УДК 004.94

*ШИРОЧИН В.П.,*
*ПЕРЕТЯТКО Е.Н.*

# ЭФФЕКТИВНЫЙ ГРАФИЧЕСКИЙ ИНТЕРФЕЙС СИСТЕМЫ МОДЕЛЛИРОВАНИЯ

Основанная на экранных графических средствах связь между человеком и компьютером широко вошла в практику работы с операционными системами и многими приложениями. Однако, эффективность работы пользователя с графической информацией, графами различной сложности и связности остается низкой. В работе предлагаются способы представления теоретико-графовых моделей с повышенным уровнем интуитивного и когнитивного восприятия разнообразных отношений элементов в формальных структурах.

Based on the on-screen graphic means of communication between man and computer are widely included in the practice of working with many operating systems and applications. However, the effectiveness of the user with graphical information, graphs of varying complexity and connectivity remains low. The paper suggests ways to view graph-theoretic models with elevated levels of intuitive perception and cognitive relations of various elements in the formal structures.

## 1. Экранное представление формальных структур

На сегодняшний день информация часто представляется и обрабатывается в виде графов[1-4]. Например: в информатике и программировании для задания алгоритмов; в химии для описания различных структур и путей сложных реакций; в схемотехнике для представления различных соединений элементов на печатной плате; в различных транспортных и коммуникационных системах и т. д.

Для различных задач графы могут быть как очень малыми, так и чрезмерно большими. Отсюда возникает задача построения изображения графа с лучшим визуальным восприятием его[3]. Под изображением графа необходимо понимать изображение множества вершин, связанных ребрами. При визуальном восприятии графа возникают проблемы[5-8], связанные с:

– восприятием графа при большом количестве связей;

– направлением дуг в ориентированном графе;

– подписями на рёбрах;

– слиянием рёбер, размещённых на одной линии.

Далее в этой статье предложены некоторые решения этих проблем.

## 2. Проблема восприятия графа с большим количеством связей

Иногда необходимо представить граф в графической форме, так чтобы была видна его структура. К примеру, это может пригодиться при визуализация иерархии классов и пакетов исходного кода программы[3]. Зачастую такие графы содержат в себе настолько большое количество рёбер, что изображенный граф визуально разобрать практически невозможно. Примером этого является граф, изображенный на Рис.1.

***Рис. 1. Граф зависимостей программной системы***

Граф представляет собой дерево разбиения на пакеты (вершины внутри окружностей – пакеты, вершины белого цвета располагающиеся по периметру окружностей – классы), на которое поверх наложены ребра зависимости одних классов от других. Можно заметить, что граф настолько визуально перегружен, что архитектуру программы невозможно проследить.

Для устранения этой проблемы можно использовать метод, принцип которого тот же, что и в кабельных сетях. Чтобы не запутаться в проводах, когда их слишком много, провода объединяют в жгуты. Применим этот метод

(назовём его – метод «жгута») для рёбер. Рассмотрим прорисовку одного ребра:
Необходимо провести ребро из вершины $P_0$ в вершину $P_4$.

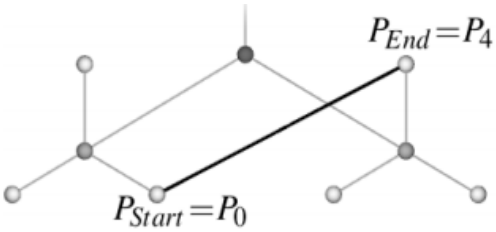


***Рис. 2. Дуга в виде прямой линии между вершинами***

Для начала необходимо найти путь между этими вершинами (Рис.3).

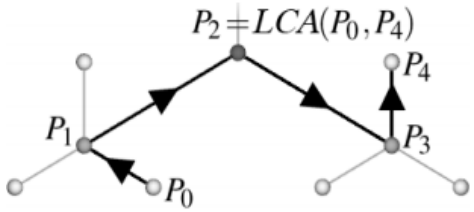


***Рис. 3. Путь между вершинами $P_0, P_4$***

Теперь проведем кривую через полигон, образованный точками $P_0$, $P_1$, $P_2$, $P_3$, $P_4$:

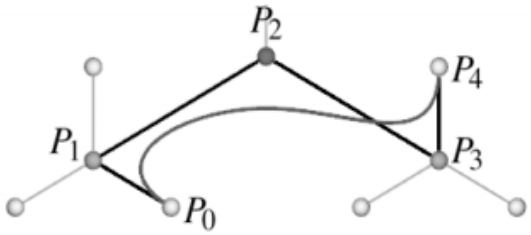


***Рис. 4. Проведение кривой между вершинами***

Для визуализации лучше всего в качестве кривых подходят кусочно-заданные кубические B-сплайны (кубический B-сплайн – это просто набор кривых третьего порядка в двухмерном пространстве, для которых выполняется условия сшивки первых и вторых производных на краях).

Введём параметр λ для управления степенью связанности ребер, который принимает значения от 0 до 1 (0 – ребра представляют собой независимые прямые линии, 1 – ребра максимально связаны друг с другом).

Математически это выглядит так:

$$S'(t) = \lambda \cdot S(t) + (1-\lambda)(P_0 + t(P_{N-1} - P_0))$$

S(t) – точки сплайна, S'(t) – результирующая кривая, которая отображается на экран в виде ребра.

После преобразований изображение графа (Рис.5) приобретает лучший вид для визуального восприятия.

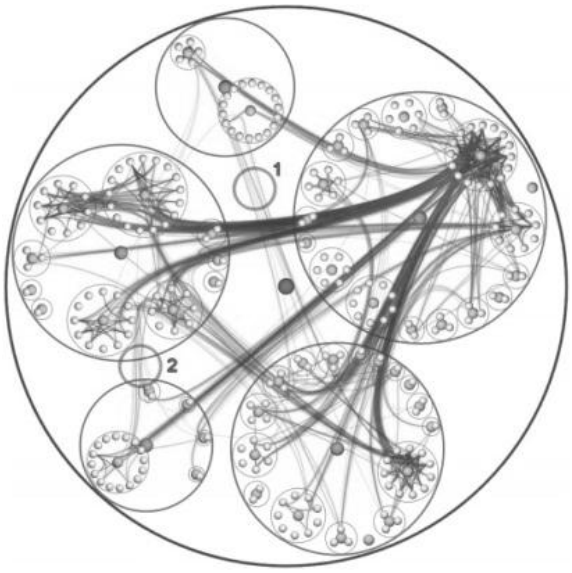


***Рис. 5. Граф после преобразования с параметром λ = 0.85***

За счет связки ребер проглядываются зависимости между пакетами, что делает представление архитектуры программы значительно яснее.

Также можно использовать радиальный способ визуализации архитектуры. Для примера возьмем тот же граф, который разобьем не по пакетам, а по радианам (Рис.6).

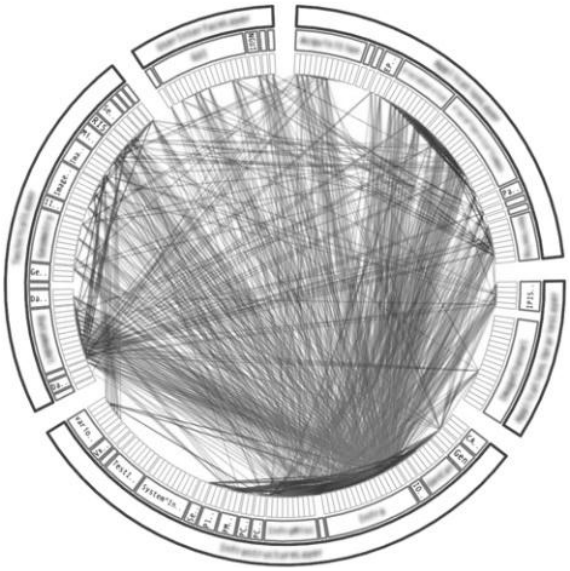

***Рис. 6. Граф с параметром λ = 0***

После применения метода связывания дуг по радианам визуальное восприятие графа улучшилось. Как видно с Рис.7, четко прослеживаются связи внутри пакетов и между ними.

Метод связывания ребер можно сочетать с различными методами рисования деревьев, что является его большим преимуществом.

### 3. Проблема представления направления дуг в ориентированном графе

При довольно-таки сложных графах с большим количеством рёбер сложно определить направление каждого ребра, потому что либо эти стрелки будут очень маленькими и их не будет заметно, либо они будут сливаться[3,4]. Чтобы не рисовать стрелки направления, ребра можно нарисовать в виде градиентных линий, где один цвет – это начало, а другой – конец ребра. Такое представление дает нам возможность однозначно определять начало и конец каждого ребра даже в очень загруженных графах.

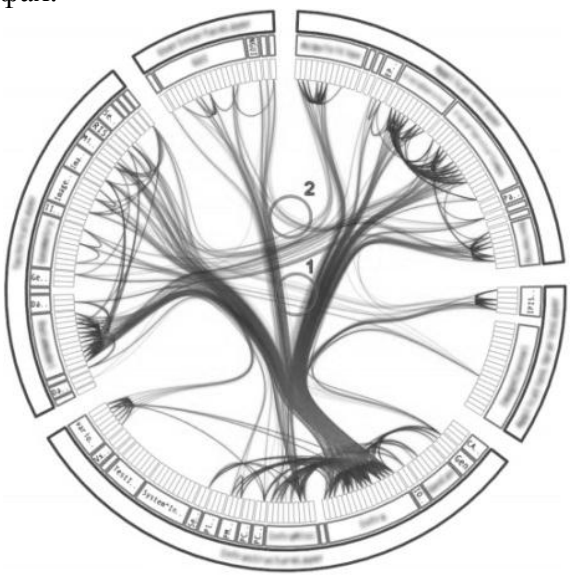

***Рис.7. Граф с параметром связывания λ=0.85***

### 4. Проблема восприятия подписей на рёбрах

Часто возникает необходимость сделать маркировку ребер графа[4-6]: указать условия, вероятности переходов и т.д. Но при большой сложности графа, не всегда можно правильно установить отношение между конкретной маркировкой и ребром. Как одно из решений этой проблемы – метод построения таблицы связей между вершинами. Поскольку большинство графов являются довольно-таки массивными (чрезмерно много вершин и рёбер), часто возникает сложность в нахождении связей между ними.

Суть метода заключается в том, что выделяя две вершины, всплывает таблица, где указаны все рёбра их соединяющие и соответствующие им маркировки. Это облегчит граф и сделает более удобным нахождение нужной вам маркировки.

### 5. Проблема слияния дуг

При отображении графов дуги часто изображаются в виде прямых линий[4]. Расположение прямых дуг на одной линии приводит к их визуальному слиянию, что затрудняет анализ графа. Следовательно, нужно изменить форму рёбер так, чтобы они не сливались. Одно из решений – представление дуги в виде окружности (метод изгиба дуг).

Для этого у нас есть точки A(x1;y1) и B(x2;y2). Построение окружности возможно только в том случае если знать центр окружности O(x0;y0) и его радиус R.

Для этого можно выбрать радиус круга равным расстоянию между точками A и B. Тогда $R=\sqrt{(x2-x1)^2+(y2-y1)^2}$, а треугольник ABO будет равносторонним (Рис.8).

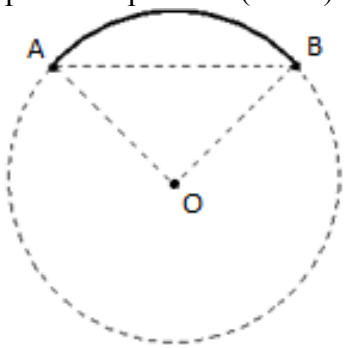


***Рис.8. Построение дуги***

Тогда высота, проведенная из точки O к отрезку AB, будет равна $\frac{\sqrt{3}}{2}R$, а точка пересечения P будет иметь координаты $\left(\frac{x1+x2}{2};\frac{y1+y2}{2}\right)$. Тогда для высоты получим следующее уравнение:

$$\frac{\sqrt{3}}{2}R=\sqrt{\left(\frac{x1+x2}{2}-x0\right)^2+\left(\frac{y1+y2}{2}-y0\right)^2} \quad (1)$$

Использовав уравнение прямой, которая проходит через 2 точки можно определить коэффициент наклона прямой AB как $K1=\frac{y2-y1}{x2-x1}$.

Поскольку произведение коэффициентов наклона перпендикулярных прямых равно -1, то коэффициент наклона прямой OP равен $K2=\frac{x2-x1}{y1-y2}$.

Используя уравнение прямой, проходящей через одну точку и коэффициент наклона, получим уравнение $y - \frac{y1+y2}{2} = K2 \cdot (x - \frac{x1+x2}{2})$. Если учесть, что точка O также лежит на этой прямой уравнение примет вид

$$y0 - \frac{y1+y2}{2} = K2 \cdot (x0 - \frac{x1+x2}{2}) \qquad (2)$$

Объединив уравнения 1 и 2 в систему и решив ее получим 2 точки, которые определяют центры двух окружностей. Какую из них выбирать определяет разработчик.

### 6. Практическое применение метода изгиба дуг

Для демонстрации метода изгиба дуг создан программный пакет MultiSim, который использует математику описанную выше. При построении дуг прямыми, они часто сливаются. Пакет реализован на языке Java.

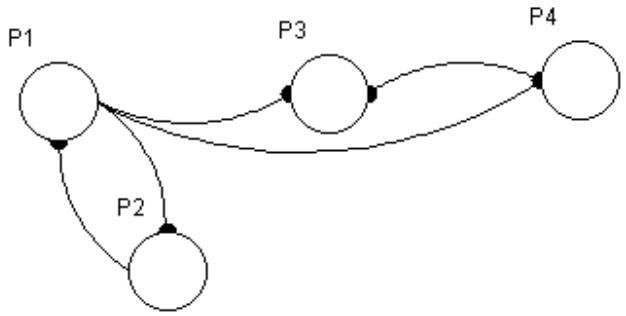


***Рис.9. Пример метода изгиба дуг***

Результат работы данной программы можно увидеть на Рис.9. Как видно, вершины (P1, P3, P4) стоят на одной линии. Если бы не был применён метод изгиба дуг, то связи сливались бы в одну линю, и определить, какие вершины связаны практически не возможно.

### 7. Заключение

Исходя из изложенного в работе видно, что при изображении графов существует множество различных проблем. Их решения не эффективны либо очень трудоёмкие. В работе предложены методы, которые являются достаточно простыми и эффективными.

Метод «жгута» позволяет связать ребра для лучшей визуализации связей между определенными группами.

Дуга в виде градиентной линии – удобное представление начала и конца ребра в массивных графах.

Вынесение в таблицу маркировок на связях между вершинами сильно облегчит визуальное восприятие графа и поиск необходимой маркировки.

Представление дуг в виде окружностей предотвратит слияние ребер графа.

Однако предложенные решения охватывают лишь небольшую часть проблем связанных с отображением графа. Поэтому необходимо их более детальное изучение и нахождение соответствующих решений.

### Список литературы

1. Брейер М. А., Теория и методы автоматизации проектирования вычислительных систем – Москва: Мир, 1977. – С. 108-205.
2. Berge C. B., The Theory of Graphs and Its Applications, New York, Wiley, Inc., 1962
3. G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999
4. Касьянов В. Н., Евстигнеев В. А., “Графы в программировании: обработка, визуализация и применение”, 2003
5. Peterson, J. L., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981
6. Kumar, D. and Harous, S., "Distributed Simulation of Timed Petri Nets: Basic Problems and Their Resolution", IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 10, 2002
7. Holliday, M. A. and Vernon, M. K., "A Generalized Timed Petri Net Model for Performance Analysis", IEEE Trans. on Software Eng., vol. SE-13, No. 12, 2006, pp. 1297-131
8. Евстигнеев В.А., Применение теории графов в программировании, 2009
9. David Joyner, Minh Van Nguyen, Nathan Cohen**,** Algorithmic Graph Theory, 2010