К ВОПРОСУ ВИЗУАЛИЗАЦИИ МАРШРУТОВ ВЫПОЛНЕНИЯ ФУНКЦИОНАЛЬНЫХ ОБЪЕКТОВ И ВЕТВЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ

В.А. Безруков, кандидат технических наук, доцент.
Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики.
С.Н. Терёхин, доктор технических наук, доцент;
В.В. Кутузов, кандидат технических наук, доцент.
Санкт-Петербургский университет ГПС МЧС России

В целях обеспечения безопасности информации в современных программных комплексах, актуален вопрос контроля отсутствия недекларированных возможностей программного обеспечения. Процедура сертификации средств защиты информации предполагает определение маршрутов выполнения функциональных объектов и их ветвей. Маршрут выполнения представляет собой ориентированный граф. В статье рассмотрена задача разработки программного комплекса, позволяющего визуализировать маршруты выполнения функциональных объектов и ветвей программного обеспечения средств защиты информации.

Ключевые слова: сертификация, граф, визуализация, программное обеспечение, защита информации

TO THE QUESTION ON THE VISUALIZATION OF ROUTES REALIZATIONS OF FUNCTIONAL OBJECTS AND SOFTWARE BRANCHES IN CERTIFIED SISTEM OF INFORMATION PROTECTION

V.A. Bezrukov.
Saint-Petersburg national research university of information technologies, mechanics and optics.
S.N. Terehin; V.V. Kutuzov.
Saint-Petersburg university of State fire service of EMERCOM of Russia

In order to ensure information security in modern software systems, control of the absence of software undeclared capabilities is claimed to be the topical issue. The certification process for information security products involves execution route determination for the functional objects and their branches. The execution route is a directed graph. The paper deals with the problem of software complex development that gives the possibility to visualize execution routes for the functional objects and software branches of information security products.

Keywords: certification, graph, visualization, software, information protection

В современных программных комплексах актуален вопрос обеспечения безопасности информации, в частности контроль отсутствия недекларированных возможностей программного обеспечения.

В соответствии с положением о сертификации средств защиты информации по требованиям безопасности информации предприятие, осуществляющее деятельность по обработке информации, составляющей государственную тайну, обязано использовать сертифицированные средства защиты информации [1].

Процедура сертификации средств защиты информации предполагает, в зависимости от степени секретности обрабатываемой информации, определение маршрутов выполнения функциональных объектов и их ветвей [1].

Маршрут выполнения представляет собой ориентированный граф. Как известно, существует три способа задания графа [2]: аналитический, матричный, геометрический.

Матричный способ заключается в задании графа в виде матрицы смежности или матрицы инцидентности и используется при обработке данных вычислительными средствами.

Визуализация графов представляет собой отдельную дисциплину на стыке теоретической информатики и математики. Данная дисциплина получила широкое развитие в силу разнообразия областей практического применения:

– разработка программного обеспечения (графы вызовов, UML-диаграммы);
– вычислительная биология;
– вычислительные сети;
– социальные науки.

Решения в области визуализации графов условно можно разделить на две группы. К первой группе относятся самостоятельные приложения [3, 4]: GePhi, Cytoscape, анализатор SciTools Understand. Вторую составляют библиотеки [5, 6]: Cytoscape.js, D3.js, GraphViz, NetworkX.

Существуют следующие алгоритмы изображения графов [7]: силовые алгоритмы, иерархические алгоритмы, спектральные алгоритмы, ортогональные алгоритмы, круговые алгоритмы, древовидные алгоритмы.

Из перечисленных видов алгоритмов иерархический или поуровневый способ изображения является наиболее широко используемым для отслеживания порядка выполнения функциональных объектов и ветвей при анализе программного обеспечения.

Традиционный метод Сугиямы предполагает наличие четырех этапов [7]: удаление циклов, разбиение на слои, минимизация пересечений ребер, назначение x-координат.

В оригинале метод Сугиямы предполагал работу с ациклическими графами, поэтому удаление циклов рассматривается в качестве этапа предварительной подготовки [8]. Для ориентированного графа понятие ацикличности отлично от неориентированного: орграф может иметь неориентированные циклы и при этом быть ациклическим.

Проблема построения наибольшего подграфа без циклов или поиска такого наибольшего множества $E_a \subset E$, что граф (V, E_a) ацикличен, – чаще всего формулируется как проблема поиска наименьшего множества ребер обратной связи или FAS. Необходимо найти такое множество $E_f \subset E$, что $G = (V, E \setminus E_f)$ не содержит циклов [8–10].

В пакете программ GraphViz [6] средство dot, предназначенное для поуровневого построения графов, удаление циклов производит посредством итерирования по сильно связанным компонентам и осуществления в них поиска в глубину.

Разбиение на слои – следующий этап алгоритма Сугиямы, который на выходе присваивает каждой вершине графа дополнительный параметр – уровень. Результатом служит разбиение L для множества V графа G=(V,E), такое что $L = \{L_1, L_2, \dots, L_h\}$ и $\bigcup_{i-1}^{k} L_i = V$ [11].

Минимизация пересечений ребер. По результатам предыдущего этапа сформирован граф $(L_1 \cup L_2 \dots \cup \dots L_h, E)$, который с немалой вероятностью обладает большим количеством пересечений. Пересечения не несут полезной информационной нагрузки, при этом затрудняя восприятие пользователем изображения. Это обусловливает необходимость минимизации пересечений.

Основным фактором наличия пересечений выступают относительные позиции вершин [12], поэтому минимизация пересечений ребер признается комбинаторной проблемой, которая сводится к нахождению такого упорядочения вершин, что число пересечений между слоями становится минимальным. Существует множество методов минимизации пересечений. Наиболее простыми методами считаются метод барицентров и метод медиан.

Назначение координат. После того, как относительный порядок вершин определен, остается найти абсолютные координаты вершин и построить изображения ребер [7]. Задача вычисления координат вершин решается либо с помощью оптимизации некоторой целевой функции при заданных ограничениях, либо методами итеративного улучшения имеющегося размещения.

Разработку архитектуры программного комплекса целесообразно разделить на два этапа формирования изображения графа: присваивания координат и вывод полученной укладки.

Возможность просмотра результата работы программного комплекса должна обеспечиваться браузерами операционных систем Windows, Linux, Mac OS X.

С учетом положений, изложенных выше, была разработана архитектура программного комплекса, представленная на рис. 1.

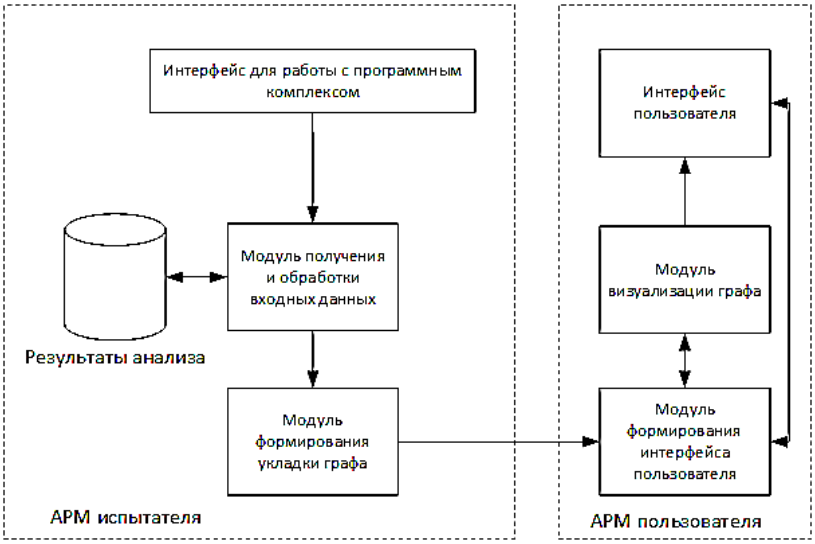

Рис. 1. **Архитектура программного комплекса (АРМ – автоматизированное рабочее место)**

Из рис. 1 видно, что передача данных на модуль формирования укладки графа осуществляется через модуль получения и обработки входных данных или результатов анализа. Данное разделение позволяет унифицировать входной формат для модуля формирования укладки при изменении характера данных, подлежащих визуализации, на стороне анализатора.

Таким образом, на сторону конечного пользователя подается результат работы модуля формирования укладки графа, представляющий собой сериализованный объект, содержащий информацию о вершинах графа (функциональных объектах или ветвях) и их связях, а также координаты вершин.

Модуль формирования интерфейса, получая на вход сериализованный объект, передает его модулю визуализации, который создает окончательное изображение, являющееся частью графического интерфейса; другая часть графического интерфейса, несущая регулирующую функцию, формируется с помощью этого же модуля.

Модуль получения и обработки входных данных:

– обрабатывает информацию, переданную при помощи прикладных программных интерфейсов от анализатора;

– осуществляет соединение с хранилищем данных, содержащим результаты работы анализатора;

– запрашивает и получает данные о функциональных объектах и ветвях программного обеспечения средств защиты информации;

– преобразует полученные данные в единый формат, пригодный для работы модуля формирования укладки графа.

Модуль формирования укладки графа:

– осуществляет анализ поступивших данных о вершинах и ребрах графа;

– производит преобразование графа посредством добавления координат к каждой вершине, обозначающей функциональный объект или ветвь программного обеспечения, в соответствии с методом Сугиямы;

– экспортирует полученный граф в сериализованный объект, пригодный для дальнейшей обработки модулем формирования интерфейса пользователя.

Модуль формирования интерфейса пользователя:

– принимает и при необходимости обрабатывает сериализованный объект, полученный в результате работы модуля формирования укладки граф, для последующей передачи модулю визуализации графа;

– получает от графического интерфейса пользователя сигналы о виде взаимодействия с графом и отправляет их модулю визуализации графа;

– осуществляет первичную обработку полученного сигнала и формирует параметры для взаимодействия с модулем визуализации графа.

Модуль визуализации графа:

– формирует изображение графа вызовов;

– принимает команды от модуля формирования интерфейса пользователя;

– изменяет изображение графа в соответствии с командами пользователя.

Архитектурное разделение программного комплекса позволило дифференцировать и языки программирования, на которых производилась разработка.

АРМ испытателя, на котором осуществляется анализ и формируется промежуточное представление графа, было поставлено условие минимизации количества сторонних компонент, не участвующих непосредственно в работе анализатора. Данное условие определило выбор языка программирования Python 3 в качестве языка разработки.

АРМ пользователя предоставляет необходимый уровень взаимодействия с изображением графа стандартными средствами интернет-браузеров. Данное требование обусловливает использование языка разметки HTML для формирования контейнера результатов визуализации и библиотеки на языке JavaScript для работы с графом. Язык программирования JavaScript был выбран языком разработки для части программного комплекса, функционирующего на АРМ пользователя.

Существует несколько разработанных на языке Python библиотек, позволяющих произвести укладку графа по методу Сугиямы: это фреймворки Grandalf, igraph и NetworkX [13, 14]. Все они являются средствами с открытым исходным кодом и, в силу базирования на языке Python, мультиплатформенными. Проведенное сравнение фреймворков показало, что наиболее предпочтительным для данной задачи является NetworkX, использующий средства пакета программ GraphViz.

Создание интерактивных изображений графа возможно с применением одной из двух технологий [15]: SVG, Canvas.

Существуют две известных библиотеки по построению, визуализации и анализу графов – D3.js и Cytoscape.js. D3.js использует технологию SVG, а Cytoscape.js технологию Canvas. Поэтому, в качестве модуля визуализации графа была выбрана Cytoscape.js.

В рамках данной работы была произведена программная реализация программного комплекса, разработка велась на языках Python 3.4 и JavaScript. В качестве тестового примера была выбрана программа fbReader, исходные тексты которой находятся в открытом доступе, было насчитано более восьми тысяч функциональных объектов.

На рис. 2 представлен общий вид результата работы программного комплекса.

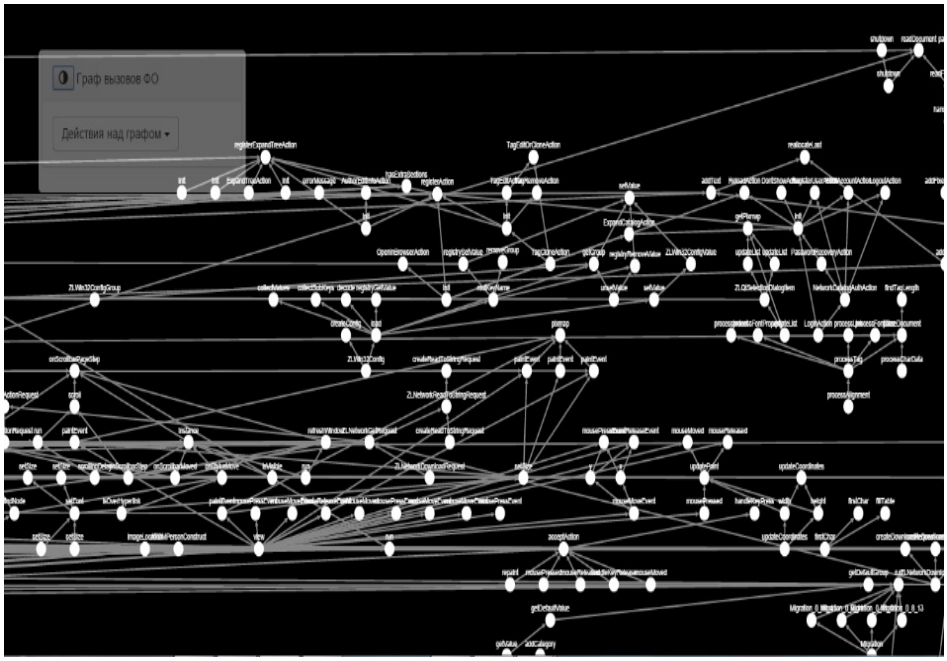

Рис. 2. **Граф вызовов функциональных объектов, увеличенный**

Вершинами являются функциональные объекты, ребра представляют связи между ними. Результат работы программы представляет собой html-файл с подгружаемыми .js и .css файлами, предназначенный для запуска в браузере операционной системы.

Информация о функциональном объекте представлена на рис. 3. Подсвеченные вершины имеют непосредственную связь с искомым функциональным объектом.

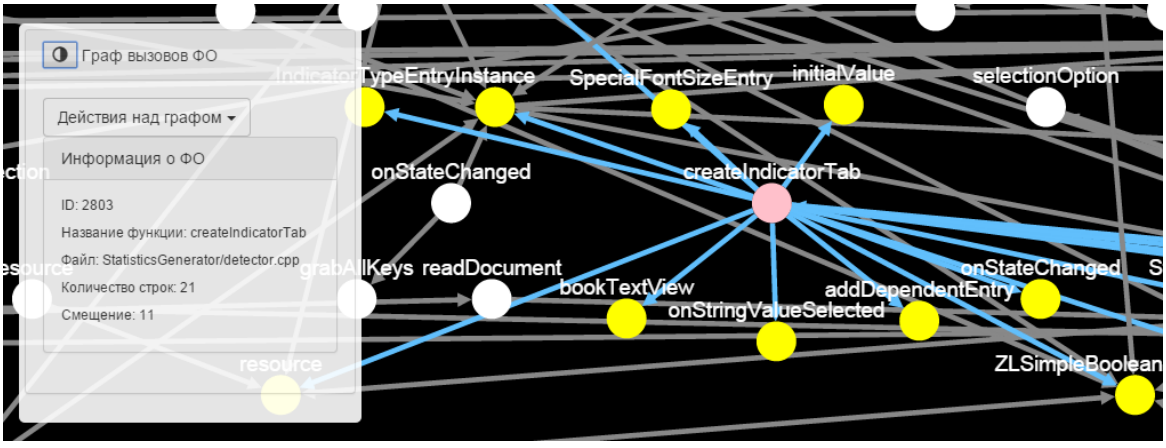

Рис. 3. **Информация о функциональном объекте**

Был реализован поиск пути между двумя вершинами графа. При исследовании исходных кодов программного обеспечения перед испытателем зачастую возникает задача

выяснения наличия связи между двумя функциональными объектами. Например, в проанализированном программном обеспечении присутствует функция registrySetValue, устанавливающая значения реестра и показанная на рис. 4. Пусть необходимо выяснить, является ли одним из маршрутов выполнения вызов данной функции из функции ZLWin32Config, управляющей конфигурациями программного обеспечения.

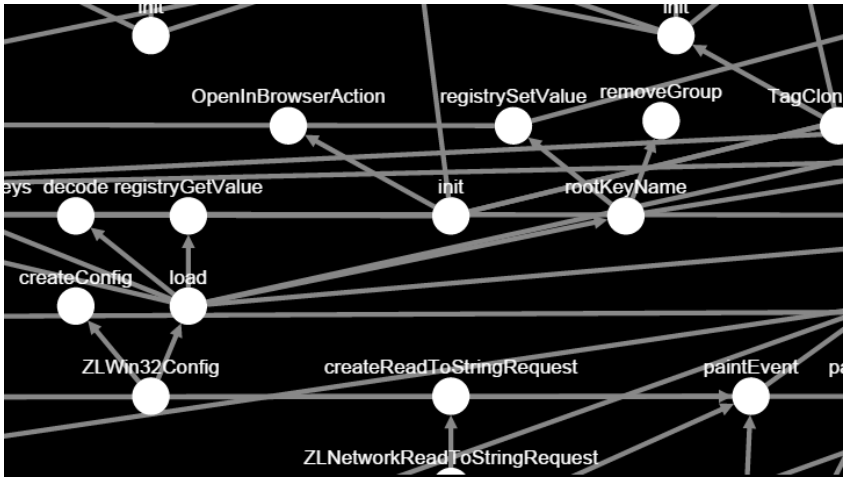

Рис. 4. **Участок графа с функцией registrySetValue**

С помощью программного комплекса удается выяснить наличие подобной связи и вывести маршрут выполнения, что показано на рис. 5.

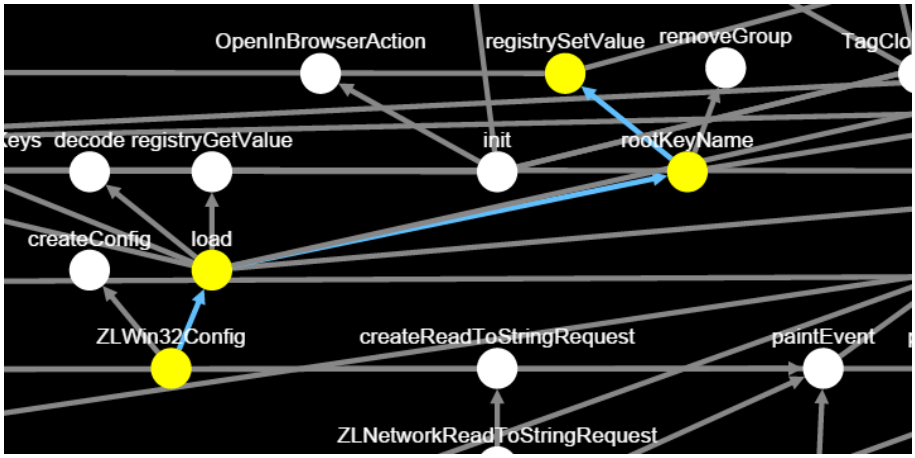

Рис. 5. **Маршрут от функции ZLWin32Config до registrySetValue**

Контрольный запуск модуля формирования укладки и модуля визуализации показал следующее:

– укладка графа средствами NetworkX производилась 25 сек.;

– время визуализации тестового графа составляло от 7 до 30 сек. в зависимости от браузера.

Таким образом, использование встроенной реализации алгоритма Сугиямы в библиотеке Cytoscape.js (средство dagre) показало, что при наличии нескольких тысяч вершин его работоспособность снижается с ростом числа ребер; при количестве ребер, большем тысячи, происходит переполнение стека вызовов в браузере и изображение не формируется. Эти данные позволяют предположить, что разделение задач укладки и визуализации в контексте построения маршрутов выполнения функциональных объектов дало эффективный результат.

В данной работе была решена задача уменьшения требований к АРМ пользователя при сохранении возможности взаимодействия с результатом визуализации. Этого результата удалось достичь благодаря архитектурному разделению программного комплекса на две компоненты, первая из которых формирует укладку графа и передает её в виде сериализованного объекта, а вторая выполняется независимо от первой, принимает на вход укладку графа и формирует средствами браузера искомое изображение, а также обеспечивает возможность взаимодействия пользователя с графом.

Литература

1. Руководящий документ. Защита от несанкционированного доступа к информации. Ч. 1: Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей (введ. с 4 июня 1999 г.). М.: Изд-во стандартов, 1999. 9 с.

2. Graph Drawing // Graph Drawing Symposium. URL: http://www.graphdrawing.org/ (дата обращения: 11.07.2017).

3. Gephi Features // Gephi.org. URL: https://gephi.github.io/features/ (дата обращения: 11.07.2017).

4. Cytoscape.js // Cytoscape Consortium. URL: http://js.cytoscape.org/ (дата обращения: 11.07.2017).

5. D3.js – Data Driven Documents // D3.js. URL: http://d3js.org/ (дата обращения: 11.07.2017).

6. NetworkX Documentation // NetworkX Developers. URL: http://networkx.github.io/documentation/networkx-1.9.1/ (дата обращения: 11.07.2017).

7. Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical system structures // IEEE Transactions on Systems, Man and Cybernetics. SMC-11: 1981.

8. Апанович З.В. Методы визуализации информации при помощи графов. Ч. 2: Методы визуализации ориентированных и неориентированных графов // Электронная образовательная среда НГУ. URL: http://193.124.209.204/?db=book_apanovich2&int=VIEW&class=ROOT&templ=VIEW (дата обращения: 11.07.2017).

9. Eades P., Lin X., Smyth W.F. A fast and effective heuristic for the feedback arc set problem // Information Processing Letters. 1993. P. 47.

10. Berger B., Shor P.W. Approximation algorithms for the maximum acyclic subgraph problem // Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, 1990. P. 236–243.

11. Nachmanson L. Notes on an Implementation of Sugiyama's Scheme // Graph Drawing – Springer. 2010. № 10. 10 p.

12. Shabbeer A., Ozcaglar C., Bennett K.P. Crossing Minimization within Graph Embeddings. Cornell University, 2012.

13. bdcht/grandalf // GitHub. URL: https://github.com/bdcht/grandalf (дата обращения: 11.07.2017).

14. igraph/igraph // GitHub. URL: https://github.com/igraph/igraph (дата обращения: 11.07.2017).

15. SVG vs canvas: how to choose // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/gg193983(v=vs.85).aspx (дата обращения: 11.07.2017).

References

1. Rukovodyashchij dokument. Zashchita ot nesankcionirovannogo dostupa k informacii. CH. 1: Programmnoe obespechenie sredstv zashchity informacii. Klassifikaciya po urovnyu kontrolya otsutstviya nedeklarirovannyh vozmozhnostej (vved. s 4 iyunya 1999 g.). M.: Izd-vo standartov, 1999. 9 s.

2. Graph Drawing // Graph Drawing Symposium. URL: http://www.graphdrawing.org/ (data obrashcheniya: 11.07.2017).

3. Gephi Features // Gephi.org. URL: https://gephi.github.io/features/ (data obrashcheniya: 11.07.2017).

4. Cytoscape.js // Cytoscape Consortium. URL: http://js.cytoscape.org/ (data obrashcheniya: 11.07.2017).

5. D3.js – Data Driven Documents // D3.js. URL: http://d3js.org/ (data obrashcheniya: 11.07.2017).

6. NetworkX Documentation // NetworkX Developers. URL: http://networkx.github.io/documentation/networkx-1.9.1/ (data obrashcheniya: 11.07.2017).

7. Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical system structures // IEEE Transactions on Systems, Man and Cybernetics. SMC-11: 1981.

8. Apanovich Z.V. Metody vizualizacii informacii pri pomoshchi grafov. CH. 2: Metody vizualizacii orientirovannyh i neorientirovannyh grafov // EHlektronnaya obrazovatel'naya sreda NGU. URL: http://193.124.209.204/?db=book_apanovich2&int=VIEW&class=ROOT&templ=VIEW (data obrashcheniya: 11.07.2017).

9. Eades P., Lin X., Smyth W.F. A fast and effective heuristic for the feedback arc set problem // Information Processing Letters. 1993. P. 47.

10. Berger B., Shor P.W. Approximation algorithms for the maximum acyclic subgraph problem // Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, 1990. P. 236–243.

11. Nachmanson L. Notes on an Implementation of Sugiyama's Scheme // Graph Drawing – Springer. 2010. № 10. 10 p.

12. Shabbeer A., Ozcaglar C., Bennett K.P. Crossing Minimization within Graph Embeddings. Cornell University, 2012.

13. bdcht/grandalf // GitHub. URL: https://github.com/bdcht/grandalf (data obrashcheniya: 11.07.2017).

14. igraph/igraph // GitHub. URL: https://github.com/igraph/igraph (data obrashcheniya: 11.07.2017).

15. SVG vs canvas: how to choose // Microsoft Developer Network. URL: https://msdn.microsoft.com/en-us/library/gg193983(v=vs.85).aspx (data obrashcheniya: 11.07.2017).