УДК 004.021

Обзорная статья

https://doi.org/10.32603/2071-8985-2023-16-10-52-59

Разработка программы построения схемы прокладки кабельных трасс на основе теории графов

Л. А. Федоричев, О. В. Букунова✉

Санкт-Петербургский государственный архитектурно-строительный университет, Санкт-Петербург, Россия

✉ bukunovaolga@yandex.ru

Аннотация. Проанализировано существующее программное обеспечение, используемое при построении схем прокладки кабельных трасс. Сделан вывод, что основная часть программ практически не содержит функционала по оптимизации прокладки трасс. Целью исследования стало создание программы-модуля, основанной на усовершенствованном алгоритме Дейкстры, которую можно использовать в комбинации с существующим программным обеспечением. Результат работы программы заключается в нахождении кратчайшего пути при прокладке кабельных трасс, учитывая логические и физические препятствия. Программа реализована на языке Java и использует модуль concurrency для распараллеливания вычислений, тем самым ускоряя процесс расчета оптимального пути.

Ключевые слова: прокладка кабельных трасс, алгоритм Дейкстры, нахождение кратчайшего пути, распараллеливание вычислений, проектирование кабеле-несущих конструкций, разработка программного обеспечения

Для цитирования: Федоричев Л. А., Букунова О. В. Разработка программы построения схемы прокладки кабельных трасс на основе теории графов // Изв. СПбГЭТУ «ЛЭТИ». 2023. Т. 16, № 10. С. 52–59. doi: 10.32603/2071-8985-2023-16-10-52-59.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Development of a Software Module for Constructing a Scheme of Laying Cable Routes on the Basis of Graph Theory

L. A. Fedorichev, O. V. Bukunova✉

Saint Petersburg State University of Architecture and Civil Engineering, Saint Petersburg, Russia

✉ bukunovaolga@yandex.ru

Abstract. This article analyzes the existing software applications used in the construction of schemes for laying cable routes. It is shown that such applications provide little functionality when optimizing route laying. In this study, we set out to create a software module based on an improved Dijkstra algorithm, which can be used in combination with the existing software. The developed module is capable of finding the shortest path when laying cable routes, taking into account logical and physical obstacles. The software module is implemented in Java and uses the concurrency module to parallelize calculations, thereby speeding up the process of calculating the optimal path.

Keywords: cable route laying, Dijkstra's algorithm, shortest path finding, calculation parallelization, cable-supporting structure design, software development

For citation: Fedorichev L. A., Bukunova O. V. Development of a Software Module for Constructing a Scheme of Laying Cable Routes on the Basis of Graph Theory // LETI Transactions on Electrical Engineering & Computer Science. 2023. Vol. 16, no. 10. P. 52–59. doi: 10.32603/2071-8985-2023-16-10-52-59.

Conflict of interest. The authors declare no conflicts of interest.

Введение. Эффективное энергоснабжение всегда было актуальным вопросом в решении задач строительства. Схемы прокладывания кабелей питания проектируются на самых первых этапах реализации любых проектов. Продолжительное время проектированием кабеленесущих конструкций занимались целые отделы или отдельные группы инженеров. Схема строилась вручную и редактировалась при ошибках или по результатам изменений концепции окончательного проекта.

Анализируя современное состояние в этой области, можно заметить, что технологии проектирования и построения разнообразных схем становятся цифровыми. Из важных аспектов развития можно выделить автоматизацию расчетов и возможность настройки параметров, влияющих на построение, человеком. Основное программное обеспечение, которое способно проводить операции построения кабеленесущих конструкций, предоставляет функционал – обширный, но сильно зависимый от внутренней логики приложения. С другой стороны, самым эффективным способом организации параллельной деятельности по нескольким направлениям всегда была деструктуризация обязанностей согласно принципам четкого разграничения ответственности групп инженеров. Современное программное обеспечение зачастую пренебрегает этими идеями ради увеличения функционала.

Авторами рассмотрено основное программное обеспечение [1], [2] для проектирования кабельных трасс. Результаты анализа представлены в таблице.

Выделим общие недостатки, присущие большинству ПО:

• Не всегда достаточно точности расчетов. Некоторые программы для оптимизации кабельных трасс могут не учитывать все факторы, необходимые для получения более точных результатов.

Анализ программ по проектированию кабельных трасс
Analysis of software applications for cable route design

Программа	Достоинства	Недостатки
PLS-CADD – программное обеспечение, специально разработанное для моделирования и оптимизации кабельных трасс на земле или над землей	Возможность работы с трехмерной моделью, удобный интерфейс	Высокая стоимость, требование высокой скорости процессора компьютера для работы при больших объемах данных
AutoCAD Electrical – программа с широким спектром функций для проектирования и оптимизации кабельных трасс и других электротехнических систем	Широкий спектр функций, простота использования, наличие библиотек уже готовых объектов	Требование больших технических навыков, возможность ошибки при замыкании цепи
ETAP – программное обеспечение для проектирования и анализа электрических систем, в котором есть функциональность для оптимизации кабельных трасс и расчета нагрузки	Возможность точно оптимизировать кабельную трассу, наличие широких функциональных возможностей	Высокая цена, требование опыта работы с программами для проектирования, возможность неверной интерпретации результатов
Bentley Raceway and Cable Management – программное обеспечение, предназначенное для проектирования и оптимизации кабельных трасс, а также других систем управления кабелями	Точность результатов, возможность работы с большим количеством данных	Высокая стоимость, технически сложный интерфейс, важность специализированного обучения
Schneider Electric Unity Pro XL – программное обеспечение для автоматизации процессов и оптимизации кабельных трасс, используемое в промышленности	Большое количество возможностей, оптимизация кабельных трасс, дополнительные функции, такие как программирование PLC	Высокая цена, сложность работы с некоторыми функциями
Caneco BT – программное обеспечение для расчета и оптимизации кабельных трасс и систем безопасности, используемое в энергетике и промышленности	Широкий функционал, точность результатов	Высокая цена, неточность учета дополнительных факторов, сложность использования для новичков

• Ограниченность материалов. Многие программы для оптимизации кабельных трасс ограничены в выборе материалов, что затрудняет оптимизацию и может привести к слишком большим затратам.

• Сложность использования. Некоторые программы для оптимизации кабельных трасс могут быть слишком сложными в использовании, что не позволяет их применять неопытным пользователям.

• Ограниченные возможности. Некоторые программы могут быть ограничены в своих возможностях для оптимизации кабельных трасс, что может повлиять на эффективность и точность расчетов.

• Требование специализированных знаний. Некоторые программы для оптимизации кабельных трасс могут требовать новых знаний, что может быть сложным для новичков в этой области.

На рис. 1 представлены доли, которые программы занимают на рынке ПО.

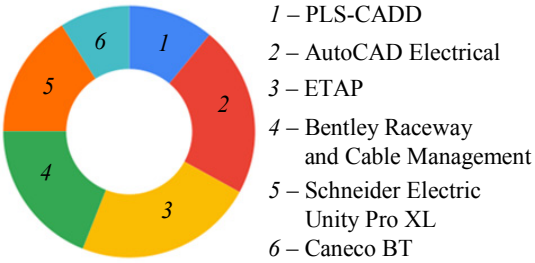

Рис. 1. Пример ошибки модуля позиционирования
Fig. 1. Example of localization module error

У большинства программ, используемых в этой сфере, можно отметить отсутствие встроенной оптимизации при реализации прокладки кабельных трасс. Кроме того, из-за узкой направленности программного обеспечения прослеживается наличие ограничений на используемые материалы, что приводит к отсутствию явного лидера на рынке программ, позволяющих проектировать кабельные трассы.

Очевидна необходимость в универсальной программе, которая позволит по готовым данным предложить оптимальный путь прокладки кабельных трасс. Получившийся результат можно будет перенести в проект.

Цели и задачи исследования. Таким образом, целью исследования стала разработка программного обеспечения, которое по символьной схеме реализует схему прокладки кабеленесущих конструкций. Для выполнения поставленной цели необходимо решить следующие задачи:

– выявить достоинства и недостатки существующего программного обеспечения;

– сформулировать основные аспекты работы программы;

– выбрать и усовершенствовать алгоритм поиска кратчайшего пути при прокладке кабельных трасс;

– разработать новое программное обеспечение.

Требования к программе. Перед автором стояла задача создать программу, которая по функционалу должна:

• Читать данные из базы данных, которые содержат информацию о кабеленесущих конструкций.

• Получать входные данные от пользователя, такие как начальная и конечная точки пути.

• Строить граф с помощью данных о кабеленесущих конструкциях.

• Находить кратчайший путь между начальной и конечной точками.

• Выводить результаты в файл в читаемом формате.

• Быть многопоточной, чтобы ускорить время обработки больших объемов данных.

Для утверждения программы как законченного проекта требовалось провести тестирование следующих аспектов:

– корректности работы и получаемого результата на данных для разработки;

– правильности работы потоков при обработке данных;

– корректности работы программы в различных сценариях.

Основные аспекты работы программы. Одним из преимуществ разработанной программы стал формат самодостаточного модуля, который способен не только самостоятельно выполнять поставленную задачу построения оптимальных кабельных трасс, но и работать как встроенный модуль в любое существующее ПО. Созданная программа способна распараллеливать вычисления и тем самым ускорять процесс оптимизации построения схем кабельных трасс.

Для разработки был выбран язык Java, имеющий следующие преимущества по сравнению с другими языками высокого уровня:

– платформенная независимость: Java-программы могут запускаться на любой платформе без изменений кода. C++ и C# имеют ограничения на определенные платформы;

– очень высокая скорость работы благодаря грамотной оптимизации кода и сборке мусора;

– высокая защита от ошибок программиста благодаря системе безопасности песочницы Java, что позволяет избежать проблем с памятью, таких как утечки;

– наличие большого количества библиотек и фреймворков, что упрощает разработку и экономит время;

– более продвинутая многопоточная обработка, чем в C++ или C#;

– наличие протестированного фреймворка JUnit, который позволяет разработчикам писать автоматические тесты и уверенно выполнять контроль качества кода;

– использование Java для создания высоконадежных приложений и систем крупными игроками в разработке программного обеспечения (например, Google и Oracle) благодаря надежности языка.

Анализ существующих алгоритмов. В ходе исследования проводился анализ основных алгоритмов нахождения кратчайшего пути. Алгоритм Дейкстры позволяет определить наименьший путь от одной вершины к другой в графе со взвешенными ребрами [3]–[5]. В случае решения нашей задачи прокладки трасс, если в графе вершинами являются объекты, а ребрами – расстояния между объектами, то можно найти кратчайший путь между объектами и проложить кабель по этому маршруту.

Использование алгоритма Дейкстры в прокладке кабельной сети подразумевает определение наименьшего пути и прокладку кабеля по этому маршруту. Это позволяет оптимизировать расходы на монтаж, количество используемой кабельной продукции и время на обслуживание кабельной инфраструктуры.

К преимуществам алгоритма можно отнести следующее [6]–[8]:

– универсальность – алгоритм может применяться для различных типов графов, в том числе и взвешенных графов;

– эффективность для поиска кратчайшего пути в графе инструмент, который может быть использован для работы с различными видами графов;

– способность работать с отрицательными весами при условии, что нет отрицательных циклов в графе.

Анализ алгоритма Беллмана–Форда показал [3], что он позволяет обрабатывать графы с отрицательными весами ребер, в то время как алгоритм Дейкстры применим только для ребер с положительными весами. Однако алгоритм Беллмана–Форда имеет большую асимптотическую сложность времени – O(N^2) по сравнению с O(Nlog(N)) для алгоритма Дейкстры. Поэтому, если граф не содержит отрицательных ребер или их количество невелико, алгоритм Дейкстры предпочтительнее. Однако если в графе присутствуют отрицательные ребра, то алгоритм Беллмана–Форда – единственный способ нахождения кратчайшего пути, даже если итоговый путь проходит через несколько ребер отрицательного веса.

Важно отметить, что граф для поиска можно составить без отрицательных ребер, поэтому алгоритм Дейкстры выгоднее по следующим причинам:

– эффективное потребление памяти – алгоритм Дейкстры использует очередь с приоритетом для хранения вершин, которые еще не были посещены, в то время как алгоритм Беллмана–Форда хранит весь граф в матрице смежности;

– легкость реализации;

– оптимальность поиска расстояния между двумя вершинами – алгоритм Дейкстры всегда находит кратчайший путь до каждой вершины из начальной вершины, тогда как алгоритм Беллмана–Форда может зациклиться и не дать правильных результатов [9]–[11].

Доработка алгоритма Дейкстры. В процессе прокладки кабеля могут возникать препятствия, которые могут повлиять на длину маршрута и количество необходимого кабеля. Такими препятствиями могут быть, например, стены, границы кадастровых участков или другие кабельные конструкции. В таких случаях обычно используют другие методы оптимизации маршрутов, например алгоритм A* или обход препятствий с помощью кода цвета.

Важно отметить, что алгоритм Дейкстры лучше, чем алгоритм A*, при использовании их в построении схемы кабеленесущих конструкций, поскольку в данном случае расстояния между узлами графа рассчитываются на основе реальных физических расстояний, а не эвристических значений, как в случае с алгоритмом A*. Кроме того, алгоритм Дейкстры работает быстрее, если граф имеет ограниченное количество вершин, а граф маршрутов кабеленесущих конструкций обычно имеет небольшое количество вершин. Если же граф имеет большое количество вершин, то использование алгоритма A* может быть более эффективным [12]–[14].

Обход препятствий с помощью кода цвета – эффективный и многообещающий метод для нахождения кратчайшего пути прокладки кабеленесущих конструкций. Он позволяет учитывать типы препятствий и находить наилучший маршрут, минуя непреодолимые препятствия. Однако для того, чтобы применять этот метод успешно,

необходимо четко определить цвета и их значения для каждого типа препятствий.

Одним из возможных способов, который позволит избежать использования кода цвета для обхода препятствий, стала доработка алгоритма Дейкстры, которая позволит ввести понятие «проходимости» для каждой вершины графа.

В данном случае вместо использования цветов для обозначения «проходимости» каждой вершины задавались числовые значения, которые означали степень проходимости данной вершины. Например, для вершины с очень узким проходом (или с большим количеством препятствий на пути) мы установили меньшее значение проходимости, а для вершин с большим пространством для прохода мы задали большее значение [15].

Таким образом, при поиске кратчайшего пути в алгоритме Дейкстры учитывались не только длина ребер между вершинами, но и их проходимость. Т. е. для перемещения с одной вершины на другую учитывалось не только расстояние между ними, но и стоимость прохождения каждой из них.

Такой подход позволил избежать использования цветов для обхода препятствий, стал универсальным, поскольку способен учитывать не только конкретные типы препятствий, но и другие факторы, влияющие на проходимость вершин (например, тип поверхности, скорость перемещения по ней и т. п.).

Реализация программы. Для написания многопоточной программы на языке Java для построения пути кабеленесущей конструкции в здании по алгоритму Дейкстры были описаны следующие абстракции:

1. Одномерный массив из двух элементов для представления вершины в графе, который хранит ее координаты и стоимость прохождения.

2. Класс для представления пары начальной и конечной точек:

```
class Pair{
   long[] start;
   long[] end;

   prevX() – возвращает точку слева от start
   nextX() – возвращает точку справа от start
   prevY() – возвращает точку снизу от start
   nextY() – возвращает точку сверху от start
}
```

3. Класс для представления пути, который содержит список вершин, составляющих маршрут и длину маршрута:

```
class Way{
   List<Long[]> points;
   long length;

   min(... Way[] ways) – возвращает путь с минимальной длинной
   add(long[] start) – добавляет точку в список и увеличивает длину пути
}
```

4. Класс для выполнения алгоритма Дейкстры, который принимает граф, начальную и конечную вершины и вычисляет кратчайший путь между ними.

5. Класс для чтения символьной схемы помещения из файла, который возвращает двумерный массив символов, где каждый символ представляет ячейку в помещении.

6. Класс для отображения результирующей символьной схемы, который принимает массив символов и маршрут и добавляет символы для отображения маршрута.

7. Класс для построения и выполнения потоков, который создает поток для каждой вершины графа, а каждый поток будет выполнять алгоритм Дейкстры от соответствующей вершины до всех остальных.

Для наглядности автором составлена блок-схема работы основной программы (рис. 2).

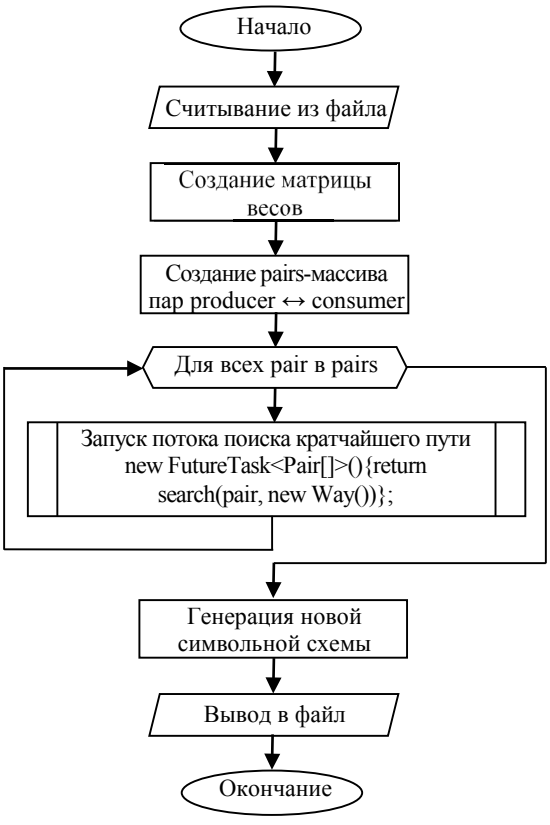

Рис. 2. Схема основной программы
Fig. 2. Scheme of the main program

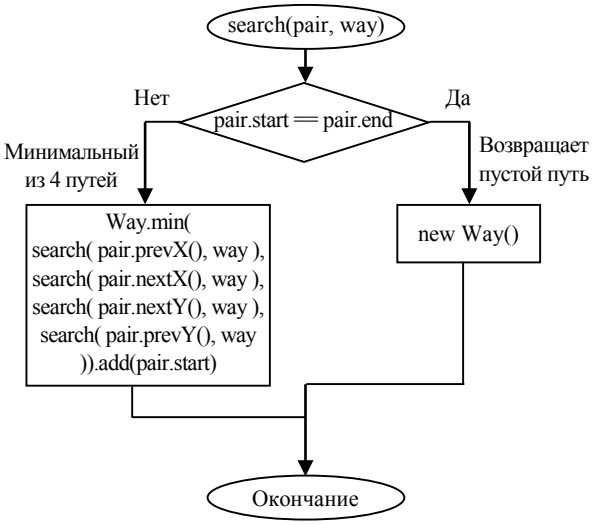

Рис. 3. Схема работы отдельного потока
Fig. 3. Working scheme of a separate thread

Также составлена схема работы отдельного потока поиска кратчайшего пути (рис. 3).

Пример работы программы при наличии препятствий и условии, что от одного источника могут питаться максимум три потребителя.

На входе программа принимает размер поля и символьную схему (рис. 4):

```
12 11
X X X X X X X X X X X X
X O O O C X C O O O O X
X E O O O X O O O O O X
X O O O O O O O O O C X
X O O O O O O O O O O X
X X X X X X O X X X X X
X E O O O O O O O O O X
X O O O O X O O O O O X
X T T O O X O O O O C X
X O O O C X O O O O O X
X X X X X X X X X X X X
```

Рис. 4. Входные данные
Fig. 4. Input data

На рис. 4 приняты следующие обозначения: X – элемент, к которому можно закрепить кабель; E – местоположение источника; C – точки потребления; T – препятствие, через которое невозможно провести кабель.

Данные на выходе (рис. 5):

```
X X X X X X X X X X X X
X S S S C X C O O O O X
X E O O S X S O O O O X
X O O O S S S O O O C X
X O O O O O S S S S S X
X X X X X X S X X X X X
X E S S S S S S S S S X
X O O O S X O O O O S X
X T T O S X O O O O C X
X O O O C X O O O O O X
X X X X X X X X X X X X
```

Рис. 5. Выходные данные
(S – элемент оптимального маршрута)
Fig. 5. Output (S – element of the optimal route)

Результаты. Проведен анализ существующего программного обеспечения, выявлены его недостатки и учтены преимущества. Сформулирована необходимость разработки и требования, которым должна соответствовать новая программа.

Разработан алгоритм, позволяющий проложить оптимальный путь от источника к потребителю, учитывая логические и физические ограничения.

Реализована программа, позволяющая по символьной схеме предложить оптимальную схему кабельных трасс на территории. Вычисления программы распараллелены на несколько потоков.

Проведены тесты работы программы на разных объемах данных. Результаты представлены на рис. 6.

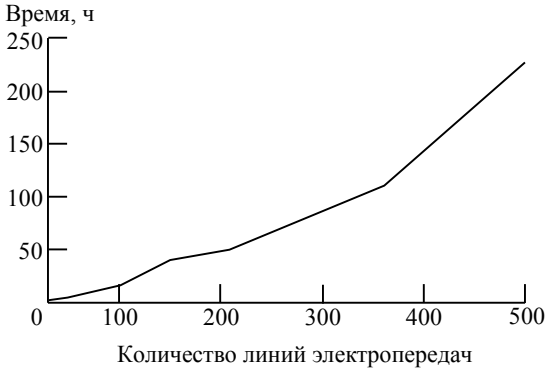

Рис. 6. Рост времени расчета относительно увеличения количества линий электропередач
Fig. 6. Growth of calculation time in relation to an increase in the number of transmission l

Также было проведено сравнение работы созданного модуля с работой программы ETAP на одинаковых объемах данных на рис. 7, где разработанная программа обозначена непрерывной линией, а ETAP – штриховой.

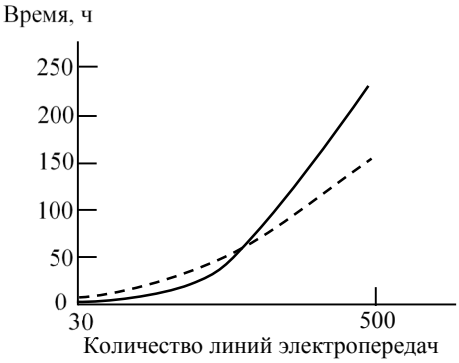

Рис. 7. Время, потраченное на произведение расчетов, по разработанной программе и по ETAP
Fig. 7. Time spent on making calculations, according to the developed program and according to ETAP

Следует отметить, что разработанная программа на меньших объемах данных (<360) быстрее получает решение.

Заключение. Существующее программное обеспечение, используемое при построении схем прокладки кабелей, почти не содержит функционала, позволяющего оптимизировать маршруты прокладки кабелей. Разработанная программа содержит функционал, которого не хватает существующим программам [16], и имеет удобный тип принимаемых и возвращаемых данных, подходящий для реализации импорта и экспорта. В перспективе программа может быть встроена как модуль в другое программное обеспечение, в котором не хватает функционала по оптимизации кабельных трасс.

Список литературы

1. Использование программно-аппаратного комплекса диспетчеризации на базе платформ Etap Real Time и PSI Control для моделирования режимов и управления в электроэнергетических системах / А. В. Кушнир, А. В. Белоусов, А. А. Виноградов, П. В. Рощубкин // Междунар. науч.-исслед. журн. 2015. № 11-2 (42). С. 50–52.

2. Корнилова Н. А. Сравнительная характеристика программных комплексов «Etap» и «Energycs», используемых для анализа, проектирования и тестирования энергосистем // Междунар. науч.-техн. конф. молодых ученых БГТУ им. В. Г. Шухова. Белгород, 2015. С. 3255–3260.

3. Барыбин Д. А., Кофман Е. Ю., Шульгин М. С. Сравнение алгоритмов Дейкстры и Беллмана–Форда при решении задачи о поиске кратчайшего пути в протоколах маршрутизации // Символ науки. 2021. № 6. С. 27–31.

4. Берж К. Теория алгоритмов и ее применение. М.: Мир, 1980. 320 с.

5. Зыков А. А. Теория конечных графов. Новосибирск: Наука, 1969. 554 с.

6. Изотова Т. Ю. Обзор алгоритмов поиска кратчайшего пути в графе // Новые информационные технологии в автоматизированных системах: материалы девятнадцатого науч.-практ. семинара. М.: ИПМ им. М. В. Келдыша, 2016. С. 341–344.

7. Алгоритмы: построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. М.: Вильямс, 2006. 1328 с.

8. Султанова А. Б. Сравнительный анализ алгоритмов поиска оптимального пути // Бюл. науки и практики. 2020. Т. 6, № 12. С. 248–255. doi: 10.33619/2414-2948/61/25.

9. Peri V. M., Simon D. Fuzzy logic control for an autonomous robot // Conf.: Fuzzy Information Proc. Society (NAFIPS' 2005). IEEE. 2005. P. 337342. doi: 10.1109/NAFIPS.2005.1548558.

10. Anytime Nonparametric A* / J. Van Den Berg, R. Shah, A. Huang, K. Goldberg // Proc. of the 25th AAAI Conf. on Artificial Intelligence. San Francisco, 2011. Vol. 25(1). P. 105–111. doi: 10.1609/aaai.v25i1.7819.

11. Multi-agent path finding for self interested agents / Z. Bnaya, R. Stern, A. Felner, R. Zivan, S. Okamoto // 6th Ann. Symp. on Combinatorial Search. Washington, 2013. P. 38–46. doi: 10.1609/socs.v4i1.18292.

12. Harabor D. D., Grastien A. Improving jump point search. ICAPS. // Proc. of the Intern. Conf. on Automated Planning and Scheduling. Portsmouth, 2014. P. 128–135.

13. Uras T., Koenig S., Hernández C. Subgoal graphs for optimal pathfinding in eight-neighbor grids // 23rd Intern. Conf. on Automated Planning and Scheduling. Rome, 2013. P. 224–232.

14. Comparison of different grid abstractions for pathfinding on maps / Y. Bjornsson, M. Enzenberger, R. Holte, J. Schaeffer, P. Yap // Intern. Joint Conf. on Artificial Intelligence (IJCAI). Acapulco, 2003. P. 1511–1512.

15. Grid task scheduling: algorithm review / T. Ma, Q. Yan, W. Liu, D. Guan, S. Lee // IETE Technical Rev. 2011. Vol. 28, no. 2. P. 158–167.

16. Java Concurrency на практике / Б. Гетц, Т. Пайес, Д. Блох, Д. Боубер, Д. Холмс, Д. Ли. СПб.: Питер, 2023. 464 с.

Информация об авторах

Федоричев Лев Александрович – студент СПбГАСУ, ул. 2-я Красноармейская, 4, Санкт-Петербург, 190005, Россия.
E-mail: fedorichev16@gmail.com

Букунова Ольга Викторовна – канд. техн. наук, доцент СПбГАСУ, ул. 2-я Красноармейская, 4, Санкт-Петербург, 190005, Россия.
E-mail: bukunovaolga@yandex.ru
https://orcid.org/0000-0002-5721-1795

References

1. Ispol'zovanie programmno-apparatnogo kompleksa dispetcherizacii na baze platform Etap Real Time i PSI Control dlja modelirovanija rezhimov i upravlenija v jelektrojenergeticheskih sistemah / A. V. Kushnir, A. V. Belousov, A. A. Vinogradov, P. V. Roshhubkin // Mezhdunar. nauch.-issled. zhurn. 2015. № 11-2 (42). S. 50–52. (In Russ.).

2. Kornilova N. A. Sravnitel'naja harakteristika programmnyh kompleksov «Etap» i «Energycs», ispol'zuemyh dlja

analiza, proektirovanija i testirovanija jenergosistem // Mezhdunar. nauch.-tehn. konf. molodyh uchenyh BGTU im. V. G. Shuhova. Belgorod, 2015. S. 3255–3260. (In Russ.).

3. Barybin D. A., Kofman E. Ju., Shul'gin M. S. Sravnenie algoritmov Dejkstry i Bellmana–Forda pri reshenii zadachi o poiske kratchajshego puti v protokolah marshrutizacii // Simvol nauki. 2021. № 6. S. 27–31. (In Russ.).

4. Berzh K. Teorija algoritmov i ejo primenenie. M.: Mir, 1980. 320 s. (In Russ.).

5. Zykov A. A. Teorija konechnyh grafov. Novosibirsk: Nauka, 1969. 554 s. (In Russ.).

6. Izotova T. Ju. Obzor algoritmov poiska kratchajshego puti v grafe // Novye informacionnye tehnologii v avtomatizirovannyh sistemah: materialy devjatnadcatogo nauch.-prakt. seminara. M.: IPM im. M. V. Keldysha, 2016. S. 341–344. (In Russ.).

7. Algoritmy: postroenie i analiz / T. H. Kormen, Ch. I. Lejzerson, R. L. Rivest, K. Shtajn. M.: Vil'jams, 2006. 1328 s. (In Russ.).

8. Sultanova A. B. Sravnitel'nyj analiz algoritmov poiska optimal'nogo puti // Bjul. nauki i praktiki. 2020. T. 6, № 12. S. 248–255. doi: 10.33619/2414-2948/61/25.

9. Peri V. M., Simon D. Fuzzy logic control for an autonomous robot // Conf.: Fuzzy Information Proc. Society (NAFIPS' 2005). IEEE, 2005. P. 337342. doi: 10.1109/NAFIPS.2005.1548558. (In Russ.).

10. Anytime Nonparametric A* / J. Van Den Berg, R. Shah, A. Huang, K. Goldberg // Proc. of the 25th AAAI Conf. on Artificial Intelligence. San Francisco, 2011. Vol. 25(1). P. 105–111. doi: 10.1609/aaai.v25i1.7819.

11. Multi-agent path finding for self interested agents / Z. Bnaya, R. Stern, A. Felner, R. Zivan, S. Okamoto // 6th Ann. Symp. on Combinatorial Search. Washington, 2013. P. 38–46. doi: 10.1609/socs.v4i1.18292.

12. Harabor D. D., Grastien A. Improving jump point search. ICAPS // Proc. of the Intern. Conf. on Automated Planning and Scheduling. Portsmouth, 2014. P. 128–135.

13. Uras T., Koenig S., Hernández C. Subgoal graphs for optimal pathfinding in eight-neighbor grids // 23rd Intern. Conf. on Automated Planning and Scheduling. Rome, 2013. P. 224–232.

14. Comparison of different grid abstractions for pathfinding on maps / Y. Bjornsson, M. Enzenberger, R. Holte, J. Schaeffer, P. Yap // Intern. Joint Conf. on Artificial Intelligence (IJCAI). Acapulco, 2003. P. 1511–1512.

15. Grid task scheduling: algorithm review / T. Ma, Q. Yan, W. Liu, D. Guan, S. Lee // IETE Technical Rev. 2011. Vol. 28, no. 2. P. 158–167.

16. Java Concurrency na praktike / B. Getc, T. Pajes, D. Bloh, D. Bouber, D. Holms, D. Li. SPb.: Piter, 2023. 464 s. (In Russ.).

Information about the authors

Lev A. Fedorichev – student of Saint-Petersburg State University of Architecture and Civil Engineering, 2-ya Krasnoarmeyskaya st., 4, Saint Petersburg, 190005, Russia.
E-mail: fedorichev16@gmail.com

Olga V. Bukunova – Cand. Sci. (Eng.), Assistant Professor of Saint-Petersburg State University of Architecture and Civil Engineering, 2-ya Krasnoarmeyskaya st., 4, Saint Petersburg, 190005, Russia.
E-mail: bukunovaolga@yandex.ru
https://orcid.org/0000-0002-5721-1795

Статья поступила в редакцию 06.07.2023; принята к публикации после рецензирования 23.10.2023; опубликована онлайн 19.12.2023.

Submitted 06.07.2023; accepted 23.10.2023; published online 19.12.2023.