**Содержание**

Введение 7

1 Постановка задачи 8

2 Существующие аналоги 9

[2.1 Векторный редактор Diagrams.net 9](#_Toc181559294)

[2.2 Векторный редактор LucidChart 10](#_Toc181559295)

[2.3 Векторный редактор Microsoft Visio 12](#_Toc181559296)

[3 Модели и алгоритмы автоматизации проектирования и визуализации схем и диаграмм 14](#_Toc74595712)

[3.1 Объектно-ориентированные модели 14](#_Toc74595713)

[3.2 Иерархические модели 15](#_Toc74595714)

[3.3 Графовые модели 16](#_Toc74595714)

[3.4 Методы оптимизации схем: алгоритм раскраски графов, линейное программирование, генетические и эвристические алгоритмы 16](#_Toc74595716)

[4 Обзор научных исследований по автоматизации визуализации схем и диаграмм 25](#_Toc74595721)

[Заключение 29](#_Toc74595728)

[Список использованных источников 31](#_Toc74595729)

Введение

Автоматизация проектирования и визуализации компьютерных схем и диаграмм является актуальной задачей, связанной с развитием сложных технических и информационных систем. Современные программные инструменты должны обеспечивать эффективное создание, редактирование и анализ диаграмм различной сложности, что требует применения адаптивных алгоритмов и методов оптимизации.

Системы автоматизированного проектирования широко применяются в различных областях, включая инженерное проектирование, бизнес-аналитику, моделирование процессов и разработку программного обеспечения. Важность таких инструментов обусловлена необходимостью повышения скорости и удобства работы с графическими моделями, обеспечения высокой читаемости диаграмм, а также минимизации временных и вычислительных затрат при обработке больших графов.

Разработка и совершенствование программных средств для проектирования схем требует учета множества факторов, таких как удобочитаемость визуальных элементов, адаптивность интерфейса, масштабируемость и интеграция с другими системами.

Анализ отечественных и зарубежных исследований показывает, что несмотря на наличие различных подходов к автоматизации визуализации диаграмм, ряд проблем остается нерешенным. В частности, недостаточно разработаны методы адаптивного построения сложных схем, оптимизированные с точки зрения читаемости и взаимодействия с пользователем. Это определяет необходимость дальнейшего изучения и разработки эффективных алгоритмов, позволяющих повысить качество автоматизированного проектирования диаграмм.

Данная работа направлена на исследование моделей, методов и алгоритмов автоматического построения схем и диаграмм с учетом адаптивности интерфейса и удобочитаемости визуальных элементов.

1 Постановка задачи

В рамках дальнейшего научного исследования планируется разработка теоретических основ и методов автоматизации построения схем и диаграмм с улучшенной адаптивностью интерфейса.

Объектом исследованиянаучной работы являются процессы автоматизации проектирования и визуализации схем и диаграмм, связанные с построением и сопровождением схем в различных областях проектирования.

Предметом исследования являются математические модели, методы, алгоритмы и программные средства, ориентированные на создание, редактирование и поддержку диаграмм и схем, отвечающие специфическим требованиям адаптивности и гибкости.

В связи с определенными выше объектом и предметом исследования была поставлена цель работы, которая заключается в разработке методов и алгоритмов автоматического построения диаграмм с учетом адаптивности интерфейса и удобочитаемости.

Для реализации поставленной цели сформулированы следующие задачи:

* Провести анализ существующих редакторов схем и диаграмм, выявить их подходы к компоновке элементов и ограничения;
* Исследовать современные алгоритмы автоматической оптимизации графов, включая раскраску графов, методы линейного программирования, генетические и эвристические алгоритмы;
* Определить основные критерии удобочитаемости схем и разработать модель, учитывающую эти параметры;
* Проанализировать влияние различных методов компоновки графов на адаптивность пользовательского интерфейса;
* Выявить перспективные направления совершенствования алгоритмов автоматизации построения диаграмм для их применения в редакторах схем.

2 Существующие аналоги

В сфере векторных редакторов компьютерных схем и диаграмм существует ограниченное количество предлагаемых сервисов. Далее будут описаны следующие аналоги: Diagrams.net, LucidChart, Microsoft Visio, SmartDraw, yEd Graph Editor. Оба приложения являются бесплатными и свободными для скачивания пользователями.

### **2.1 Векторный редактор Diagrams.net**

Редактор схем и диаграмм Diagrams.net — это веб-приложение для построения схем и диаграмм, поддерживающее широкий спектр инструментов для их создания и редактирования. Благодаря своей бесплатной и доступной онлайн-платформе, «Diagrams.net» стал популярным выбором среди пользователей, которым необходимо создавать профессиональные и качественные графические представления. Платформа поддерживает несколько типов диаграмм, в том числе организационные, потоковые и сетевые схемы, а также поддерживает совместную работу в реальном времени, что делает ее идеальным выбором для командной работы над проектами. Одним из главных преимуществ Diagrams.net является его бесплатность и возможность использования без установки дополнительного программного обеспечения. Тем не менее, некоторые расширенные функции могут быть доступны только в платной версии, а также могут возникнуть ограничения в поддержке форматов файлов [1].

Интерфейс приложения представлен на рисунке 1.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.1 – Интерфейс приложения Diagrams.net

К основным достоинствам Diagrams.net можно отнести следующее:

* полный функционал доступен бесплатно;
* разнообразие диаграмм;
* интеграции с G Suite, Google Drive, Dropbox, OneDrive, Github, GitLab;
* широкий выбор инструментов и шаблонов;
* поддержка плагинов для популярной платформы для отслеживания ошибок и координации проектов Jira.

К основным недостаткам Diagrams.net можно отнести следующее:

* невозможность одновременной работы над одним проектом нескольким пользователям;
* недостаточная гибкость в настройке элементов диаграмм;
* ограниченный выбор форматов экспорта файлов;
* отсутствие некоторых основных категорий шаблонов;
* нет версий приложений для Windows и Mac;
* медленная работа с большими диаграммами.

### **2.2 Векторный редактор LucidChart**

Редактор схем и диаграмм LucidChart является онлайн-платформой для создания диаграмм и схем, предоставляющей пользователю широкие возможности для визуализации различных процессов и концепций. Основной функционал включает в себя создание различных типов диаграмм, таких как организационные схемы, блок-схемы, ER-диаграммы, потоковые диаграммы и многие другие. Пользователи могут создавать новые диаграммы с нуля или использовать готовые шаблоны для ускорения процесса. LucidChart также предлагает возможность совместной работы над проектами в режиме реального времени, что делает его удобным инструментом для работы в команде [2].

Интерфейс приложения представлен на рисунке 1.2.

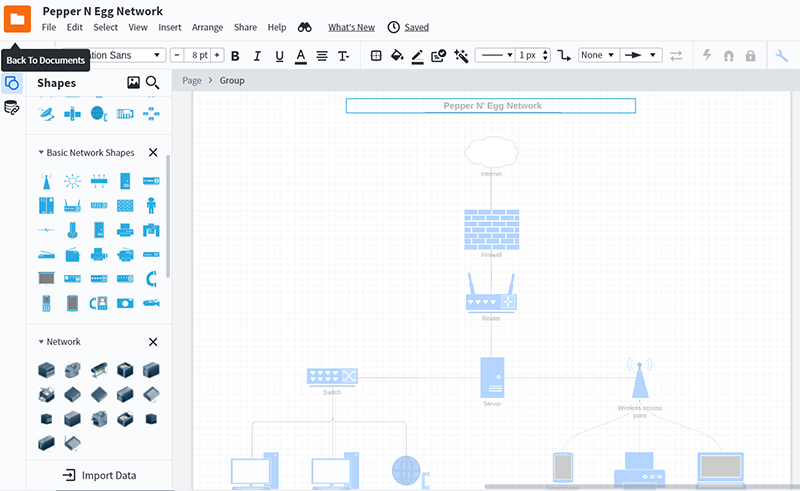


Рисунок 2.2 – Интерфейс приложения LucidChart

К основным достоинствам LucidChart можно отнести следующее:

* легкость в использовании и доступность;
* возможность совместной работы в режиме реального времени;
* широкий выбор инструментов и шаблонов.

К основным недостаткам LucidChart можно отнести следующее:

* ограниченные возможности бесплатной версии;
* небольшие проблемы с производительностью при работе с большими проектами;
* нет возможности работать без подключения к интернету;
* проблемы с синхронизацией данных между устройствами.

### **2.3 Векторный редактор Microsoft Visio**

Редактор схем и диаграмм Microsoft Visio является программным обеспечением для создания профессиональных диаграмм и схем, разработанным компанией Microsoft. Он предоставляет широкий набор инструментов для создания различных типов диаграмм, включая организационные диаграммы, сетевые диаграммы, планы этажей и т. д. Microsoft Visio интегрируется с другими приложениями Microsoft Office, что обеспечивает пользователю единый и удобный интерфейс для работы [3].

Интерфейс приложения представлен на рисунке 1.3.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3 – Интерфейс приложения Microsoft Visio

К основным достоинствам Microsoft Visio можно отнести следующее:

* богатый выбор инструментов для создания различных типов диаграмм;
* интеграция с другими приложениями Microsoft Office;
* возможность работы в офлайн-режиме;
* простота освоения и использования;
* удобный интерфейс и инструменты;
* наличие большого количества шаблонов;
* возможность соединять блок схемы и диаграммы с данными в режиме реального времени.

К основным недостаткам Microsoft Visio можно отнести следующее:

* непросто распечатать и опубликовать диаграммы;
* сложность освоения для новичков;
* недоступно для операционных систем macOS и Linux.

3 Модели и алгоритмы автоматизации проектирования и визуализации схем и диаграмм

Автоматизация проектирования и визуализации схем и диаграмм — это многогранная задача, требующая применения различных моделей, методов, алгоритмов и подходов. В данной области накоплен значительный объем исследований, посвященных как теоретическим аспектам, так и практическим решениям, включая разработку программных средств. Среди популярных подходов к автоматизации проектирования стоит отметить объектно-ориентированные, иерархические и графовые модели, а также алгоритмы оптимизации и эвристические методы.

### **3.1 Объектно-ориентированные модели**

Объектно-ориентированные модели позволяют представить элементы системы в виде объектов, которые обладают свойствами и методами. Это облегчает управление сложными структурами, поддерживает модульность и обеспечивает повторное использование компонентов. Например, каждый элемент диаграммы (узел, блок, соединитель) может быть представлен как объект с чётко определёнными атрибутами, такими как размеры, положение, цвет или логическое поведение. Наследование и полиморфизм дают возможность создавать специализированные элементы, сохраняя при этом их интеграцию в общую структуру.

Преимущества объектно-ориентированных моделей:

* Модульность. Система разбивается на независимые компоненты (объекты), что упрощает сопровождение, тестирование и повторное использование;
* Наследование и повторное использование кода. Позволяет создавать новые компоненты на основе существующих, снижая трудозатраты и повышая согласованность архитектуры;
* Полиморфизм и расширяемость. Универсальные интерфейсы позволяют реализовывать поведение, адаптируемое к контексту, облегчая модификации и расширение функциональности без нарушения текущей логики.

Объектно-ориентированное моделирование применимо, например, в системах автоматизации проектирования инженерных сетей. При этом можно описать систему в виде набора классов, где каждый класс представляет определённый тип оборудования или соединения. Это позволяет строить сложные структуры из модульных компонентов, поддерживать их обновление и масштабирование [4].

### **3.2 Иерархические модели**

Иерархические модели, в свою очередь, обеспечивают структурированное представление данных. В них элементы верхнего уровня объединяют связанные элементы нижнего уровня, что позволяет разделять сложные схемы на логические подсистемы. Это особенно важно для крупных проектов, где необходима наглядность и возможность работы на разных уровнях детализации. Например, в проектировании электронных схем иерархия позволяет группировать компоненты по подсистемам, обеспечивая как локальную, так и глобальную оптимизацию.

В контексте автоматизации проектирования схем и диаграмм иерархические модели позволяют:

* упрощать визуализацию. Элементы верхнего уровня могут представлять агрегаты, такие как подсистемы или группы, а элементы нижнего уровня — их детали.
* оптимизировать работу с данными. Переход на более высокий уровень позволяет скрыть незначимые детали и сосредоточиться на общем представлении системы.
* поддерживать модульность. Иерархические подходы облегчают повторное использование элементов и упрощают внесение изменений.

Примером использования иерархических моделей может служить создание UML-диаграмм, где различные уровни иерархии (классы, методы, свойства) описывают архитектуру программного обеспечения. Аналогичный подход применим к проектированию электронных схем, где верхний уровень может представлять схему в целом, а нижние уровни — отдельные компоненты, такие как транзисторы или резисторы [5].

### **3.3 Графовые модели**

Одним из ключевых подходов к автоматизации проектирования схем является использование графовых моделей, позволяющих представлять элементы диаграмм в виде вершин, а связи между ними — в виде рёбер графа. Эти модели упрощают структурирование схем, делая их оптимальными и легко читаемыми. Задача построения и редактирования диаграмм в графовых моделях часто сводится к решению задач размещения и маршрутизации. Графовые модели позволяют применить разнообразные методы, такие как раскраска графов для минимизации пересечений, и алгоритмы поиска кратчайшего пути для оптимизации связей.

Графовые модели имеют несколько ключевых преимуществ:

* гибкость представления данных. Практически любой тип связей и элементов схемы можно описать с помощью графа;
* масштабируемость. Графы позволяют работать с большими и сложными структурами, которые возникают в реальных приложениях;
* интеграция алгоритмов. Использование проверенных алгоритмов из теории графов (например, поиска кратчайшего пути или раскраски графов) делает решения эффективными и удобными для реализации;
* визуальная наглядность. Графы легко интерпретируются визуально, что облегчает понимание структуры схем и процессов;
* универсальность. Графовая модель подходит для множества предметных областей от электротехники до бизнес-процессов и сетевых структур [6].

### **3.4 Методы оптимизации схем: алгоритм раскраски графов, линейное программирование, генетические и эвристические алгоритмы**

### **3.4.1 Алгоритм раскраски графов**

В редакторе схем и диаграмм элементы (например, узлы электрической цепи или блоки) представляют вершины графа, а соединения между ними — рёбра. При наличии связей с несколькими элементами получается многосвязный граф. Дополнительные атрибуты рёбер, такие как вес или направление, учитывают физические или логические характеристики связей.  
Для повышения читаемости применяются алгоритмы раскраски графов, позволяющие минимизировать пересечения и визуальный беспорядок. Например, при проектировании электрических схем раскраска помогает разделить токовые цепи.

Формально, если — граф, где V — множество вершин, а E — множество рёбер, то задача раскраски заключается в нахождении функции , такой что для любого ребра :

Минимальное число цветов необходимое для раскраски графа, называется его хроматическим числом.

Если необходимо провести линии связи между компонентами схемы, это сводится к задаче поиска кратчайшего пути на графе. Алгоритмы Дейкстры или Беллмана-Форда применяются для определения оптимальных маршрутов, учитывая такие параметры, как длина связи или пропускная способность.

Пусть — вес ребра между вершинами и . Задача поиска кратчайшего пути между двумя вершинами и может быть выражена как:

где — путь в графе, соединяющий начальную вершину и конечную вершину , — ребро графа, входящее в путь , — вес ребра, который может представлять расстояние, стоимость или любую другую метрику.

В сложных диаграммах может возникнуть необходимость внесения изменений в режиме реального времени, например, при добавлении новых элементов или связей. Для этого применяются алгоритмы динамической обработки графов, которые позволяют быстро пересчитывать раскраску или маршруты без необходимости полной переработки всей схемы.

Допустим, задача — это визуализировать бизнес-процесс в виде графа. Вершины этого графа будут представлять отдельные процессы или задачи в рамках бизнес-операции, а рёбра — зависимости между ними, например, порядок выполнения задач или их взаимное влияние. Это типичная задача для схем бизнес-процессов, где важно наглядно показать связи между элементами и минимизировать перекрестные линии и сложность восприятия.

На первом этапе важно расположить вершины графа таким образом, чтобы избежать перегрузки визуальной информации и минимизировать перекрытия рёбер. Для этого используется алгоритм force-directed layout. Этот алгоритм моделирует вершины как частицы, между которыми действуют силы: сила отталкивания (аналогичная электростатическим силам) и сила притяжения (аналогичная пружинной силе между вершинами, соединенными ребром).

Алгоритм находит равновесие, в котором минимизируются пересечения рёбер и вершины распределяются по пространству, обычно равномерно. Сначала вершины расположены случайным образом, но по мере выполнения алгоритма они начинают "отталкиваться" друг от друга, пока не находят оптимальное положение. В результате, граф становится менее запутанным и легко воспринимаемым.

На 2 этапе, когда вершины графа уже размещены, необходимо заняться маршрутизацией рёбер. Это означает, что нужно прокладывать линии между вершинами так, чтобы они не пересекались, и их длина была как можно короче. Для этого используется алгоритм Дейкстры, который является эффективным методом поиска кратчайших путей в графе.

Алгоритм Дейкстры находит оптимальные маршруты от одной вершины к другой, минимизируя общую длину рёбер. В контексте схемы бизнес-процесса это может быть полезно, например, для минимизации затрат времени или других ресурсов на выполнение бизнес-процессов, где каждая связь между задачами имеет свою "стоимость" или важность.

На заключительном этапе проводится раскраска графа. Этот шаг помогает выделить важные элементы схемы, такие как критические пути или узлы с высокой нагрузкой. Раскраска может быть использована для выделения процессов, которые влияют на общий результат бизнес-процесса.

Критический путь — это последовательность взаимосвязанных процессов, где каждый процесс зависит от предыдущего. Отклонение в сроках выполнения любого процесса на критическом пути напрямую влияет на сроки выполнения всего проекта. Эти пути могут быть выделены особым цветом или стилем, чтобы акцентировать внимание на их важности.

Узлы с высокой нагрузкой — это такие процессы, которые требуют значительных ресурсов или времени, что делает их узкими местами в бизнес-процессе. Выделение этих узлов позволяет менеджерам сосредоточиться на оптимизации таких процессов и улучшении эффективности.

В результате после всех этих этапов схема бизнес-процесса становится наглядной и оптимизированной. Все элементы связаны по логике работы, а важные и критичные элементы выделены для дальнейшего анализа и возможной оптимизации [7].

### **3.4.2 Линейное программирование**

Для повышения эффективности построения схем также используются линейное программирование и генетические алгоритмы.

Линейное программирование (ЛП) — это математический метод, используемый для оптимизации линейных функций при наличии ограничений, которые также являются линейными. В контексте редакторов схем линейное программирование применяется для задачи оптимизации размещения элементов на рабочем пространстве, где необходимо минимизировать или максимизировать некоторые параметры, такие как пространство, занимаемое элементами, или количество пересечений рёбер.

Предположим, что в редакторе схем имеется несколько объектов, которые должны быть размещены в двумерном пространстве. Пусть и ​ — это координаты элемента на плоскости. Задача заключается в минимизации функции стоимости, например, общей длины рёбер между объектами или количества пересечений рёбер.

Целевая функция может быть представлена в виде:

где ​ — вес (важность) рёбер между элементами и , а — расстояние между этими элементами. Это расстояние рассчитывается по формуле Евклида:

При этом для размещения элементов могут быть наложены различные ограничения. Например, для предотвращения наложения элементов или их выхода за пределы рабочего пространства, ограничения могут быть следующими:

, для всех ,

где и — размеры рабочей области (например, размеры экрана или страницы). Также можно учитывать расстояние между элементами для минимизации перекрытий:

, для всех .

Задача линейного программирования заключается в нахождении значений ​ и , которые минимизируют целевую функцию при соблюдении всех этих ограничений.

Линейное программирование эффективно работает для задач, где количество элементов не слишком велико, а сами ограничения достаточно просты и линейны. Однако при большом числе элементов задача может становиться вычислительно сложной, и тогда переходят к более сложным методам, таким как генетические алгоритмы [8].

### **3.4.3 Генетические алгоритмы**

Генетические алгоритмы (ГА) представляют собой класс эвристических методов оптимизации, вдохновлённых процессом естественного отбора в биологии. Они используются для поиска решений в сложных или многозадачных пространствах, где традиционные методы, такие как линейное программирование, могут быть недостаточно эффективными. Генетические алгоритмы являются подходящими для задач оптимизации размещения элементов на диаграмме, где нет явных аналитических решений или когда пространство поиска слишком велико.

Генетический алгоритм работает через несколько итераций, называемых поколениями, на каждом из которых популяция решений эволюционирует, улучшая свои характеристики. Процесс включает несколько ключевых этапов:

* инициализация популяции. На начальном этапе генерируется случайная популяция возможных решений. Каждое решение представляет собой конфигурацию расположения элементов на рабочем пространстве.
* оценка приспособленности. Для каждого решения вычисляется функция приспособленности (fitness function), которая отражает качество данного расположения. Например, функция приспособленности может учитывать такие факторы, как минимизация длины рёбер и уменьшение числа пересечений.
* скрещивание (Crossover). Два "родительских" решения комбинируются для создания нового решения, называемого "потомком". Этот процесс напоминает биологическое скрещивание, когда наследуются характеристики двух родителей, что позволяет комбинировать сильные стороны разных решений.
* мутация. Существует вероятность, что потомок подвергнется мутации, что означает случайное изменение его характеристик. Мутации могут быть полезны для исследования новых областей пространства решений и предотвращения застревания в локальных оптимумах.
* селекция. Из текущей популяции выбираются лучшие решения, которые будут составлять следующее поколение. Это помогает сохранять решения с высокой приспособленностью, направляя эволюцию популяции к более качественным решениям.
* повторение процесса. Алгоритм повторяет шаги скрещивания, мутации и селекции до тех пор, пока не будет достигнут приемлемый результат или не будет выполнено заданное количество поколений.

Для задачи оптимизации размещения элементов на диаграмме функция приспособленности может быть определена как комбинация нескольких факторов. Например, для минимизации длины рёбер и уменьшения количества перекрытий можно использовать следующую функцию:

где и — веса, определяющие важность соответствующих факторов, — индикатор перекрытия элементов и , который принимает значение 1, если элементы перекрываются, и 0 в противном случае.

Генетический алгоритм будет пытаться минимизировать эту функцию, что позволит найти оптимальное расположение элементов на диаграмме [9].

### **3.4.4 Эвристические алгоритмы**

Эвристические методы играют важную роль в автоматизации проектирования, особенно в задачах, связанных с размещением элементов на графах и диаграммах. В отличие от более сложных математических методов, таких как линейное программирование или генетические алгоритмы, эвристические подходы фокусируются на нахождении хороших, но не обязательно оптимальных, решений за минимальное время. Эти методы идеально подходят для задач, где необходим быстрый результат, и они широко применяются в редакторах схем и диаграмм для упрощения и ускорения процесса создания визуальных представлений информации. Два из наиболее распространённых эвристических методов — это алгоритм поиска ближайшего соседа и жадный алгоритм. Оба метода обеспечивают простоту реализации и высокую скорость.

Метод поиска ближайшего соседа (Nearest Neighbor Search) представляет собой эвристический алгоритм, который используется для последовательного размещения объектов на основе их близости друг к другу. Идея заключается в том, что на каждом шаге выбирается следующий элемент, минимизирующий расстояние до уже размещённых объектов [10].

Формально алгоритм можно описать следующим образом. Пусть и — координаты элемента на плоскости, а — множество всех ещё не размещённых элементов. На каждом шаге алгоритм выбирает элемент, который минимизирует расстояние до уже размещённого:

где — это расстояние между элементом и ближайшим элементом , а — Евклидово расстояние между координатами этих элементов.

В контексте редакторов схем и диаграмм этот метод может быть использован для размещения объектов на рабочем пространстве так, чтобы минимизировать пересечения рёбер. Например, при проектировании электрических схем или графов, элементы (например, узлы или объекты) часто должны располагаться с минимальными пересечениями рёбер, что облегчает восприятие схемы.

Хотя метод поиска ближайшего соседа не гарантирует глобально оптимальное решение, он может эффективно работать в случаях, когда важно быстро получить решение, особенно если количество объектов ограничено. Преимущество метода заключается в его простоте и скорости работы, что делает его удобным инструментом для первичной расстановки элементов.

Жадный алгоритм (Greedy Algorithm) является ещё одним популярным методом, широко используемым в оптимизации и автоматизации проектирования. В отличие от метода поиска ближайшего соседа, который ориентируется на минимизацию расстояний между ближайшими элементами, жадный алгоритм на каждом шаге принимает локально оптимальное решение для всей задачи [11].

Основная идея жадного алгоритма заключается в том, что на каждом шаге выбирается элемент, который улучшает общую цель задачи, не учитывая возможные негативные последствия для других шагов. Например, для задачи минимизации длины рёбер между элементами, жадный алгоритм на каждом шаге выбирает позицию для нового элемента, минимизируя суммарную длину рёбер между ним и уже размещёнными объектами.

Формально это можно записать как:

где и — координаты элементов и , а — расстояние между ними.

Жадный алгоритм может быть использован для решения задач, где требуется минимизация определённых параметров, таких как длина рёбер в графах или перекрытия элементов в диаграммах. Например, в редакторах схем этот метод может помочь уменьшить длину соединений между объектами, что в свою очередь улучшает читаемость и визуальную ясность схемы.

Подобно методу поиска ближайшего соседа, жадный алгоритм не всегда приводит к глобально оптимальному решению, но его главная сила — в простоте и скорости. Он может быть полезен в задачах, где время отклика критично, а необходимость в точном решении не столь велика.

4 Обзор научных исследований по автоматизации визуализации схем и диаграмм

А.Ю. Ткаченко из Бурятского государственного университета представил подход к визуализации графов с использованием принципов объектно-ориентированного программирования. Исследование акцентируется на создании программного компонента, где классы объектов графа реализуют хранение и манипуляцию данными о вершинах и рёбрах, включая их визуализацию. Методы оптимизации, такие как буферизация и отображение только видимой части графа, позволили значительно снизить нагрузку на графическую подсистему, что увеличило производительность системы. Применение абстракции и инкапсуляции обеспечило модульность, а наследование и полиморфизм дали возможность легко расширять функционал компонента. Представленные методы успешно применимы в задачах проектирования сложных графовых структур, таких как сети и процессные диаграммы, минимизируя системные ресурсы и повышая эффективность пользовательского взаимодействия. Модульная структура и оптимизация системы могут применяться для повышения производительности интерфейса редактора диаграмм. Это напрямую связано с разработкой адаптивного редактора, где требуется эффективная обработка больших графов [12].

В свою очередь, И.В. Капустин из Тульского государственного университета в своей работе акцентировал внимание на разработке веб-ориентированных графических редакторов с использованием JavaScript. Основным инструментом разработки выбрана библиотека SigmaJS, позволяющая эффективно строить графы и визуализировать их в веб-приложениях. В работе подробно рассматриваются свойства узлов (например, количество портов и уникальные идентификаторы) и рёбер (такие как пропускная способность и тип среды передачи), которые отображаются в графе. Для улучшения интерактивности графического редактора использованы обработчики событий, такие как двойной клик по узлу или правый клик по рёбру, что позволило реализовать редактирование свойств элементов графа и добавление новых объектов. Его подход позволил внедрить алгоритмы маршрутизации линий связи в браузерных приложениях, обеспечив простоту интеграции таких инструментов в информационные системы. Это решение показало эффективность в области автоматизации проектирования сетевых схем, где требуется высокая динамичность изменений. Результаты этого исследования могут быть использованы для разработки пользовательского интерфейса в редакторе схем и диаграмм [13].

В статье Ю.М. Васильева и Г.М. Фридмана из Санкт-Петербургского государственного экономического университета представлен гибридный эвристический алгоритм, предназначенный для минимизации числа пересечений рёбер в иерархических графах. Исследование сосредоточено на решении задачи оптимального размещения вершин графа на слоях с учётом ограничений по времени вычислений. Для реализации метода применяются подходы, включающие кластеризацию графов по их характеристикам, последовательное использование неточных алгоритмов и подбор оптимальных цепочек таких алгоритмов для каждого кластера. Проведённые расчёты, охватывающие 2000 тестовых графов с различными параметрами, демонстрируют, что предложенный гибридный алгоритм уменьшает число пересечений рёбер на 10.9% по сравнению с лучшими отдельными эвристическими методами. Предложенный гибридный эвристический алгоритм улучшает читаемость диаграмм и повышает их визуальную ясность. Это особенно актуально для автоматизации построения сложных визуальных представлений [14].

Исследователи из Монреальского университета в Канаде разработали инновационный подход к автоматическому построению UML-диаграмм классов из текстовых спецификаций, написанных на естественном языке. В своей работе они использовали большой набор данных из 305 UML-диаграмм, извлеченных из открытого репозитория AtlanMod Zoo. Этот набор данных включал в себя 8706 фрагментов, каждый из которых был аннотирован вручную для создания текстовых описаний классов и отношений. Основной этап их метода включал классификацию предложений текста на описывающие либо классы, либо отношения с использованием алгоритмов машинного обучения. Полученные результаты показывают, что точность метода при строгих критериях остается ограниченной, но подход демонстрирует перспективы в автоматизации проектирования диаграмм. Этот подход не только упрощает процесс генерации диаграмм, но и позволяет сократить время, затрачиваемое на их создание вручную, что особенно важно в условиях работы с большими массивами документации [15].

Исследование, выполненное студентами из Гонконгского университета науки и технологий (Гуанчжоу, Китай), сосредоточено на использовании генеративного искусственного интеллекта (GenAI) для автоматизации визуализации. Авторы обобщили современные подходы к применению генеративных моделей, таких как GAN и диффузионные модели, в задачах визуализации данных. Они выделяют четыре основные стадии процесса: улучшение данных, генерация визуальных отображений, стилизация и интерактивность. Работа также акцентирует внимание на ограничениях современных подходов, таких как нехватка специализированных данных и сложность оценки качества генерируемых визуализаций. Исследование имеет важное значение для дальнейшей интеграции технологий GenAI в средства автоматизации проектирования и визуализации диаграмм [16].

Другое значительное исследование выполнено исследователями из Тунцзийского университета (Шанхай, Китай). В своей работе они проанализировали автоматизацию в процессе создания повествовательных визуализаций, включая использование инструментов, которые автоматически преобразуют структурированные данные в визуальные элементы, такие как диаграммы и схемы. Авторы выделяют уровни автоматизации и описывают, как современные инструменты помогают минимизировать человеческое вмешательство при создании сложных визуализаций, сохраняя при этом их читабельность и точность. Это исследование демонстрирует, как автоматизация может улучшить процессы визуализации в контексте больших массивов данных [17].

Исследование было проведено специалистами из Университета технологии Сиднея (University of Technology Sydney, Австралия), Тяньцзиньского университета (Tianjin University, Китай) и Университета Западного Сиднея (University of Western Sydney, Австралия). Работа посвящена визуализации крупных взвешенных графов, часто используемых для анализа корпоративных связей и структуры баз данных. Основная цель исследования заключалась в разработке гибридного метода, сочетающего кластеризацию узлов и улучшенные алгоритмы визуализации. Для кластеризации использовались методы на основе цепей Маркова (Markov Chain) и алгоритм DPCW, которые эффективно разделяют граф на логические группы. После этого применяется модифицированный алгоритм, основанный на силовом моделировании (force-directed layout), который минимизирует пересечения рёбер и равномерно распределяет узлы. Узлы графа моделируются как частицы с отталкивающимися зарядами, а рёбра — как пружины, притягивающие связанные узлы, что позволяет достичь равновесного состояния. Такой подход упрощает восприятие и анализ больших графов, обеспечивая удобство работы с социальными сетями, маршрутными схемами и сетевыми структурами [18].

Заключение

В ходе выполнения научно-исследовательской работы был проведен анализ существующих моделей, методов и алгоритмов автоматизации проектирования и визуализации схем и диаграмм. Рассмотрены объектно-ориентированные, иерархические и графовые модели, а также алгоритмы, используемые для автоматического размещения и оптимизации схем. Проанализированы линейное программирование, генетические алгоритмы и эвристические методы, применяемые для решения задач минимизации пересечений, маршрутизации связей и улучшения читаемости диаграмм.

Выполненный обзор существующих редакторов схем и диаграмм (Diagrams.net, LucidChart, Microsoft Visio) позволил выявить их преимущества и недостатки, а также определить основные направления для разработки новых решений. В рамках обзора научных исследований рассмотрены современные подходы к автоматизации визуализации, включая использование машинного обучения, генеративных моделей и эвристических методов.

Результаты исследования подтверждают актуальность дальнейшей работы над разработкой методов автоматизации построения схем и диаграмм с улучшенной адаптивностью интерфейса. Основными направлениями дальнейших исследований являются:

* разработка адаптивных алгоритмов размещения элементов диаграмм с учетом удобочитаемости и эргономики;
* интеграция методов машинного обучения для автоматической оптимизации структуры диаграмм;
* создание интеллектуальных пользовательских интерфейсов с возможностью динамической настройки визуального представления;
* исследование способов минимизации временных затрат на обработку больших графовых структур.

Проведенная работа формирует основу для дальнейших исследований в области автоматизации проектирования и визуализации схем, что позволит повысить эффективность работы пользователей с диаграммами, сократить время на их создание и редактирование, а также улучшить визуальное восприятие сложных структур.

Список использованных источников

1. Недостатки и преимущества Diagrams.net [Электронный ресурс], редактировано URL: https://softwarekeep.com/nl‑nl/blogs/comparisons/visio‑vs‑diagrams‑net?\_\_ya\_mt\_enable\_static\_translations=1.
2. Недостатки и преимущества LucidChart [Электронный ресурс], URL: <https://junior3d.ru/article/lucidchart.html?ysclid=lvjjepcdkg260121195>.
3. Недостатки и преимущества Microsoft Visio [Электронный ресурс], URL: <https://studfile.net/preview/4365534/page:3/>.
4. Rumbaugh J., Blaha M. Объектно-ориентированное моделирование и проектирование с использованием UML / пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 688 с.
5. Tan S. X.-D. A General Hierarchical Circuit Modeling and Simulation Algorithm // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2005. – Vol. 24, No. 3. – P. 418–431.
6. Kwon O.-H., Ma K.-L. A Deep Generative Model for Graph Layout // arXiv preprint arXiv:1904.12225. – 2019. – 15 p.
7. Лекция 6. Алгоритмы раскраски графа [Электронный ресурс], URL: https://hpc-education.unn.ru/files/courses/graphs/LEC-Graph-6.pdf.
8. Hussain A., Rahman M. S. Edge Crossing Minimization in Graphs – A Survey // Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology. – 2010. – 18 p. – DOI: 10.13140/RG.2.2.23379.91688.
9. Zhou W., Huang J., Duan Y. Genetic Algorithm for Drawing Directed Acyclic Graphs // Journal of Information and Computational Science. – 2007. – Vol. 4, No. 4. – P. 1119–1125.
10. Yandex Research. Graph-based nearest neighbor search // Yandex Research Blog. – URL: <https://research.yandex.com/blog/graph-based-nearest-neighbor-search>.
11. GeeksforGeeks. Greedy Algorithms // GeeksforGeeks. – URL: <https://www.geeksforgeeks.org/greedy-algorithms/>.
12. Ткаченко А. Ю. Визуализация графа с использованием парадигмы объектно-ориентированного программирования // Вестник Бурятского государственного университета. Математика, информатика. - 2015. - № 1. - c. 33-41.
13. Капустин И.В. Разработка графического редактора сетей средствами языка JavaScript // Теория и практика современной науки. - 2016. - № 9 (15). - c. 176-181.
14. Васильев Ю.М., Фридман Г.М. Визуализация кооперативных схем: гибридный эвристический алгоритм для минимизации количества пересечений ребер при укладке графа // [Известия Санкт-Петербургского государственного экономического университета](https://cyberleninka.ru/journal/n/izvestiya-sankt-peterburgskogo-gosudarstvennogo-ekonomicheskogo-universiteta). - 2017. - №1-2 (103). - c. 87-93.
15. Yang S. Towards Automatically Extracting UML Class Diagrams from Natural Language Specifications, [MODELS '22: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings](https://dl.acm.org/doi/proceedings/10.1145/3550356). Montreal University, Canada, 2022, pp. 396-403.
16. Y. Ye, J. Hao, Y. Hou, Z. Zhan, W. Shishi, X. Yuyu, L. W. Zeng Generative AI for Visualization: State of the Art and Future Directions, journal: Visual Informatics 8(5), 2024.
17. Chen Q., Cao S., Wang J., Cao N. How does automation shape the process of narrative visualization: A survey of tools, Intelligent Big Data Visualization Lab, Tongji University, 2024, pp. 4429-4448, vol. 30.
18. Hua J., Huang M. L., Nguyen Q. V., Drawing large weighted graphs using clustered force-directed algorithm, Proceedings of the 18th International Conference on Information Visualisation, The Hong Kong University of Science and Technology, 2014, pp. 13-17.