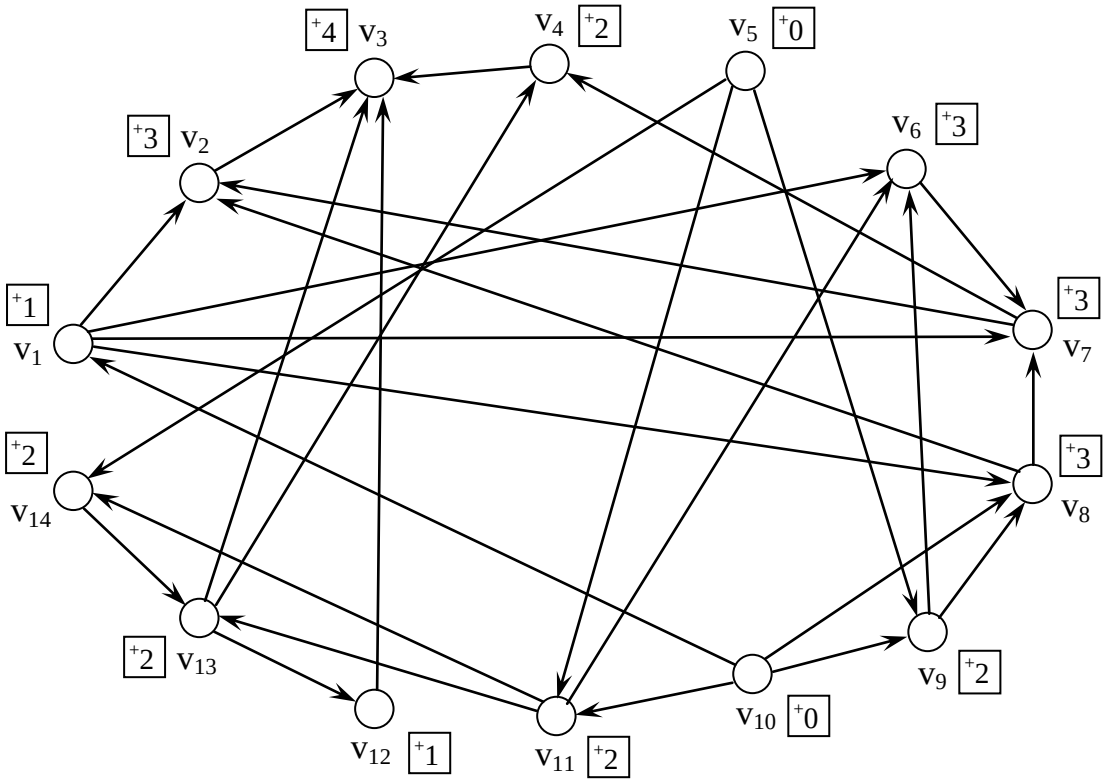


Рисунок 19 – Пример графа для поиска порядковых уровней

Получаем, что $\rho^+_5=0$ и $\rho^+_{10}=0$, и помещаем эти вершины в первый уровень $L_1=\{v_5,v_{10}\}$. Удаляем эти вершины из графа, т.е. $V=V-L_1$, а вместе с ними

удаляются исходящие дуги: (v_5,v_9), (v_5,v_{11}), (v_5,v_{14}), (v_{10},v_1), (v_{10},v_8), (v_{10},v_9), (v_{10},v_{11}). В результате полустепени захода у вершин v_1, v_8, v_{14} уменьшаются на единицу, а у вершин v_9 и v_{11} уменьшаются на два. $M=M-|L_1|=14-2=12$ – продолжаем, $n=n+1=2$. Результат первого шага представлен на рис. 20-а.

шаг 2. Определяем L_2. Рассчитываем полустепени захода всех вершин графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	0	3	4	2	±	3	3	2	0	±	0	1	2	1

Получаем, что $\rho^+_1=0$ и $\rho^+_9=0$ и $\rho^+_{11}=0$, и помещаем эти вершины во второй уровень $L_2=\{v_1,v_9,v_{11}\}$. Удаляем эти вершины из графа, т.е. $V=V-L_2$, а вместе с ними удаляются исходящие дуги: (v_1,v_2), (v_1,v_6), (v_1,v_7), (v_1,v_8), (v_9,v_6), (v_9,v_8), (v_{11},v_6), (v_{11},v_{13}), (v_{11},v_{14}). В результате полустепени захода у вершин v_2, v_7, v_{13},v_{14} уменьшаются на единицу, у вершины v_8 – на два, у вершины v_6 – на три. $M=M-|L_2|=12-3=9$ – продолжаем. Результат второго шага представлен на рис. 20-б.

шаг 3. Определяем L_3. Рассчитываем полустепени захода всех вершин графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	±	2	4	2	±	0	2	0	±	±	±	1	1	0

Получаем, что $\rho^+_6=0$ и $\rho^+_8=0$ и $\rho^+_{14}=0$, и помещаем эти вершины в третий уровень $L_3=\{v_6,v_8,v_{14}\}$. Удаляем эти вершины из графа, т.е. $V=V-L_3$, а вместе с ними удаляются исходящие дуги: (v_6,v_7), (v_8,v_7), (v_8,v_2), (v_{14},v_{13}). В результате полустепени захода у вершин v_2, v_{13} уменьшаются на единицу, а у вершины v_7 – на два. $M=M-|L_3|= 9-3=6$ – продолжаем. Результат третьего шага представлен на рис. 20-в.

шаг 4. Определяем L_4. Рассчитываем полустепени захода всех вершин графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	±	1	4	2	±	±	0	±	±	±	±	1	0	±

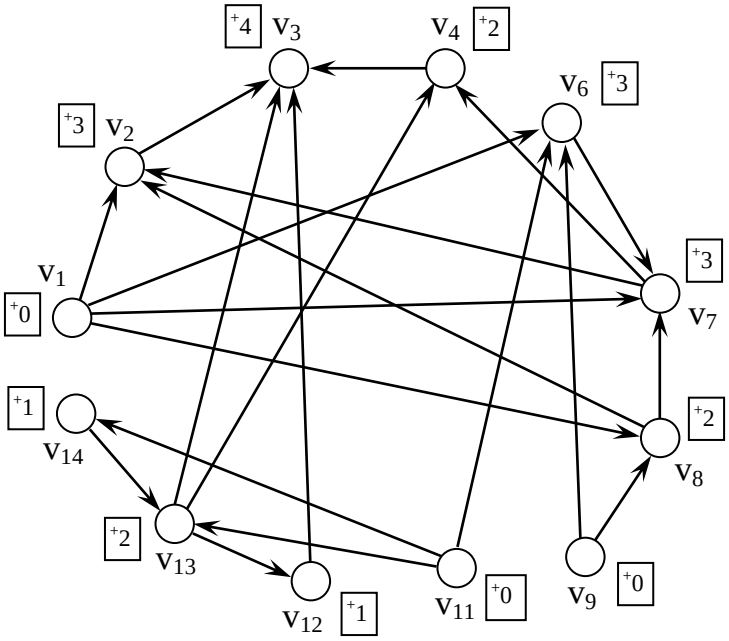
Получаем, что $\rho^+_7=0$ и $\rho^+_{13}=0$, и помещаем эти вершины в четвертый уровень $L_4=\{v_7,v_{13}\}$. Удаляем эти вершины из графа, а вместе с ними удаляются дуги: (v_7,v_2), (v_7,v_4), (v_{13},v_3), (v_{13},v_4), (v_{13},v_{12}). В результате полустепени захода у вершин v_2, v_3, v_{12} уменьшаются на единицу, а у вершины v_4 – на два. $M=M-|L_4|=6-2=4$ – продолжаем. Результат четвертого

шага представлен на рис. 20-г.

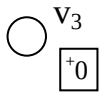
шаг 5. Определяем L_5. Рассчитываем полустепени захода всех вершин графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	±	0	3	0	±	±	±	±	±	±	±	0	±	±

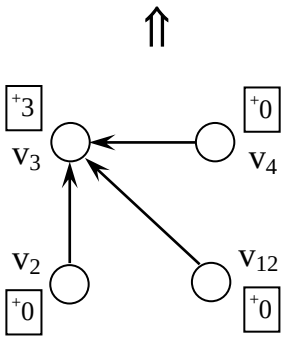
Получаем, что $\rho^+_2=0$ и $\rho^+_4=0$ и $\rho^+_{12}=0$, и помещаем эти вершины в пятый уровень $L_5=\{v_2,v_4,v_{12}\}$. Удаляем эти вершины из графа, а вместе с ними удаляются дуги: (v_2,v_3), (v_4,v_3), (v_{12},v_3). В результате полустепень захода у вершины v_3 уменьшается на три. $M=M-|L_5|=4-3=1$ – продолжаем. Результат шага пять представлен на рис. 20-д.



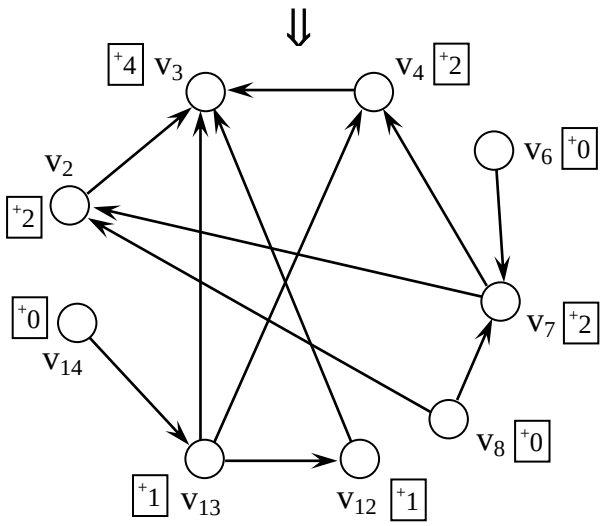
а). результат шага 1.



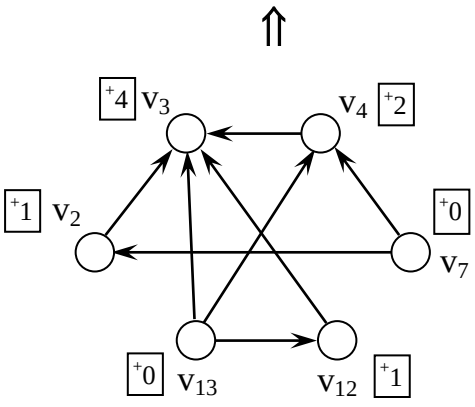
д). конец.



г). результат шага 4.



б). результат шага 2.



в). результат шага 3.

Рисунок 20 – Результаты шагов алгоритма разбиения графа на уровни

шаг 6. Определяем L_6. Рассчитываем полустепени захода последней вершины графа:

V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
ρ^+_i	±	±	0	±	±	±	±	±	±	±	±	±	±	±

Получаем, что $\rho^+_3=0$ и помещаем вершину v_3 в шестой уровень $L_6=\{v_3\}$. Удаляем ее из графа. $M=M-|L_6|=1-1=0$ – остановка.

Результат работы алгоритма – граф на рис. 19, разбитый на порядковые уровни, представлен на рис. 21.

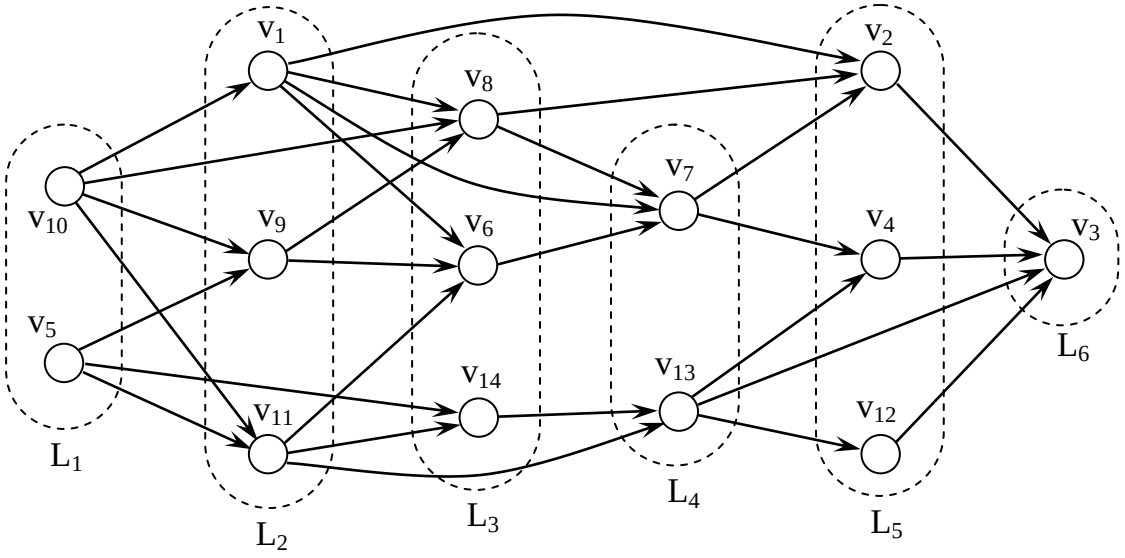


Рисунок 21 – Результат разбиения графа на порядковые уровни

Возможно разбиение графа системы на порядковые уровни и наоборот от выходов к входам системы, если потоки вещества, энергии, информации анализируются в обратном порядке. Тогда в алгоритме разбиения графа на порядковые уровни нужно вычислять полустепени исхода вершин, а в остальном, алгоритм такой же.

Алгоритм разбиения графа на уровни может использоваться также и для проверки цикличности графа. Если граф содержит цикл, то на шаге 4 на каком-то очередном уровне не окажется вершин, поскольку они замкнутся с предыдущими уровнями, и алгоритм будет прерван.

11 Метрика на графе и алгоритм Дейкстры

При проектировании или модернизации систем возникают задачи анализа потоков вещества, энергии, информации, которые могут передаваться от элемента к элементу. Соответственно через систему за заданное время может пройти только определенное количество потоков. Анализ прохождения потоков в зависимости от соединения элементов системы составляет специальный раздел анализа графов по связям.

Для анализа потоков системы с помощью графовых методов вводят функцию расстояния, зависящую от вершин и дуг графа системы. Пусть задан граф G=(V,U) и на графе определена ***функция расстояния*** $\mathbf{L(}v_i \to v_j\mathbf{)}$ между двумя вершинами *i* и *j* как количество дуг, входящих в путь, начинающейся в вершине *i* и заканчивающийся в *j*. Таким образом, каждой дуге сопоставляется единица условной длины. Под условной единицей можно понимать устройство в системе управления, единицу оборудования в технологической схеме, участок транспортной сети, и т.д. Функция расстояния обладает обычными метрическими свойствами:

1. $L(v_i \to v_i)=0$, если $v_i=v_j$, т.е. расстояние от элемента до самого себя равно нулю
2. $L(v_i \to v_j) \le L(v_i \to v_k) + L(v_k \to v_j)$, т.е. правило треугольника – расстояние через промежуточный элемент больше, чем расстояние непосредственно между двумя элементами.
3. $L(v_i \to v_k) = L(v_k \to v_i)$, т.е. расстояния от элемента до другого элемента и обратно одинаковы – всегда выполняется для неориентированных графов и только для некоторых видов ориентированных графов.

При таком определении расстояния предполагается, что поток через все элементы распространяется с одинаковой скоростью, поэтому количество дуг на графе является мерой прохождения. На практике, поток через разные элементы системы распространяется с различной скоростью. Поэтому естественнее ввести расстояние другим образом. Если ввести в граф некоторые технические характеристики самих элементов системы такие, как время прохождения потока через элемент, скорость движения потока, задержка потока в элементе, инерционность элемента, то можно провести более точный анализ системы по характеристикам.

Вещественное число **w**, сопоставленное дуге (вершине) графа системы и обозначающее некоторую характеристику элемента системы называется ***весом дуги*** (***вершины***). Если каждая дуга графа является некоторым устройством в системе, то число **w** может иметь значение скорости, времени прохождения потока через данное устройство, времени запаздывания устройства и т.д. Таким образом, в граф включаются технические характеристики элементов системы. ***Функция расстояния*** $\mathbf{W}(v_i \to v_j)$ между двумя вершинами *i* и *j* определяется как сумма весов дуг, входящих в путь, начинающейся в вершине *i* и заканчивающийся в *j*, т.е.

$$\mathbf{W}(v_i \to v_j) = \Sigma_k \, w(u_k), \text{ где } u_k \in s^{i \to j} \qquad (17)$$

В отличии от функции расстояния L, для функции расстояния W правило треугольника уже не выполняется, т.е. расстояние через промежуточную вершину может оказаться меньше, чем расстояние по дуге непосредственно соединяющей две вершины, если эта дуга имеет вес больший, чем сумма весов дуг соединяющих промежуточную вершину. Единственное чёткое свойство – $W(v_i \to v_i)=0$.

Анализ потоков системы по связям с помощью этих функций выполняется на графе системы. Известно, что чем меньше расстояние, т.е. чем короче путь, по которому поток проходит от одного элемента системы к другому, тем меньше ресурсов тратится на передачу данного потока. Чем больше расстояние между элементами, т.е. чем длиннее путь, тем больше ресурсов требуется на прохождение потока. Таким образом, в задачах анализа потоков возникает потребность нахождения путей с минимальным расстоянием, т.е. ***кратчайших путей***, или максимальным расстоянием, т.е. ***критических путей***.

Интерпретация кратчайшего (критического) пути между двумя вершинами зависит от определения функции расстояния. При первом определении функции расстояния, задача о кратчайшем пути связана с нахождением такой цепи в системе между двумя заданными элементами, по которой поток проходит наименьшее число элементов. Если в граф включены числовые характеристики элементов системы с помощью весов, то в этом случае, используется второе определение функции расстояния и это соответствует

нахождению цепи с минимальной характеристикой. Например, если вес обозначает время прохождения сигнала через устройство, то кратчайший путь соответствует самой быстрой цепи, а критический путь – самой медленной. Если вес обозначает стоимость передачи энергии через элемент системы, то кратчайший путь обозначает самый дешевый способ передачи энергии, а критический путь – самый дорогой.

Для решения задачи поиска кратчайших путей во взвешенном графе широко используется Алгоритм Де́йкстры, изобретённый нидерландским учёным Э. Дейкстрой в 1959 году. Этот алгоритм находит кратчайшее расстояние от одной вершины графа до всех остальных. Алгоритм работает только для графов, которые не имеют дуг отрицательного веса и широко применяется в программировании и сетевых технологиях. Алгоритм ставит в соответствие каждой вершине метку, которая сигнализирует о прохождении вершины, и потенциал, который хранит значение суммарного веса для достижения вершины из заданной вершины. Значение потенциала постоянно корректируется на каждом шаге алгоритма. Изначально потенциал заданной вершины считается равным нулю, а потенциалы всех остальных вершин считаются равными бесконечности. Это отражает, что расстояния изначально неизвестны. Все вершины, кроме заданной, помечаются как непройденные. На каждом шаге алгоритм выбирает непройденную вершину с наименьшим значением потенциала. Для выбранной вершины алгоритм просматривает её окрестность и выбирает из окрестности те вершины, которые являются ещё непройденными. Для каждой такой вершины алгоритм пересчитывает значение потенциала, используя функцию расстояния, заданную на графе. Для этого он использует значения весов дуг, соединяющих выбранную вершину с вершинами из её окрестности. Если пересчитанное значение потенциала оказывается меньше его текущего значения, то значение потенциала корректируется для данной вершины. После пересчёта потенциала каждой вершины, алгоритм помечает выбранную вершину как пройденную, и выбирает новую вершину, исходя из критерия – минимум значения потенциала. Алгоритм продолжается до тех пор, пока все вершины не будут помечены как пройденные и выбор нового потенциала является возможным.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины *v*, способа хранения множества непройденных вершин и способа обновления потенциалов. В простейшем случае, когда для поиска вершины с минимальным весом просматривается всё множество вершин, время работы алгоритма определяется как $O(P^2+Q)$. Основной цикл выполняется порядка P раз, и в каждом из них на нахождение минимума тратится порядка P операций, плюс количество смен меток, которое не превосходит количества дуг Q в исходном графе. Для разряженных графов, т.е. для которых Q много меньше P×P, непройденные вершины можно хранить в двоичной куче, а в качестве ключа использовать значение веса, тогда время извлечения вершины станет log P, а так как цикл выполняется порядка P раз и количество смен меток не больше Q, то скорость работы алгоритма определяется как $O(P \cdot \log P + Q \cdot \log P)$.

В целом, алгоритм является эффективным и широко используется на практике в различных модификациях.

12 Поиск кратчайших и критических путей

В общем случае, кратчайший путь по количеству дуг и кратчайший путь по сумме весов не совпадают в реальных системах. Например, для графа, представленного на рис. 22, кратчайший путь из истока v_1 в сток v_6 по сумме весов проходит: v_1,v_2,v_3,v_4,v_6 (показан пунктиром), а по количеству дуг – v_1,v_3,v_5,v_6 (показан жирно).

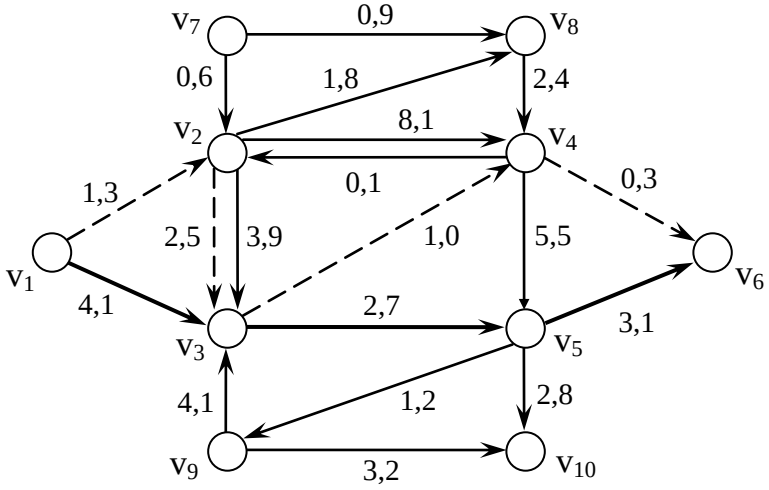


Рисунок 22 – Кратчайшие пути на графе

Для условности, кратчайший (критический) путь по количеству дуг будем называть минимальным (максимальным) путем. Хотя, не трудно видеть, что функция расстояния по количеству дуг является частным случаем функции расстояния по сумме весов, если считать все веса равными единице.

Задача нахождения кратчайшего пути формулируется как задача нахождения такого пути между вершинами v_i и v_j, что $L(v_i \to v_j)=\min$ или $W(v_i \to v_j)=\min$, а для критического пути $L(v_i \to v_j)=\max$ или $W(v_i \to v_j)=\max$. При этом возможны различные постановки задачи: найти кратчайший (критический) путь между двумя вершинами, найти дерево кратчайших путей с корнем в заданной вершине, найти все кратчайшие пути между всеми парами вершин, и т.д.

По числу приложений задача о кратчайшем (критическом) пути занимает первое место среди других задач теории графов. Метод поиска кратчайшего или критического пути основан на последовательном шаг за шагом перемещении пометки по вершинам графа от начальной вершины $\mathbf{v_н}$ до конечной вершины $\mathbf{v_к}$ искомого пути с вычислением на каждом шаге функции расстояния и проверкой вычисленного значения с наилучшим значением w^{eff} по критерию поиска пути. Так критерий для кратчайшего пути $w<w^{eff}$, а для критического $w>w^{eff}$. Вершина **x**, до которой найдено наилучшее расстояние, помечается как пройденная Ins(**x**) и становится текущей. Изначально, текущей становится начальная вершина $\mathbf{x}=\mathbf{v_н}$ искомого пути и расстояние до неё самой принимается равным нулю $w_{vн}=0$, а расстояния до всех остальных вершин графа полагается равным наихудшему значению критерия w^{krit}. Так для кратчайшего пути $w^{krit}=\infty$, а для критического $w^{krit}=0$. На каждом шаге метода осуществляется просмотр выходной окрестности текущей вершины $\mathbf{v}\in\mathbf{\Gamma}^{-}\mathbf{x}$ и вычисление расстояния от текущей вершины до вершин, входящих в её окрестность $w_p=f(x, v, u(x,v))$. В специальное множество **K** помещается тройка: вершина из окрестности **v**, расстояние до неё $\mathbf{w_p}$, дуга, связывающая её с текущей вершиной u=(x,v). При этом, если расстояние до некоторой вершины из окрестности уже было вычислено на предыдущих шагах, то в множестве **K** тройка корректируется с учётом критерия поиска пути. Затем из непомеченных вершин множества **K** выбирается вершина **z**, до которой расстояние является наилучшим с точки зрения критерия поиска пути. Так для

кратчайшего пути $w_z=\min(w_v)$, где $v\in K$ и НЕ(Ins(v)), а для критического пути $w_z=\max(w_v)$, где $v\in K$ и НЕ(Ins(v)). Эта вершина помечается как пройденная Ins(**z**) и делается текущей **x**=**z**. Шаги метода повторяются пока не будет достигнута конечная вершина $\mathbf{v_к}$ искомого пути. Если на очередном шаге оказалось, что в **K** все вершины помечены и добавить больше нечего, то значит пути в конечную вершину не существует.

Пусть дан граф G=(V,U), состоящий из P>3 вершин. На графе G заданы начальная $v_н$ и конечная $v_к$ вершины искомого пути, причем $v_н\neq v_к$. **Алгоритм поиска кратчайшего пути** следующий:

1. Считать все расстояния до вершин наихудшими $W_P(v_i)=\infty$, i=1,...,P. Сделать начальную вершину текущей $x=v_н$ и для неё $W_P(v_н)=0$. Специальное множество пусто K=∅.
2. Пометить текущую вершину как пройденную Ins(x).
3. Рассмотреть окрестность по выходам текущей вершины $\Gamma^{-}x=\{v_1,...,v_j,...,v_\rho\}$. Если окрестность не пуста, то положить j=1, иначе перейти на шаг 8.
4. Взять вершину из окрестности $v_j\in\Gamma^{-}x$. Вычислить расстояние до этой вершины $w_j=W_P(x)$ + min(w($u_l(v_j,x)$), где $l=1,...,L_x$ - количество кратных дуг между v_j и x, т.е. выбирается дуга с минимальным весом.
5. Сравнить по критерию поиска вычисленное значение расстояния со старым значением для этой вершины, т.е. для кратчайшего: $w_j < W_P(v_j)$.
   Если значение улучшается, то положить расстояние до этой вершины равным новому значению, т.е. $W_P(v_j)=w_j$, иначе оставить старое значение.
6. Поместить или скорректировать в множестве K тройку:
$$K = K \cup (v_j, w_j, u_l)$$
7. Проверить, если в окрестности еще есть дуги, т.е. $j < \rho$, то j=j+1 и перейти к п.4, иначе перейти к п.8.
8. Проверить есть ли в множестве K непомеченные вершины, т.е. еще не пройденные вершины. Если нет, то все возможные вершины пройдены и перейти к п.11, иначе перейти к п.9.
9. Выбрать следующую вершину пути и сделать её текущей. Для этого просмотреть все непомеченные вершины в множестве K и выбрать вершину с наилучшим по критерию значением расстояния:

x – такая, что $W_p(x) = \min(W_p(v_i))$, где $v_i \in K$ и НЕ$(Ins(v_i))$, i=1,...,|K|

10. Если новая текущая вершина x – это заданная конечная вершина пути, т.е. $x=v_к$, то искомый путь найден и перейти к п.11, иначе перейти к п.2.
11. Тройки в множестве K составляют поддерево кратчайших путей из вершины $v_н$. Если в множестве K есть тройка, содержащая вершину $v_к$, то из троек можно построить кратчайший путь из $v_н$ в $v_к$, иначе между заданными вершинами пути не существует.

Алгоритм поиска критического пути отличается от этого алгоритма пунктом 1, в котором наихудшими расстояниями являются $W_p(v_i)=0$; пунктом 5, в котором используется критерий $w_j > W_p(v_j)$; пунктом 9, в котором вершина выбирается по критерию $W_p(x) = \max(W_p(v_i))$.

Рассмотрим работу алгоритма на примере графа, представленного на рис. 22. Требуется найти кратчайший путь по сумме весов из вершины v_1 в вершину v_6. Для условности, пройденные помеченные вершины будут обозначаться знаком ‘+’, а следующая текущая вершина, на которую переместится пометка, будет обозначаться знаком ‘*’. Кроме того, рядом с вершиной в прямоугольнике будет указываться значение расстояния до этой вершины. Окрестность текущей вершины будет выделяться пунктиром на графе. Результат каждого шага продемонстрируем на соответствующем рисунке.

Подготовка. Полагаем, что расстояние до всех вершин графа равно наихудшему значению, т.е. $W_p(v_i)=\infty$, i=1,...,10. В качестве текущей выбираем начальную вершину искомого пути $x=v_1$. До начальной вершины расстояние нулевое $W_p(v_1)=0$. В множество K помещаем тройку (v_1, 0.0, ∅). Выполним шаги работы алгоритма.

шаг 1. $x=v_1$. Помечаем текущую вершину как пройденную $Ins(v_1)$.

Берём её окрестность по выходам $\Gamma^{-}v_1=\{v_2,v_3\}$.

Ей соответствуют дуги с весами: $w(v_1,v_2)=1.3$ и $w(v_1,v_3)=4.1$.

Вычисляем расстояние W_p до вершин v_2, v_3 и сравниваем его по критерию поиска, корректируя множество K:

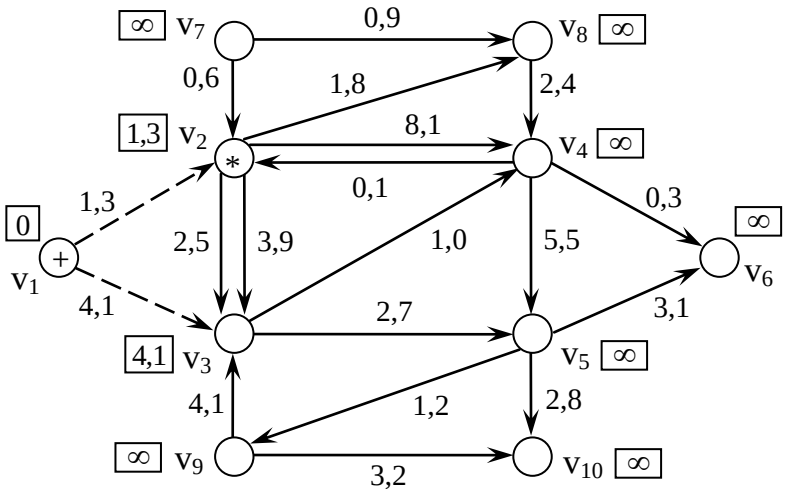
$w_{v2}=W_p(v_1) + w(v_1,v_2)=0.0+1.3=1.3 < W_p(v_2) =\infty \Rightarrow$ да $\Rightarrow W_p(v_2)=w_{v2}=1.3$

Добавляем в K тройку (v_2, 1.3, (v_1,v_2,1.3)).

$w_{v3}=W_p(v_1) + w(v_1,v_3)=0.0+4.1=4.1 < W_p(v_3) =\infty \Rightarrow$ да $\Rightarrow W_p(v_3)=w_{v3}=4.1$

Добавляем в К тройку $(v_3, 4.1, (v_1, v_2, 4.1))$.

Результат первого шага работы алгоритма – граф и множество К представлены на рис. 23.



множество К					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
*	v_2	1.3	v_1	v_2	1.3
	v_3	4.1	v_1	v_3	4.1

Рисунок 23 – Результат 1-ого шага поиска пути

Выбираем новую текущую вершину. Для этого просматриваем в множестве К непомеченные вершины и выбираем вершину с наименьшим расстоянием: $\min(W_p(v_2), W_p(v_3))=\min(1.3,4.1)=1.3$, что соответствует вершине v_2 – делаем её текущей.

шаг 2. $x=v_2$. Помечаем текущую вершину как пройденную $Ins(v_2)$.

Берём её окрестность по выходам $\Gamma^- v_2=\{v_3,v_4,v_8\}$.

Ей соответствуют дуги с весами: $w(v_2,v_4)=8.1$, $w(v_2,v_8)=1.8$,

$w_1(v_2,v_3)=3.9$, $w_2(v_2,v_3)=2.5$ – последние две дуги – кратные.

Вычисляем расстояние W_p до вершин v_4, v_8, v_3 и сравниваем его по критерию поиска, корректируя множество К:

$w_{v4}=W_p(v_2) + w(v_2,v_4)=1.3+8.1=9.4 < W_p(v_4) =\infty \Rightarrow$ да $\Rightarrow W_p(v_4)=w_{v4}=9.4$

Добавляем в К тройку $(v_4, 9.4, (v_2,v_4,8.1))$.

$w_{v8}=W_p(v_2) + w(v_2,v_8)=1.3+1.8=3.1 < W_p(v_8) =\infty \Rightarrow$ да $\Rightarrow W_p(v_8)=w_{v3}=3.1$

Добавляем в К тройку $(v_8, 3.1, (v_2,v_8,1.8))$.

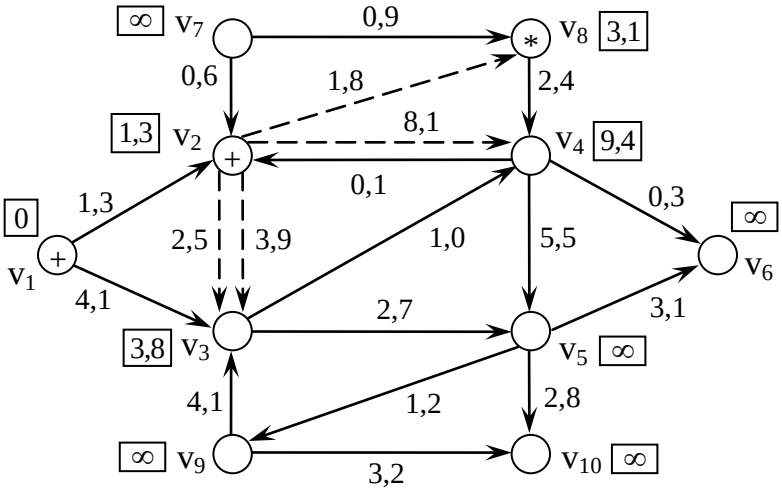
А теперь разберёмся с кратными дугами в вершину v_3.

$w_{v3}=W_p(v_2) + \min(w_1(v_2,v_4), w_2(v_2,v_8))=1.3+\min(3.9, 2.5)=1.3+2.5=3.8$

Получаем: $w_{v3}=3.8 < W_p(v_3)=4.1 \Rightarrow$ да $\Rightarrow W_p(v_4)=w_{v4}=3.8$

Поскольку в множестве K уже есть тройка с вершиной v_3, т.е. (v_3, 4.1, (v_1, v_2, 4.1)), а новое значение расстояния до v_3 лучше прежнего, то заменяем тройку в K на новую тройку (v_3, 3.8, (v_2, v_3, 2.5)).

Результат второго шага представлен на рис. 24.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
	v_3	3.8	v_2	v_3	2.5
	v_4	9.4	v_2	v_4	8.1
*	v_8	3.1	v_2	v_8	1.8

Рисунок 24 – Результат 2-ого шага поиска пути

Выбираем новую текущую вершину из непомеченных вершин множества K: $\min(W_p(v_3), W_p(v_4), W_p(v_8))=\min(4.1, 9.4, 3.1)=3.1$, что соответствует вершине v_8 – делаем её текущей.

шаг 3. $x=v_8$. Помечаем текущую вершину как пройденную $Ins(v_8)$.

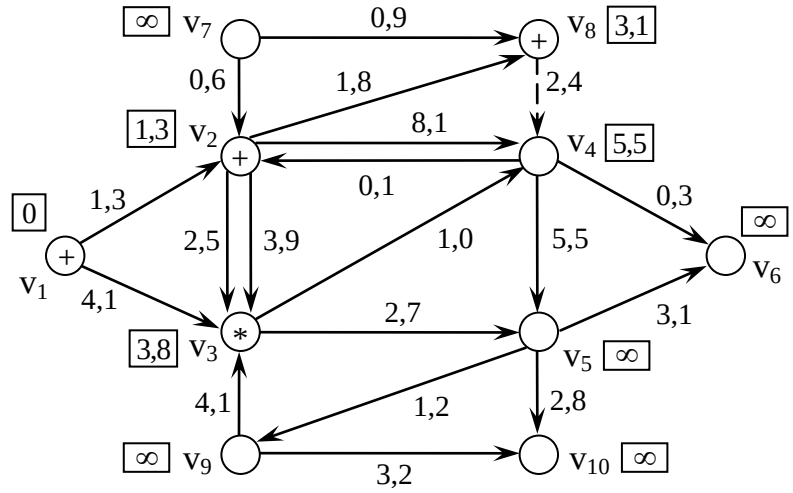
Берём её окрестность по выходам $\Gamma^{-}v_8=\{v_4\}$ с одной дугой $w(v_8,v_4)=2.4$.

Вычисляем расстояние W_p и сравниваем его по критерию поиска:

$w_{v4}=W_p(v_8) + w(v_8,v_4)=3.1+2.4=5.5 < W_p(v_4)=9.4 \Rightarrow$ да $\Rightarrow W_p(v_4)=w_{v4}=5.5$

Заменяем тройку в K на улучшенную тройку (v_4, 5.5, (v_8, v_4, 2.4)).

Результат третьего шага представлен на рис. 25.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
*	v_3	3.8	v_2	v_3	2.5
	v_4	5.5	v_8	v_4	2.4
+	v_8	3.1	v_2	v_8	1.8

Рисунок 25 – Результат 3-его шага поиска

Выбираем новую текущую вершину из непомеченных вершин множества K с наименьшим значением расстояния: $\min(W_p(v_3),W_p(v_4))=$ $=\min(3.8, 5.5)=3.8$, что соответствует вершине v_3 – делаем её текущей. Таким образом, на данном шаге происходит возврат к ранее пройденному пути и выбор другого направления продвижения, поскольку продвижение к вершине v_6 данным путём не улучшает функцию расстояния.

шаг 4. $x=v_3$. Помечаем текущую вершину как пройденную $\text{Ins}(v_3)$.

Берём её окрестность по выходам $\Gamma^- v_3=\{v_4,v_5\}$.

Ей соответствуют дуги с весами: $w(v_3,v_4)=1.0$, $w(v_3,v_5)=2.7$.

Вычисляем расстояние W_p до вершин v_4, v_5 и сравниваем его по критерию поиска, корректируя множество K:

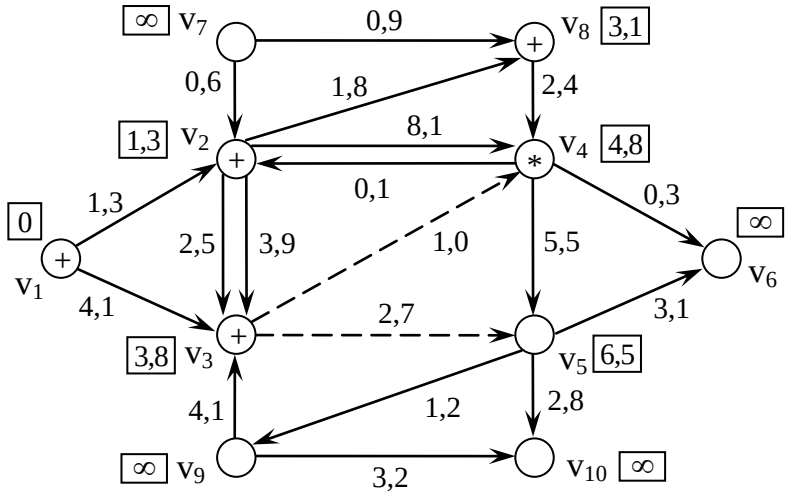
$w_{v4}=W_p(v_3) + w(v_3,v_4)=3.8+1.0=4.8 < W_p(v_4)=5.5 \Rightarrow$ да $\Rightarrow W_p(v_4)=w_{v4}=4.8$

Снова заменяем тройку с вершиной v_4 в множестве K на улучшенную тройку (v_4, 4.8, (v_3, v_4, 1.0)).

$w_{v5}=W_p(v_3) + w(v_3,v_5)=3.8+2.7=6.5 < W_p(v_5)=\infty \Rightarrow$ да $\Rightarrow W_p(v_5)=w_{v5}=6.5$

Добавляем в K тройку (v_5, 6.5, (v_3, v_5, 2.7)).

Результат четвертого шага представлен на рис. 26.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
+	v_3	3.8	v_2	v_3	2.5
*	v_4	4.8	v_3	v_4	1.0
+	v_8	3.1	v_2	v_8	1.8
	v_5	6.5	v_3	v_5	2.7

Рисунок 26 – Результат 4-ого шага поиска путей

Выбираем новую текущую вершину из непомеченных вершин множества K: $\min(W_p(v_4), W_p(v_5))=\min(4.8, 6.5)=4.8$, что соответствует вершине v_4 – делаем её текущей.

шаг 5. $x=v_4$. Помечаем текущую вершину как пройденную $Ins(v_4)$.

Берём её окрестность по выходам $\Gamma^{-}v_4=\{v_2,v_5,v_6\}$.

Ей соответствуют дуги с весами: $w(v_4,v_2)=0.1, w(v_4,v_5)=5.5, w(v_4,v_6)=0.3$

Учитывая, что вершина v_2 уже пройдена, её отбрасываем из рассмотрения, поскольку возврат к v_2 не улучшит расстояние.

Вычисляем расстояние W_p до вершин v_5, v_6 и сравниваем его по критерию поиска, корректируя множество K:

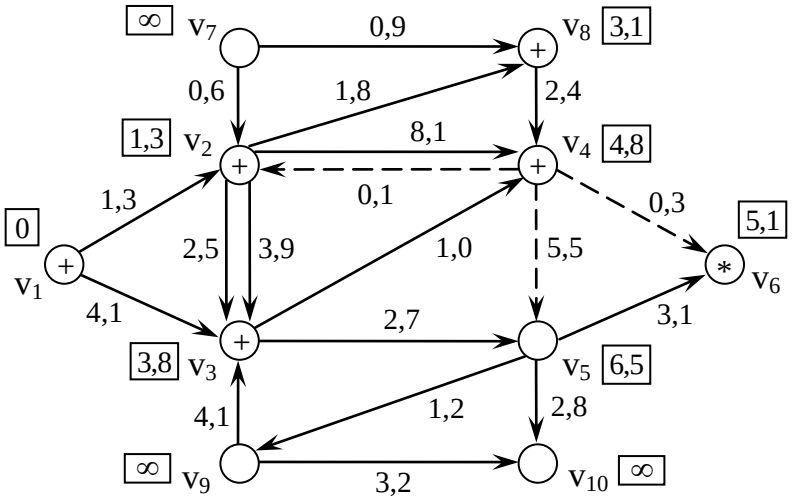
$w_{v5}=W_p(v_4) + w(v_4,v_5)=4.8+5.5=10.3 < W_p(v_5)=6.5 \Rightarrow$ нет

Новое значение не улучшает критерий, значит оставляем тройку с вершиной v_5 в множестве K без изменений.

$w_{v6}=W_p(v_4) + w(v_4,v_6)=4.8+0.3=5.1 < W_p(v_6)=\infty \Rightarrow$ да $\Rightarrow W_p(v_6)=w_{v6}=5.1$

Добавляем в K тройку $(v_6, 5.1, (v_4, v_6, 0.3))$.

Результат пятого шага представлен на рис. 27.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
+	v_3	3.8	v_2	v_3	2.5
+	v_4	4.8	v_3	v_4	1.0
+	v_8	3.1	v_2	v_8	1.8
	v_5	6.5	v_3	v_5	2.7
*	v_6	5.1	v_4	v_6	0.3

Рисунок 27 – Результат 5-ого шага поиска путей

Выбираем новую текущую вершину из непомеченных вершин множества K: $\min(W_p(v_5), W_p(v_6))=\min(6.5, 5.1)=5.1$, что соответствует вершине v_6 – делаем её текущей.

Проверяем, $x=v_6$ – конечная вершина искомого пути – поиск прекращается. Анализируя множество K от конечной вершины можно построить искомый кратчайший путь. Так, по множеству K имеем: из вершины v_6 возвращаемся к вершине v_4, от v_4 возвращаемся к v_3, от v_3 возвращаемся к v_2, от v_2 возвращаемся к v_1 – начальная вершина. Результат поиска кратчайшего пути по сумме весов представлен (жирно) на рис. 28.

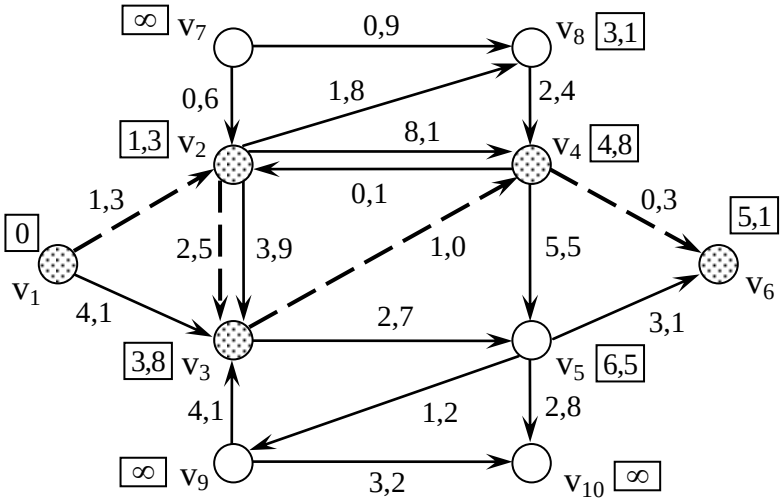


Рисунок 28 – Результат поиска кратчайшего пути

Другой важной задачей анализа систем является нахождение дерева кратчайших (критических) путей с корнем в заданной вершине. В этой задаче нужно найти кратчайшие (критические) пути из заданной вершины до всех остальных вершин графа. По этим путям поток от данного элемента системы распространяется быстрее, чем по другим путям. Далее можно определить максимальное и минимальное значение этих путей и, таким образом, оценить передачу потоков от одного элемента до всех остальных элементов системы.

Если немножко модифицировать алгоритм поиска кратчайших (критических) путей, то его можно использовать для решения задачи построения дерева кратчайших (критических) путей из заданной начальной вершины до всех остальных вершин графа. Для этого нужно исключить из алгоритма п.10 – проверку на конечную вершину. Тогда алгоритм будет работать до тех пор, пока не обойдет все вершины, к которым существуют пути из заданной начальной вершины и вычислит до них кратчайшие (критические) расстояния. У вершин, до которых не существует пути, значение расстояния останется $W_P(v)=\infty$ (или $W_P(v)=0$).

Например, для графа, представленного на рис. 28 рассмотрим построение дерева кратчайших путей с корнем в вершине v_1. Большая часть работы уже сделана при поиске кратчайшего пути из вершины v_1 в вершину v_6, в частности, была найдена ветка из v_2 в v_8. Просто исключим п.10 из алгоритма и продолжим выкладки предыдущего примера, ведь алгоритм ещё не обошел все вершины, к которым существуют пути из вершины v_1. Текущей

вершиной стала v_6, но её окрестность по выходам пуста, т.е. $\Gamma^{-}v_6=\varnothing$, и следовательно путей из неё нет, значит сразу нужно переходить к выбору следующей текущей вершины, из которой возможно продолжать построение путей от вершины v_1. Итак, помечаем вершину v_6 как пройденную $Ins(v_6)$, и выбираем новую текущую вершину для поиска, но из непомеченных вершин множества K можно выбрать только вершину v_5 – делаем её текущей. Продолжаем шаги:

шаг 6. $x=v_5$. Помечаем текущую вершину как пройденную $Ins(v_5)$.

Берём её окрестность по выходам $\Gamma^{-}v_5=\{v_6,v_9,v_{10}\}$.

Ей соответствуют дуги с весами: $w(v_5,v_6)=3.1, w(v_5,v_9)=1.2, w(v_5,v_{10})=2.8$

Вычисляем расстояние W_p до вершин v_6, v_9, v_{10} и сравниваем его по критерию поиска, корректируя множество K:

$w_{v6}=W_p(v_5) + w(v_5,v_6)=6.5+3.1=9.6 < W_p(v_6)=5.1 \Rightarrow$нет

Новое значение не улучшает критерий, что очевидно, поскольку вершина v_6 уже пройдена.

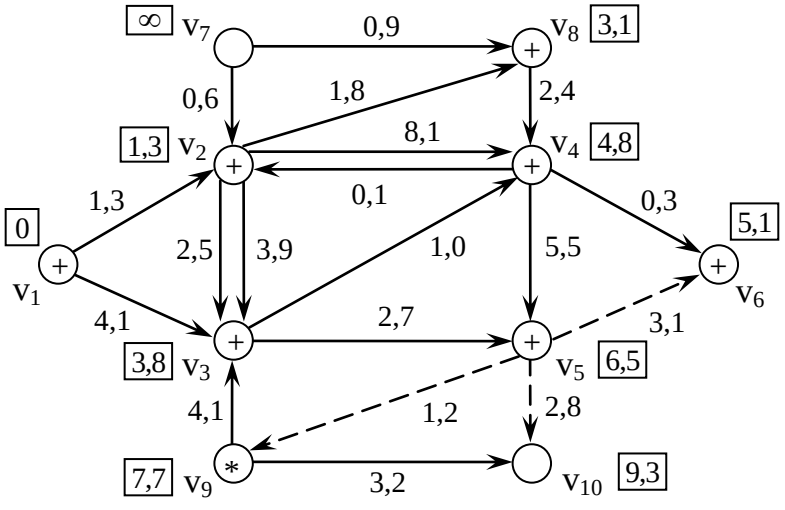
$w_{v9}=W_p(v_5) + w(v_5,v_9)=6.5+1.2=7.7 < W_p(v_9)=\infty \Rightarrow$да$\Rightarrow$ $W_p(v_9)=w_{v9}=7.7$

Добавляем в K тройку $(v_9, 7.7, (v_5, v_9, 1.2))$.

$w_{v10}=W_p(v_5)+w(v_5,v_{10})=6.5+2.8=9.3<W_p(v_{10})=\infty \Rightarrow$да$\Rightarrow$ $W_p(v_{10})=w_{v10}=9.3$

Добавляем в K тройку $(v_{10}, 9.3, (v_5, v_{10}, 2.8))$.

Результат шестого шага представлен на рис. 29.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
+	v_3	3.8	v_2	v_3	2.5
+	v_4	4.8	v_3	v_4	1.0
+	v_8	3.1	v_2	v_8	1.8
+	v_5	6.5	v_3	v_5	2.7
+	v_6	5.1	v_4	v_6	0.3
*	v_9	7.7	v_5	v_9	1.2
	v_{10}	9.3	v_5	v_{10}	2.8

Рисунок 29 – Результат 6-ого шага поиска путей

Выбираем новую текущую вершину из непомеченных вершин множества K: $\min(W_p(v_9), W_p(v_{10}))=\min(7.7, 9.3)=7.7$, что соответствует вершине v_9 – делаем её текущей.

шаг 7. $x=v_9$. Помечаем текущую вершину как пройденную $Ins(v_9)$.

Берём её окрестность по выходам $\Gamma^- v_9=\{v_3,v_{10}\}$.

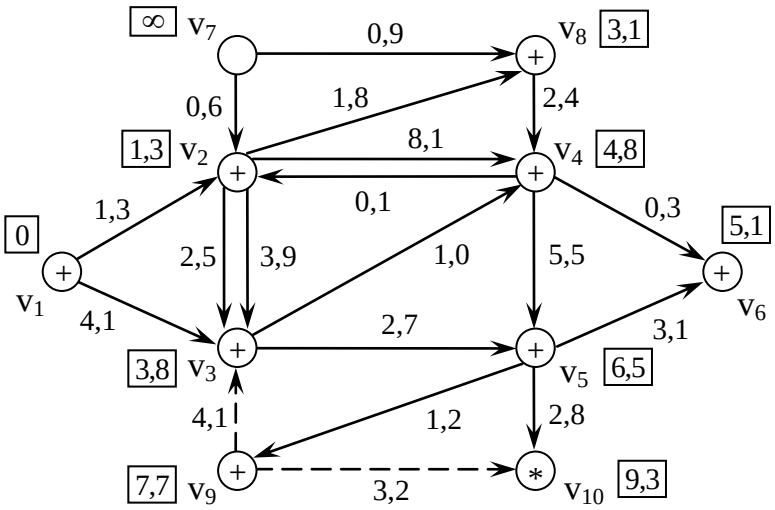
Ей соответствуют дуги с весами: $w(v_9,v_3)=4.1, w(v_9,v_{10})=3.2$

Вершина v_3 уже пройдена, поэтому отбрасываем её. Вычисляем расстояние W_p до вершины v_{10}:

$w_{v10}=W_p(v_9) + w(v_9,v_{10})=7.7+3.2=10.9 < W_p(v_{10})=9.3 \Rightarrow$нет

Новое значение не улучшает критерий, значит оставляем тройку с вершиной v_{10} в множестве K без изменений.

Результат седьмого шага представлен на рис. 30.



множество K					
вершина			дуга		
⊕	v	Wp	нач	кон	вес
+	v_1	0.0	∅		
+	v_2	1.3	v_1	v_2	1.3
+	v_3	3.8	v_2	v_3	2.5
+	v_4	4.8	v_3	v_4	1.0
+	v_8	3.1	v_2	v_8	1.8
+	v_5	6.5	v_3	v_5	2.7
+	v_6	5.1	v_4	v_6	0.3
+	v_9	7.7	v_5	v_9	1.2
*	v_{10}	9.3	v_5	v_{10}	2.8

Рисунок 30 – Результат 7-го шага поиска путей

Выбираем новую текущую вершину. Из непомеченных вершин множества K осталась только v_{10} – делаем её текущей.

шаг 8. $x=v_{10}$. Помечаем текущую вершину как пройденную $Ins(v_{10})$.

Берём её окрестность по выходам $\Gamma^- v_{10}=\{\}$.

Выбрать новую текущую вершину из непомеченных вершин множества K невозможно, поскольку все вершины в нём помечены.

Таким образом, согласно п.8 алгоритма, все вершины, до которых существуют пути из заданной вершины v_1, пройдены – **алгоритм останавливается.**

В результате работы алгоритма в множестве K сформировалось дерево кратчайших путей. Столбец W_p показывает кратчайшие расстояния до соответствующей вершины графа из заданной вершины. Как видно, до вершины v_7 расстояние осталось равным наихудшему значению $W_p=\infty$, так как пути до неё не существует. Окончательный результат – дерево кратчайших путей – представлены на рис. 31.

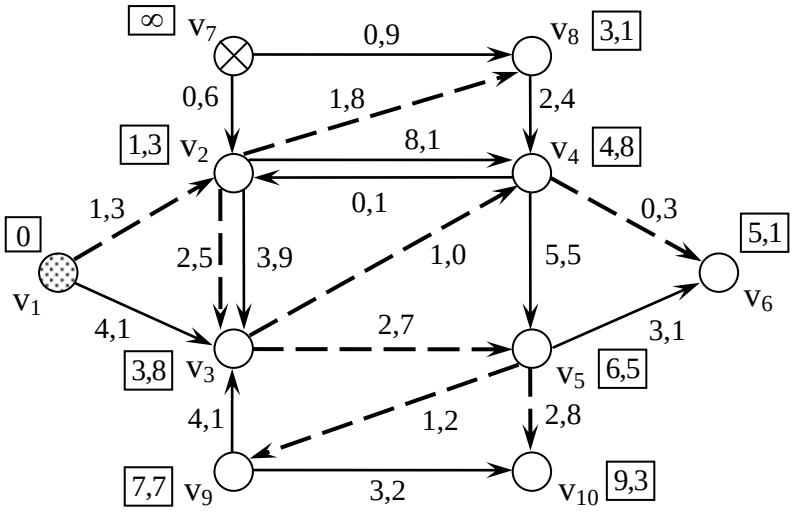


Рисунок 31 – Результат построения дерева кратчайших путей

Задача нахождения всех кратчайших путей от всех вершин графа возникает при оценке метрических свойств графа и вычисления обобщенных характеристик. Для решения задачи нахождения всех кратчайших путей от всех вершин нужно применять описанный выше алгоритм нахождения дерева кратчайших путей к каждой вершине графа и запоминать вектор расстояний W_p для каждой вершины v_i, i=1,..,P. В результате получается квадратная матрица размерностью P×P, в (*i*,*j*)-ячейке которой находится значение кратчайшего расстояния $w^{кр}$ из вершины *i* в вершину *j*, которая называется ***матрицей кратчайших расстояний***. По ней можно оценить минимальное и максимальное значение расстояния между любой парой вершин.

Все указанные задачи применяются для выявления особенностей соединения элементов в системе, правильного выбора методов анализа и составления математических моделей системы на последующих этапах проектирования.

Для графа на рис. 31 матрица кратчайших расстояний имеет вид:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	min	max
v_1	0	1,3	3,8	4,8	6,5	5,1	∞	3,1	7,7	9,3	1,3	9,3
v_2	∞	0	2,5	3,5	5,2	3,8	∞	1,8	6,4	8,0	1,8	8,0
v_3	∞	1,1	0	1,0	2,7	1,3	∞	2,9	3,9	5,5	1,0	5,5
v_4	∞	0,1	3,6	0	5,5	0,3	∞	1,9	6,7	8,3	0,1	8,3
v_5	∞	6,4	5,3	6,3	0	3,1	∞	8,2	1,2	2,8	1,2	8,2
v_6	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	сток	
v_7	∞	0,6	3,1	3,3	5,8	3,6	0	0,9	7,0	8,6	0,6	8,6
v_8	∞	2,5	5,0	2,4	7,7	2,7	∞	0	8,9	10,5	2,4	10,5
v_9	∞	5,2	4,1	5,1	6,8	5,4	∞	7,0	0	3,2	3,2	7,0
v_{10}	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	сток	

13 Обобщенные характеристики графа

При решении задач анализа потоков результаты удобно представлять некоторыми обобщенными характеристиками, которыми являются радиус и диаметр графа. Прежде чем дать определение этим характеристикам необходимо ввести понятие эксцентриситета вершины.

Пусть G=(V,U) – связанный граф, P – количество вершин графа.

Эксцентриситет вершины – это длина максимального из наикратчайших путей от данной вершины до всех остальных вершин графа, т. е.

$$exc(v) = \max L(v \rightarrow v_i)\ ,\ \ i=1,...,P \qquad (18)$$

Разные вершины в графе имеют разный эксцентриситет, поэтому интересно знать верхнее и нижнее значение эксцентриситетов вершин.

Радиусом графа называется минимальный из эксценриситетов его вершин, а ***диаметром графа*** называется максимальный из эксцентриситетов его вершин, т. е.

$$R(G) = \min exc(v_i) \quad \text{и} \quad D(G) = \max exc(v_i), \quad i=1,...,P \qquad (19)$$

Диаметр графа дает верхнюю оценку длин путей. Радиус графа дает нижнюю оценку длин путей в графе. Однако, если R(G)=0, то это означает, что граф содержит вершины-тупики (стоки).

Вершины, эксцентриситет которых равен радиусу или диаметру, может быть несколько. Поэтому, после вычисления эксцентриситетов всех вершин графа, в нём можно выделить три подмножества вершин: эксцентриситет

равен радиусу (центральные вершины), эксцентриситет равен диаметру (периферийные вершины), и находится между радиусом и диаметром (обычные вершины).

Центр графа – это подмножество вершин графа, эксцентриситет которых равен радиусу, т.е.

$$\forall v \in V^{ц} \{v \in V \mid exc(v) = R(G)\} \qquad (20)$$

Это множество элементов системы, от которых поток до самого удаленного элемента распространяется быстрее, чем от других элементов системы. Эти элементы находятся как бы в центре системы.

Периферия графа – это подмножество вершин графа, эксцентриситет которых равен диаметру, т.е.

$$\forall v \in V^{п} \{v \in V \mid exc(v) = D(G)\} \qquad (21)$$

Это множество элементов системы, от которых поток до самого удаленного элемента распространяется дольше всего, хотя до некоторых ближних элементов он может доходить достаточно быстро.

Для графа системы, представленного на рис.32 имеем: $exc(v_1)=exc(v_3)=3$, $exc(v_2)=exc(v_4)=exc(v_5)=exc(v_6)=4$, $exc(v_7)=exc(v_8)=5$, далее получаем $R(G)=3$, $D(G)=5$. В центр графа входят вершины v_1 и v_3, а периферию – вершины v_7 и v_8.

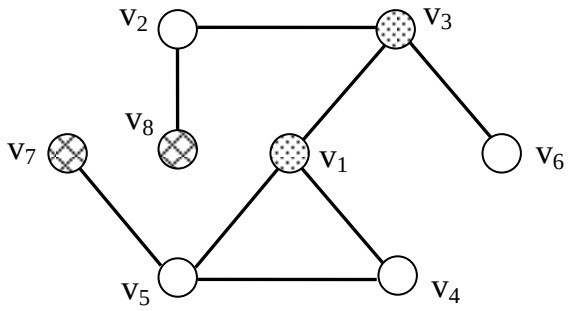


Рисунок 32 – Обобщенные характеристики графа

Во многих задачах требуется искать кратчайшие пути и вычислять обобщенные характеристики не для всех элементов системы, а только для определенного их подмножества, например, входов и выходов системы. Если в системе требуется увеличить количество периферийных или центральных вершин, то это можно сделать, если добавить в граф вершины и симметрично соединить их с остальными вершинами.

В задачах анализа систем может потребоваться оценка насколько больше содержится связей между элементами системы от минимально необходимого числа связей, особенно если на создание каждой связи нужно тратить значительные средства. Для этого удобно использовать такую обобщенную характеристику, как индекс избыточности графа по связям.

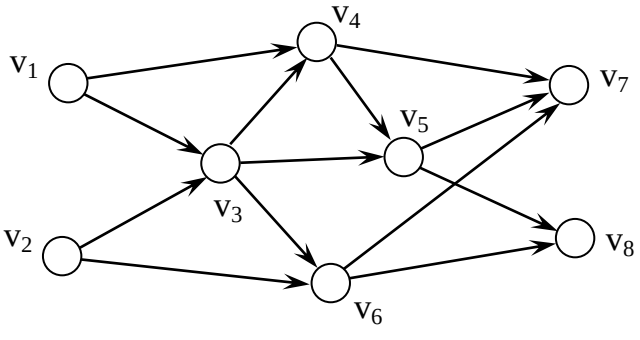
Индекс избыточности – это относительная разность числа связей **Q**, имеющихся в данной структуре, и числа связей $\mathbf{Q_{min}}$, необходимого для того, чтобы граф был связанным. Индекс избыточности используется для оценки связанности графов и интерпретируется как мера избыточности структуры системы по связям. Если граф содержит **P** вершин, то чтобы граф был связанным, необходимо $\mathbf{Q_{min} = P–1}$ дуг, независимо от того, является ли граф ориентированным или нет. Следовательно, для индекса избыточности α имеем формулу:

$$\alpha = (Q - Q_{min}) / Q_{min} = (Q - (P - 1)) / (P - 1) = (Q / (P - 1)) - 1 \qquad (22)$$

Если граф дополнить любым числом изолированных вершин, то значение α останется прежним. Значение **Q** вычисляется по матрице смежности для ориентированного и неориентированного графов по следующим формулам:

$$Q = \sum_{i=1}^{P} \sum_{j=1}^{P} v_{ij} = \sum_{i=1}^{P} v_i \qquad Q = \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} v_{ij} = \frac{1}{2} \sum_{i=1}^{P} v_i \qquad (23)$$

Для графа, представленного на рис. 33, имеем: P=8, Q=13 и получается α=(2+2+3+2+2+2)/(8–1)–1=13/7–1=0,857.



	1	2	3	4	5	6	7	8	
1	0	0	1	1	0	0	0	0	2
2	0	0	1	0	0	1	0	0	2
3	0	0	0	1	1	1	0	0	3
4	0	0	0	0	1	0	1	0	2
5	0	0	0	0	0	0	1	1	2
6	0	0	0	0	0	0	1	1	2
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
	0	0	2	2	2	2	3	2	

Рисунок 33 – Пример графа для расчёта индекса избыточности и сложности

Во многих задачах анализа важно определить как рассматривать реальную систему – относится она ближе к центральной или ближе к распределенной. Для этого удобно использовать такую обобщенную характеристику, как степень централизации структуры.

Степень централизации графа характеризует степень неравномерности загрузки элементов системы и вычисляется по формулам:

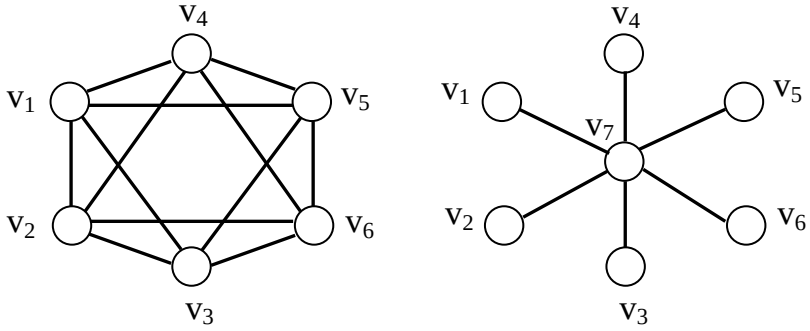
$$\beta = (P - 1)\,(2 \cdot Z_{max} - P) / Z_{max} \cdot (P - 2)$$

$$Z_{max} = \max_{i} \{\, 0.5 \sum_{i=1,P} \sum_{j=1,P} d_{ij} \;\; (\sum_{j=1,P} d_{ij}) - 1 \,\} \qquad (24)$$

где d_{ij} – это длина кратчайшего пути из i-ой вершины в j-ую вершину в дугах, при этом, если i=j, то $d_{ij}=0$;

P – количество вершин.

Если связи в структуре распределены равномерно рис. 34-а, то все вершины инцидентны одному и тому же числу ребер и β=0. Если все вершины связаны с одной единственной – центральной рис. 34-б, то β=1. В реальной системе 0<β<1.



а) Распределенная б) Центральная

Рисунок 34 – Примеры графов для расчёта степени централизации

Для ориентированного графа индекс центральности β вычисляется по формулам:

$$\beta = \frac{1}{(P-1)\,(\rho_k^{\,max} - 1)} \sum_{i=1,i\neq k}^{P} (\rho_k^{\,max} - \rho_i)\,, \qquad (25)$$

где ρ_i – степень i-ой вершины, т.е. суммарное число входящих и исходящих дуг, $\rho_k^{\,max} = \max_{i=1,P} (\rho_i)$ – это максимальная из степеней всех вершин.

Для графа системы, представленного на рис. 33, имеем:

$\rho_1 = \rho_2 = \rho_8 = 2$, $\rho_3 = 5$, $\rho_4 = \rho_5 = \rho_6 = 4$, $\rho_7 = 3$, следовательно $\rho_k^{max} = \rho_3 = 5$, подставляя в формулу, получаем:

$\beta = ((5-2)+(5-2)+(5-4)+(5-4)+(5-4)+(5-3)+(5-2)) / ((8-1)(5-1)) =$

$= (3+3+1+1+1+2+3) / (7*4) = 14 / 28 = 0.5$

Этого следовало ожидать, ведь в данной структуре связи между элементами хотя и распределены неравномерно, но ярко выраженные центральные элементы отсутствуют.

В задачах анализа систем важно дать количественную оценку сложности структуры. Понятие сложности с трудом поддается формализации. Если функционирование системы представить себе как процесс обработки входных потоков в выходные потоки, направленный соответственно от входных элементов системы к выходным элементам, то можно предположить, что изучать свойства этой системы будет тем труднее, чем разнообразнее пути, ведущие от входов к выходам системы. Это предположение берётся за основу обобщенного показателя сложности графа.

Сложность графа определяется по формуле:

$$\mu = \frac{1}{m^{и}\, m^{с}} \sum_{i=1}^{m^{и}} \sum_{j=1}^{m^{с}} n_{ij} - 1 \qquad (26)$$

где $m^{и}$ и $m^{с}$ – количество вершин истоков и стоков в графе ;

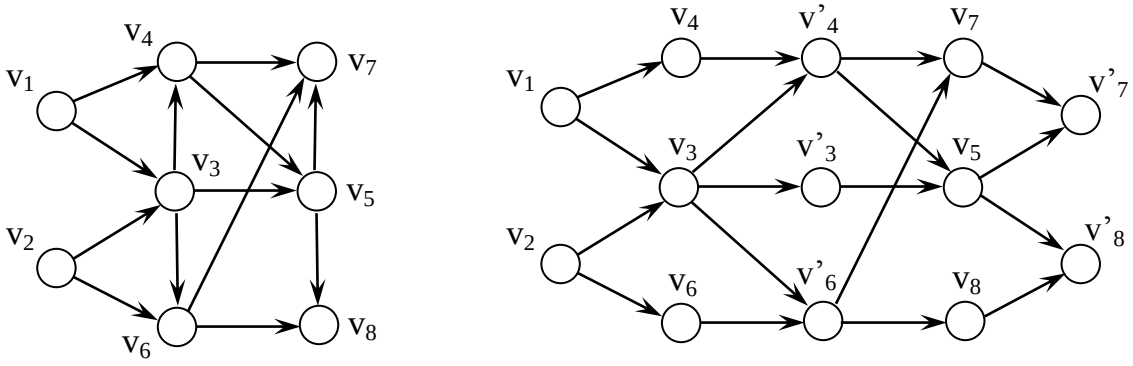
n_{ij} – количество различных путей, ведущих из i-ой вершины-истока в j-ю вершину-сток.

Как видно из формулы величина μ численно равна уменьшенному на единицу среднему числу путей, ведущих из вершины-истока графа в вершину-сток, т.е. от входа к выходу системы. В структуре с минимальной сложностью из каждой вершины-истока можно попасть в вершину-сток единственным путем, и следовательно μ=0.

Для сложных графов, содержащих большое число путей, определить сложность структуры непосредственным перечислением путей довольно трудно. Поэтому для вычисления значения μ графа без петель и контуров удобно использовать следующий алгоритм:

1. Исходный граф представляется в виде многоуровневого иерархического графа, в котором уровень иерархии каждой вершины определяется минимальным числом дуг, связывающих её с истоками;
2. Иерархический граф, преобразуется в эквивалентный ему граф, не содержащий смежных вершин, расположенных на одном и том же уровне иерархии ;
3. Полученный в результате граф представляется совокупностью гиперграфов ;
4. Перемножая матрицы инцидентности гиперграфов, получают матрицу $W=||\mu_{ij}||$ размерности $m^{и}\times m^{с}$; суммируя элементы которой вычисляют μ.

Например, вычислим сложность графа, представленного на рис. 33. Прежде всего изобразим его в виде иерархического графа (шаг 1), как показано на рис. 35-а, который все же содержит дуги, связывающие вершины одного уровня. Чтобы избавиться от этих дуг (шаг 2), все инцидентные им вершины дублируются фиктивными вершинами, расположенными на один уровень иерархии выше, как показано на рис. 35-б (фиктивные вершины обозначены номерами со штрихом). Введение фиктивных вершин позволяет заменить связи между вершинами одного уровня эквивалентными им связями между вершинами соседних уровней иерархии. Так, ребро (v_3,v_4) графа заменяется ребром (v_3,v_4'). Аналогично производится замена ребер (v_3,v_6) на (v_3,v_6'), (v_5,v_7) на (v_5,v_7'), (v_5,v_8) на (v_5,v_8').



а) Шаг 1 б) Шаг 2

Рисунок 35 – Пример вычисления сложности графа на рисунке 33

Полученный пятиуровневый иерархический граф описывается четырьмя гиперграфами (шаг 3) с матрицами инцидентности:

$$\boldsymbol{W}_1 = \begin{array}{c|ccc} & 4 & 3 & 6 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{array} \quad \boldsymbol{W}_2 = \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 4 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \end{array} \quad \boldsymbol{W}_3 = \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 4' & 1 & 1 & 0 \\ 3' & 0 & 1 & 0 \\ 6' & 1 & 0 & 1 \end{array} \quad \boldsymbol{W}_4 = \begin{array}{c|cc} & 7' & 8' \\ \hline 7 & 1 & 0 \\ 5 & 1 & 1 \\ 8 & 0 & 1 \end{array}$$

Перемножим полученные матрицы (шаг 4):

$$\boldsymbol{W} = \boldsymbol{W}_1 \boldsymbol{W}_2 \boldsymbol{W}_3 \boldsymbol{W}_4 = \begin{array}{c|ccc} & 4 & 3 & 6 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{array} \bullet \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 4 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \end{array} \bullet \boldsymbol{W}_3 \boldsymbol{W}_4 =$$

$$= \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 1 & 2 & 1 & 1 \\ 2 & 1 & 1 & 2 \end{array} \bullet \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 4' & 1 & 1 & 0 \\ 3' & 0 & 1 & 0 \\ 6' & 1 & 0 & 1 \end{array} \bullet \boldsymbol{W}_4 = \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 1 & 3 & 3 & 1 \\ 2 & 3 & 2 & 2 \end{array} \bullet \begin{array}{c|cc} & 7' & 8' \\ \hline 7 & 1 & 0 \\ 5 & 1 & 1 \\ 8 & 0 & 1 \end{array} = \begin{array}{c|cc} & 7' & 8 \\ \hline 1 & 6 & 4 \\ 2 & 5 & 4 \end{array}$$

Для полученного графа имеем: n_{17}=6, n_{18}=4, n_{27}=5, n_{28}=4, а поскольку $m^{и}$=$m^{с}$=2, то получаем: μ = (6+4+5+4) / (2*2) – 1 = 3,75.

14 Независимое, критическое, полное подмножество

Висячая вершина – это вершина, которой инцидентна только одна дуга. Причём, необязательно входная или выходная дуга, важно что она единственная у этой вершины. Очевидно, что висячая вершина является либо истоком, либо стоком в графе системы. Дуга, инцидентная висячей вершине, называется ***висячей дугой***.

С точки зрения анализа систем, поскольку висячая вершина связана с остальными вершинами графа единственной связью, то разрыв этой связи приводит к отсоединению элемента от системы. Таким образом, надежность системы зависит не от надежности элемента, обозначенного висячей вершиной, а от надежности единственной связи. Поэтому проектировщик должен позаботится о дополнительных связях, повышающих надежность системы, или о повышении надежности самой связи.

Метод нахождения висячих вершин состоит в вычислении полустепеней захода ρ^+ и исхода ρ^- для всех вершин графа и проверке, чтобы одна из полустепеней была равна нулю, а другая – единице, т.е. ρ^+=0 и ρ^-=1 или ρ^+=1 и ρ^-=0.

Для графа системы, представленного на рис. 36, висячими вершинами будут v_6, v_7, v_9, v_{12}, а висячими дугами (v_3,v_6), (v_7,v_5), (v_{10},v_9), (v_{12},v_4). Видно, что v_6 и v_9 являются стоком, а v_7 и v_{12} – истоком.

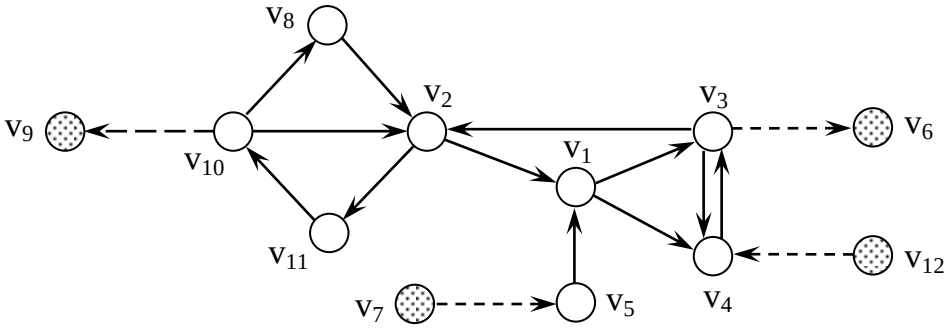


Рисунок 36 – Пример графа с висячими вершинами

Точка сочленения – это вершина, удаление которой из графа увеличивает число изолированных компонент графа. При удалении точки сочленения из графа нарушается достижимость отдельных вершин графа и в системе появляются обрывы цепей. Система как бы распадается на несколько несвязанных между собой подсистем, что приводит к появлению изолированных подграфов. Большое количество точек сочленения ухудшает надежность системы, поскольку при выходе из строя одного из таких элементов, система может перестать нормально функционировать. Точки сочленения показаны на рис. 37.

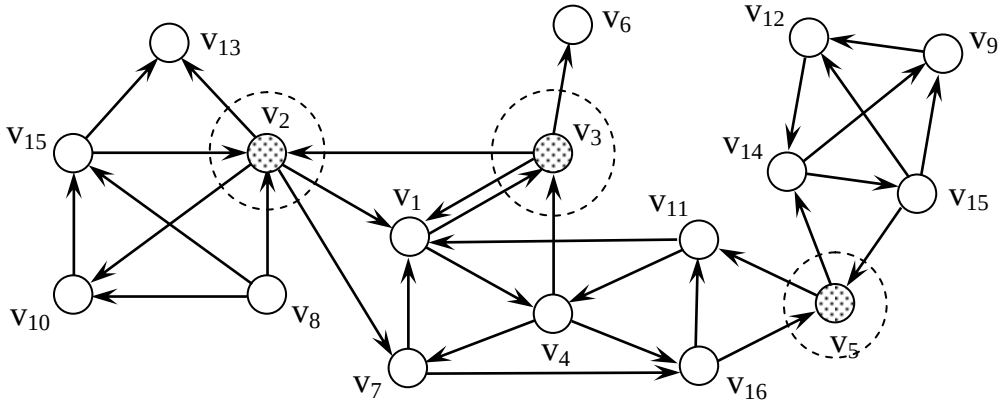


Рисунок 37 – Пример графа с точками сочленения

Точек сочленения в графе может быть несколько и они образуют **критическое множество $T^{кр}$**. Критическое множество содержит элементы, при удалении которых система перестанет существовать как единое целое. В это множество обычно входят вершины-перешейки и вершины, смежные с висячими вершинами.

Для графа, представленного на рис. 37, точками сочленения будут вершины v_2, v_3, v_5. Удаление любой из этих вершин нарушит связанность между остальными вершинами графа. Так, вершина v_3 смежна с висячей вершиной и её удаление приводит к образованию изолированной вершин v_6. Вершины v_2 и v_5 являются перешейками и их удаление приводит к образованию двух изолированных компонент $Z_1=\{v_8, v_{10}, v_{13}, v_{15}\}$ для v_2 и $Z_2=\{v_9, v_{12}, v_{14}, v_{15}\}$ для v_5, которые в результате обрыва цепей станут самостоятельными подсистемами. При этом, никакая другая вершина не нарушает связанности графа. Например, при удалении вершин v_1 или v_4 или v_{16} хотя и нарушается достижимость отдельных вершин, из-за разрушения некоторых путей, но все же граф системы остается связанным. Таким образом, критическое множество $T^{кр}=\{v_2, v_3, v_5\}$ и его мощность $|T^{кр}|=3$.

Для характеристики связанности графа структуры при анализе на обрыв цепей используют число вершинной и реберной связанности. ***Число вершинной связанности*** – это наименьшее число вершин, удаление которых приводит к несвязанному или тривиальному графу. ***Число реберной связанности*** – это наименьшее число дуг, удаление которых приводит к несвязанному или тривиальному графу. Например, для графа на рис. 38 число вершинной связанности равно 2 – это вершины v_4 и v_5, а число реберной связанности равно 3, например дуги (v_6,v_4), (v_6,v_7), (v_6,v_8) делают вершину v_6 изолированной.

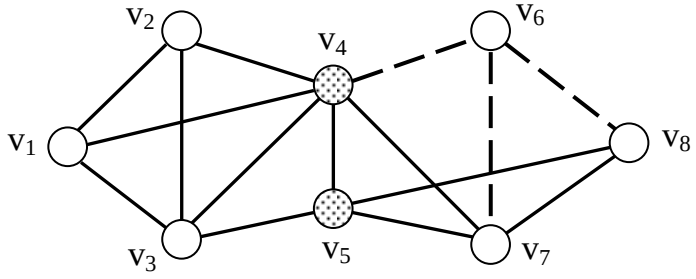


Рисунок 38 – Пример графа с вершинной и реберной связанностью

Независимое множество – это максимальное по числу вершин подмножество вершин графа, таких, что любая пара вершин, входящих в это подмножество, не смежна.

В независимое множество входят элементы системы, которые непосредственно не связаны между собой. Однако это не означает, что

элементы вообще не взаимодействуют друг с другом. Элементы независимого множества взаимодействуют между собой через другие элементы системы. Иногда, независимые множества называют внутренне устойчивыми. Независимые множества различаются по числу входящих в них вершин и в графе их может быть несколько. Независимое множество, содержащее наибольшее число вершин, называется ***предельным***.

В качестве характеристик независимых множеств используются: мощность независимого множества, число внутренней устойчивости, число предельных множеств. ***Мощность независимого множества*** – это число элементов, входящих в данное независимое множество. ***Число внутренней устойчивости*** – это мощность предельного независимого множества. Например, для графа на рис. 39, независимыми множествами будут: $S_1=\{v_3,v_5,v_8\}$, $S_2=\{v_2,v_5,v_7\}$, $S_3=\{v_1,v_3,v_4,v_7\}$, $S_4=\{v_2,v_5,v_8\}$, $S_5=\{v_2,v_6\}$, $S_6=\{v_3,v_4,v_6\}$, $S_7=\{v_3,v_5,v_7\}$, $S_8=\{v_1,v_3,v_8\}$. Для данного графа число независимых множеств равно 8, предельным множеством является S_3, число внутренней устойчивости равно 4.

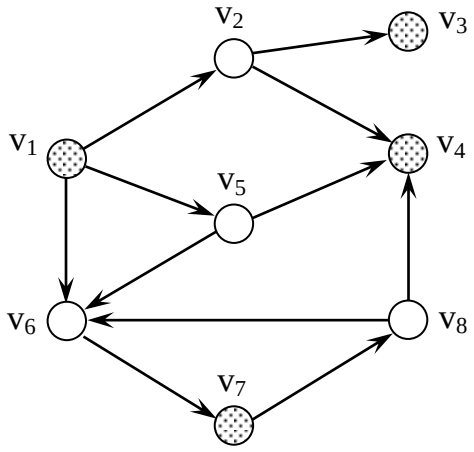


Рисунок 39 – Пример графа с независимым множеством

Противоположностью независимого множества является полное множество. ***Полное множество (или клика)*** – это максимальный по числу вершин подграф, в котором любая пара вершин смежна. Другими словами, в клику входят элементы, которые непосредственно связаны друг с другом.

С одной стороны, ошибочная работа или изменение характеристик одного из элементов клики сразу же сказывается на работе других элементов клики. С другой стороны, выход из строя одного элемента клики не приводит

к обрыву связей между остальными элементами клики. Таким образом, система, имеющая в своем ядре клику, обладает определенной живучестью при отказах отдельных элементов ядра системы. Клики придают системам распределенную структуру. Клики различаются по числу входящих в них вершин и в графе их может быть несколько. Клика, содержащая наибольшее число вершин, называется ***максимальной.***

В качестве характеристик полных множеств используются: мощность клики, число внутренней полноты, число максимальных клик. ***Мощность клики*** – это число вершин, входящих в данную клику. ***Число внутренней полноты*** – это мощность максимальной клики. Например, для графа на рис. 40, клики: $K_1=\{v_5,v_8,v_9\}$-ориентированная мощностью 3 и $K_2=\{v_1,v_3,v_4,v_7\}$-неориентированная мощностью 4, максимальной будет клика K_2, число внутренней полноты равно 4.

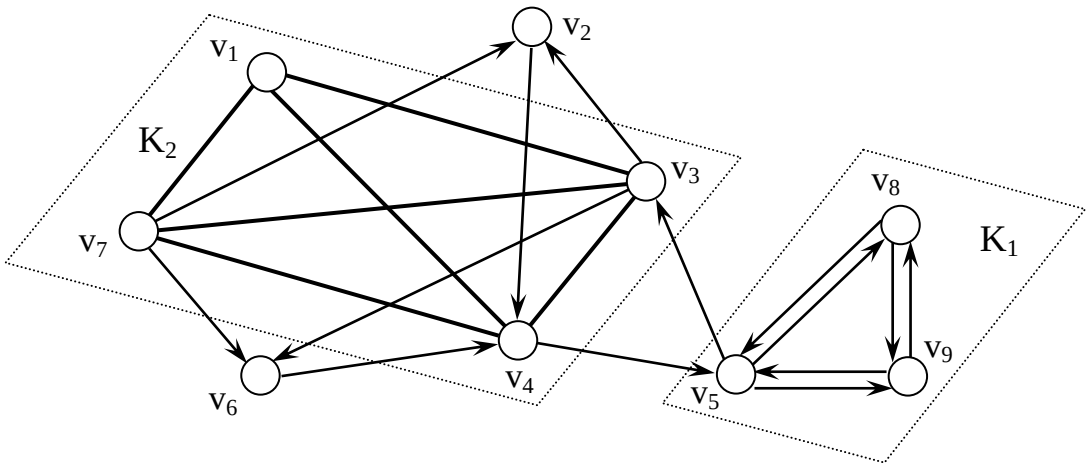


Рисунок 40 – Пример графа с полным множеством (кликой)

Представленные в этом разделе множества дают качественную оценку структуры системы и позволяют определить в каком направлении следует прилагать усилия по обеспечению требуемых структурных показателей системы на последующих стадиях проектирования.

15 Свертывание графа

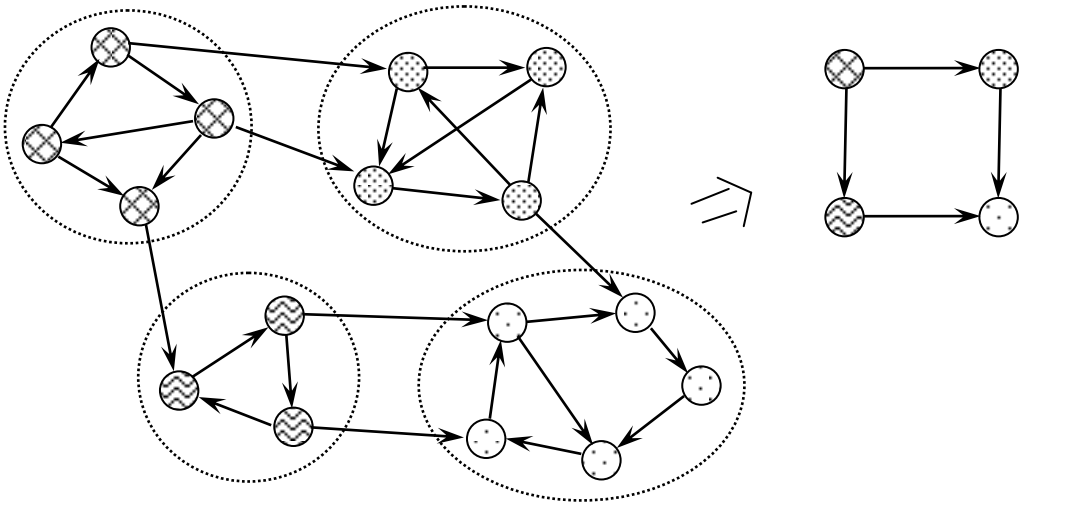
При анализе систем часто возникает необходимость перейти от графа элементов системы к графу подсистем системы, т.е. объединить некоторые элементы системы по каким-либо признакам или свойствам в подсистемы и исследовать взаимосвязь подсистем. Это задача свертки графа.

Пусть дан граф G=(V,U), и P – количество вершин, Q – количество дуг. Пусть каждой вершине $v \in V$ сопоставлено **T** свойств и каждое свойство либо проявляется у элемента, обозначаемого вершиной, либо нет, а также задано, какие значения свойств считаются важными, а какие несущественными, т.е. определена маска свойств.

Требуется перейти от графа G к графу G'=(V',U'), |V'|=P', |U'|=Q', в котором каждая вершина получается объединением нескольких вершин исходного графа G в вершину $v' \in V'$ нового графа и имеющих одинаковое сочетание свойств $t \in T$, а каждая связь $u' \in U'$ получается объединением дуг исходного графа G таким образом, что если две вершины были смежны в исходном графе и они объединились в разные вершины нового графа G', то эти две вершины нового графа будут также смежны, а если объединились в одну вершину нового графа, то над этой вершиной будет петля.

С помощью свертывания графа можно переходить от исследования взаимосвязей элементов системы к исследованию взаимосвязей подсистем системы. Изменяя наборы глобальных и локальных свойств можно получать новые графы свертки, различные с точки зрения анализа структуры, и рассматривать систему как бы с разных позиций. В качестве свойств для свертки можно использовать структурные свойства графа. Например, выше было рассмотрено разбиение графа на сильные компоненты связанности, которые являются замкнутыми подсистемами. Если вершинам присвоить номер сильной компоненты связанности, к которой они принадлежат, и рассматривать этот номер как свойство вершины для свертки, то в результате свертки получится граф конденсации, отражающий связи между замкнутыми подсистемами.

Например, на рис. 41-а представлен граф системы, состоящей из четырех замкнутых подсистем, если каждую из них представить обобщенной вершиной, а дуги между ними обобщенной дугой, то получится граф свертки, представленный на рис. 41-б:



а) Исходный граф б) Свернутый граф

Рисунок 41 – Пример свертки графа

Свертку можно также провести по любому другому найденому подграфу, например, по порядковым уровням, изолированным компонентам, сильным компонентам, независимым множествам, полным множествам, хроматическим множествам, и т.д.

З А К Л Ю Ч Е Н И Е

Для закрепления теоретического материала учебного пособия, рекомендуется выполнить цикл лабораторных работ с использованием специальной обучающей программы «Graf Toolbox», имеющий свидетельство о государственной регистрации программы для ЭВМ № 2002611658 от 12.11.2002. Данная программа позволяет в интерактивном графическом режиме создавать графы и применять к ним описанные методы анализа графов. Результаты работы методов визуально представляются в интерфейсе программы.

Л И Т Е Р А Т У Р А

1. Кузнецов О.П., Дискретная математика для инженера. / О.П. Кузнецов, Г.М. Адельсон-Вельский – М.: Энергоатомиздат, 1988.- 480 с.
2. Зыков А.А., Теория конечных графов. / А.А. Зыков - Новосибирск: Наука, 1969.- 543 с.
3. Ловас Л., Прикладные задачи теории графов. / Л. Ловас, М. Пламмер – М.: Мир, 1998.- 496 с.
4. Кристофиде Н., Теория графов. Алгоритмический подход. / пер. с англ. Э.В. Вершкова и И.В. Коновальцева - М.: Мир, 1978.- 432 с.
5. Басакер Р., Конечные графы и сети. / Р. Басакер, Т Саати - М.: Наука, 1974.- 368 с.
6. Свами М., Графы, сети и алгоритмы. / М. Свами, К. Тхуласираман - М.: Мир, 1984.- 456 с.
7. Берж К., Теорию графов и её применение / пер. в англ. А.А. Закова - М.: Изд.ин.лит., 1962.- 320 с.
8. Татт У., Теория графов.- М.: Мир, 1988.- 424 с.
9. Харари Ф., Теория графов. / пер. с англ. В.П. Козырева - М.: Мир, 1973.- 300 с.
10. Оре О., Теория графов. / пер. с англ. И.Н.Врублевской - М.: Наука, 1980.- 336 с.
11. Халимон В.И., Использование программного комплекса «КОМПЛЕКС ГРАФ» для исследования структур сложных систем: методические указания к выполнению лабораторных работ. / В.И. Халимон, О.В. Проститенко, А.В. Крюков, А.Ю. Рогов, СПб.: СПбГТИ(ТУ), 2001.- 43 с.
12. Халимон В.И., Дискретная математика (операции на графах, булева алгебра) учебное пособие. / В.И. Халимон, О.В. Проститенко, А.Ю. Рогов, СПб.: СПбГТИ(ТУ), 2009.- 48 с.

Кафедра систем автоматизированного
проектирования и управления

Учебное пособие

Графовые методы анализа в дискретной математике

Виктория Ивановна Халимон
Александр Юрьевич Рогов
Олег Владимирович Проститенко

Компьютерный набор и верстка Рогов А.Ю.

Отпечатано с оригинал-макета. Формат 60×90 $^{1}/_{16}$.

Печ. л. 5,5. Тираж 50 экз.

Санкт-Петербургский государственный технологический институт
(Технический университет)

198013, Санкт-Петербург, Московский пр., 26
Типография издательства СПбГТИ(ТУ) т. 49-49-365