**© 2024 г. А.Г. СОРОКА (andrew.soroka@student.msu.ru),**
**Г.В. МИХЕЛЬСОН (mikhelson.g@gmail.com)**
**(Московский государственный университет им. М.В. Ломоносова),**
**А.В. МЕЩЕРЯКОВ, канд. физ.-мат. наук (mesch@cosmos.ru)**
**(Московский государственный университет им. М.В. Ломоносова;**
**Институт космических исследований РАН),**
**С.В. ГЕРАСИМОВ (sergun@gmail.com)**
**(Московский государственный университет им. М.В. Ломоносова)**

# SMART ROUTES: СИСТЕМА ДЛЯ РАЗРАБОТКИ И СРАВНЕНИЯ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ ОПТИМИЗАЦИИ МАРШРУТОВ С РЕАЛИСТИЧНЫМИ ОГРАНИЧЕНИЯМИ

Задача оптимизации маршрутов с реалистичными ограничениями становится крайне актуальной в условиях глобального роста городского населения. Подходы к оптимизации маршрутов, включая точные методы, сталкиваются с проблемой экспоненциальной сложности при увеличении размера задачи оптимизации маршрутов. В работе сравнивается точный решатель SCIP с эвристическими методами (LKH, 2-OPT, 3-OPT, OR-Tools) и моделью глубокого обучения JAMPR+. Для задач размером 50 глубокое обучение и классические эвристики достигают точности, сравнимой с SCIP, но требуют меньше времени. Для задач размером 100, эвристики и нейронные сети значительно опережают SCIP как по времени, так и по качеству первого найденного решения. Для проведения экспериментов разработана платформа Smart Routes для решения задачи оптимизации маршрутов, которая включает в себя точные, эвристические и нейросетевые модели и облегчает удобное интегрирование собственных алгоритмов и наборов данных.

*Ключевые слова*: CVRPTW, система оптимизации маршрутов, эвристики, точные решения, глубокое обучение с подкреплением.

**DOI:** 10.31857/S0005231024030083, **EDN:** TOTOOH

## 1. Введение

Задача оптимизации маршрутов (***Vehicle Routing Problem, VRP***) — это класс задач логистики, направленных на минимизацию затрат на транспортные ресурсы, стоимость маршрута и времени доставки груза группе клиентов. В условиях реального мира оптимизация маршрутов является актуальной проблемой для большинства компаний. По мере увеличения числа городов и клиентов, становится необходимым разработать решение, которое позволит оптимально использовать выделенные ресурсы при сохранении качества услуг. Чем короче маршрут, тем быстрее клиент может получить свой товар и тем больше времени остается для доставки товаров другим клиентам.

В логистических задачах зачастую *VRP* рассматривают с различными ограничениями, которые необходимо учитывать при построении маршрута. Наиболее популярными из них являются ограничения на *время посещения и обслуживания клиентов* (TW) и *вместимость транспортных средств* (C). На сегодняшний день возникают большие трудности при разработке и модификации алгоритмов, способных решать задачи с увеличивающимися размерностями и при этом соблюдающих компромисс между качеством решений и временем их поиска в зависимости от количества наложенных ограничений на задачу. Помимо этого, не существует единой платформы, которая позволяла бы пользователям не только решать задачи с помощью встроенных алгоритмов и экспериментировать с ними, но и добавлять собственные алгоритмы, методы чтения и обработки данных и т.п. без изменения архитектуры системы.

Вклад данной статьи заключается в следующем: 1) платформа для оптимизации задач маршрутизации; 2) сравнительный анализ эффективности распространенных методов оптимизации маршрутов. Усилия авторов сосредоточены на разработке инновационной системы под названием Smart Routes и подробном сравнительном анализе различных методов в контексте описанных задач и ограничений. Изучены эвристики, глубокое обучение с подкреплением и точные методы на базе библиотеки (SCIP). Исследование завершается полезной информацией об эффективности и качестве этих решений, в первую очередь оцениваемых двумя основными метриками: временем оптимизации решения и окончательной стоимостью маршрута, определенной целевой функцией. В статье раскрывается архитектура системы Smart Routes. Система без проблем интегрирует точные и эвристические подходы, а также подходы глубокого обучения с подкреплением, разработанные для решения проблемы маршрутизации транспортных средств (VRP) в различных формах и модификациях. Она полезна как экспертам в области логистики, так и тем, кто менее опытен, предоставляя единое место для тестирования новых идей. Следовательно, данное исследование не только углубляется в поиск наиболее эффективного алгоритмического подхода к решению проблемы маршрутизации транспортных средств с временными окнами (CVRPTW) — подкатегории VRP с популярными и практическими ограничениями, но также представляет инновационный инструмент, готовый решать эту проблему всесторонне и эффективно.

Статья устроена следующим образом: раздел 2 предоставляет обзор последних публикаций, содержащих как классический, так и глубокий подход к оптимизации маршрутов с использованием обучения с подкреплением; разделы 3 и 4 содержат описания моделей и разработанной системы Smart Routes соответственно; раздел 5 описывает данные, на основе которых проводились эксперименты; в разделе 6 представлены полученные результаты, а в разделе 7 содержатся выводы, сделанные в ходе проведенного исследования.

## 2. Обзор литературы

В качестве обзора рассмотрены три группы алгоритмов, способных решать рассматриваемую задачу. Отметим, что, несмотря на богатую историю данной проблемы, существует много исследований, сравнивающих различные подходы к решению задачи оптимизации маршрута. Большинство статей часто сталкиваются с отсутствием точных методов в сравнении, а также ограниченным размером рассматриваемых задач и, иногда, недостатком ограничений. Здесь постараемся охватить все требования, чтобы предоставить полную картину применимости классических подходов.

### *2.1. Эвристические алгоритмы*

***Эвристические алгоритмы*** включают *конструктивные эвристики* и *метаэвристики*, предоставляя субоптимальные решения для VRP. **Конструктивные эвристики** пошагово создают решения, например *эвристика ближайшего соседа* и *эвристика вставки*. Они обеспечивают быстрое формирование качественных решений. **Метаэвристические алгоритмы**, такие как *имитация отжига* и *генетические алгоритмы*, исследуют пространство решений с использованием рандомизации. Они могут находить более качественные решения, но требуют больше времени на вычисления.

Использование этих алгоритмов зависит от задачи. Конструктивные эвристики применяются благодаря их скорости и эффективности в получении быстрых качественных решений. Метаэвристики более гибки, часто используются в широком спектре задач, но могут требовать больше времени на вычисления.

### *2.2. Точные алгоритмы*

Алгоритмы, которые предоставляют гарантированно оптимальное решение, — это **точные алгоритмы**. К ним относятся алгоритмы линейного программирования. Они решают задачи, в которых функции цели и ограничений являются линейными. Задачи линейного программирования хорошо изучены, и известны их свойства с точки зрения существования решения.

*Задача линейного программирования* (***LP***) — это задача оптимизации в стандартной форме, выражаемая соотношением

$$\max\left\{c^{\mathrm{T}}x \mid Ax \leqslant b, x \geqslant 0\right\}, \tag{1}$$

где $A \in R^{m,n}$ – технологическая матрица, $b \in R^m$ – вектор ресурсов, $c \in R^n$ – вектор цен, $x \in R^n$ – неизвестный вектор. Если некоторые или все переменные в векторе $\boldsymbol{x}$ являются целыми числами, проблема называется *проблемой смешанного целочисленного линейного программирования* (***Mixed Integer Linear Programming, MILP***). Существует несколько алгоритмов, которые могут решать задачи *MILP*:

1. *Метод ветвей и границ* (*Branch&Bound* [1]) разделяет задачу на подзадачи (узлы) и решает их с использованием линейного программирования. Если решение не целочисленное, алгоритм разветвляет узел и решает подзадачи, добавляя ограничения. Процесс повторяется до нахождения целочисленного решения или доказательства его отсутствия.
2. *Метод секущей плоскости* (*Cutting Plane* [2]) добавляет к задаче линейные неравенства, называемые разрезами, чтобы исключить нецелые решения. Разрезы генерируются путем решения $LP$-релаксации задачи. Алгоритм продолжается до нахождения целочисленного решения или доказательства неосуществимости.
3. *Метод ветвей и отсечений* (*Branch&Cut* [3]) комбинирует *Branch & Bound* и *Cutting Plane*. Генерируются разрезы для устранения нецелых решений, и одновременно разветвляются узлы. Этот метод часто более эффективен, чем отдельно взятые *Branch & Bound* или *Cutting Plane*.

В целом выбор алгоритма зависит от конкретного экземпляра задачи, требований к качеству решения и времени его поиска. Некоторые решатели *MILP*, такие как *CPLEX* [4], *SCIP* [5] и *Gurobi* [6], реализуют несколько из этих алгоритмов и автоматически выбирают наиболее подходящий из них для данного экземпляра задачи.

Основная трудность с задачами линейного программирования заключается в их размерности, так как могут быть тысячи переменных и ограничений. Объем памяти и время решения могут увеличиваться экспоненциально по мере добавления целочисленных переменных. Поэтому специально для того, чтобы находить приближенное к оптимальному решение за меньшее время, были разработаны эвристики. Для сложных проблем эвристические подходы часто могут предложить лучший компромисс между качеством решения и вычислительным временем.

### *2.3. Алгоритмы глубокого обучения и обучения с подкреплением*

Первая модель глубокого обучения для решения VRP была предложена в [7], где был адаптирован Pointer Network (PtrNet) из [8] для работы с CVRP. В [7] полностью отказались от исходной части модели кодировщика RNN и заменили ее линейным слоем с общими параметрами. Более новый алгоритм AM в [9] заменил эту архитектуру адаптированной моделью трансформера с использованием внимания [10]. Прямым улучшением этой модели является подход JAMPR, предложенный в [11], где авторы добавили дополнительные полносвязные сети для текущего пути и положения грузовиков. Это дополнение позволило алгоритму успешно решить проблему CVRPTW. В [12] предлагают подход к улучшению на основе RL, который итеративно выбирает область на графике, а затем выбирает и применяет установленные локальные эвристики. Этот подход был дополнительно улучшен оператором разрушения, представленным в [13]. Последняя попытка использовать глубокое

обучение для разделения набора точек на подзадачи и решения их с помощью решателя черного ящика была предложена в [14]. Авторы предложили два подхода: регрессионное прогнозирование возможного улучшения конечной стоимости и классификацию на лучшую подзадачу. За счет уменьшения размерности и использования классических метаэвристических подходов в каждой подзадаче авторам удалось показать хорошие результаты на задачах большой размерности (более 1000 точек). Была подробно рассмотрена и исследована применимость подхода на базе глубокого обучения с подкреплением для решения задачи оптимизации маршрута с реалистичными ограничениями и продемонстрировано, что быстрое субоптимальное решение может быть получено с использованием обученной нейронной эвристики, как показано в статье авторов [15].

## 3. Алгоритмы и модели

### *3.1. Классические подходы*

При рассмотрении классических эвристических подходов было принято решение сосредоточиться на группе алгоритмов, известных как *локальный поиск*. Эти алгоритмы обеспечивают хороший баланс между качеством решения и эффективностью поиска в реальных задачах и также служат фундаментальными компонентами для метаэвристик и генетических алгоритмов [16].

Основным эвристическим алгоритмом был выбран один из лучших эвристических подходов — *эвристика Лина–Кернигана* (**Lin–Kernighan heuristic**) [17]. Данный алгоритм относится к классу так называемых алгоритмов локальной оптимизации. Алгоритм задается в терминах *opt* (обмен или ходы), которые могут преобразовать один маршрут в другой. При наличии осуществимого маршрута алгоритм неоднократно выполняет обмены, которые сокращают продолжительность текущего маршрута, пока не будет достигнут маршрут, для которого никакой обмен не приведет к улучшению. Этот процесс может повторяться много раз с начальных маршрутов, сгенерированных каким-либо рандомизированным способом. В алгоритме предполагается, что некоторое начальное разбиение маршрута уже существует, затем имеющееся приближение улучшается в течение некоторого количества итераций. Применяемый способ улучшения состоит в обмене вершинами в каждом подмаршруте в отдельности.

SCIP [5], программное обеспечение с открытым исходным кодом, представляет собой мощный инструмент оптимизации, специально разработанный для решения задач смешанного целочисленного программирования [18, 19]. Он предназначен для решения сложных задач оптимизации, возникающих, например, в области логистики, маршрутизации и планирования производства. SCIP использует широкий спектр методов и алгоритмов оптимизации, включая методы Branch&Bound, методы секущей плоскости, методы распространения ограничений, эвристики, методы декомпозиции и целочисленное про-

граммирование. В целом *SCIP* представляет собой передовое программное обеспечение для оптимизации, использующее широкий спектр алгоритмов и методов для эффективного и точного решения сложных задач оптимизации.

В дополнение к SCIP был выбран OR-Tools (Operations Research Tools) [20], популярный фреймворк, заслуживший признание своей эффективностью в решении задач VRP (маршрутизации транспортных средств). OR-Tools — это универсальное программное обеспечение, предназначенное для решения комбинаторных задач оптимизации, уравнений и неравенств, а также задач планирования и маршрутизации. Он предлагает разнообразный набор методов и алгоритмов оптимизации, включая *линейное программирование (LP)*, *целочисленное программирование (IP)*, *методы дискретной оптимизации* и *методы решения уравнений и неравенств*. OR-Tools представляет собой мощный инструмент для решения различных задач оптимизации. Благодаря своей гибкости и широкому набору методов, OR-Tools может быть использован для решения разнообразных задач в областях логистики, планирования, транспорта и др.

### *3.2. Модели глубокого обучения с подкреплением*

Модифицировав модель JAMPR [11], используем ее в качестве подхода глубокого обучения с подкреплением [15]. Модель JAMPR является модификацией модели внимания (AM) [9], которая использует архитектуру энкодер–декодер на основе внимания [10]. Обе модели рассматривают проблему оптимизации маршрута как задачу последовательного принятия решений, моделируемую как марковский процесс принятия решений и решаемую с использованием обучения с подкреплением. Решение проблемы строится поэтапно, создавая маршруты по одному узлу за раз. Текущее решение, маршрут и непосещенные узлы рассматриваются как состояние, а индекс всех непосещенных узлов, доступных для добавления к текущему маршруту, рассматривается как действие.

К подходу JAMPR были добавлены обучаемые матрицы (блок M+ на рис. 1), отдельные для каждого ограничения. Они применяются к результату работы декодера, модифицируя тем самым политику ($\pi_\theta$). Окончательным результатом работы сети является измененная политика размером ($k*$, $n*$). Процесс оптимизации заключается в минимизации стоимости маршрута с учетом пропущенных клиентов (мягкая постановка задачи):

$$Q = \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} x_{ij} + \lambda \sum_{i=1}^{n} z_i, \tag{2}$$

где $i$, $i = 1, \ldots, N$ — клиенты, $j$, $j = 1, \ldots, M$ — транспортные средства, $c_{ij}$ — стоимость маршрута от $j$ транспортного средства к $i$ клиенту, $z_i$ — бинарный параметр, указывающий, пропущен ли клиент.

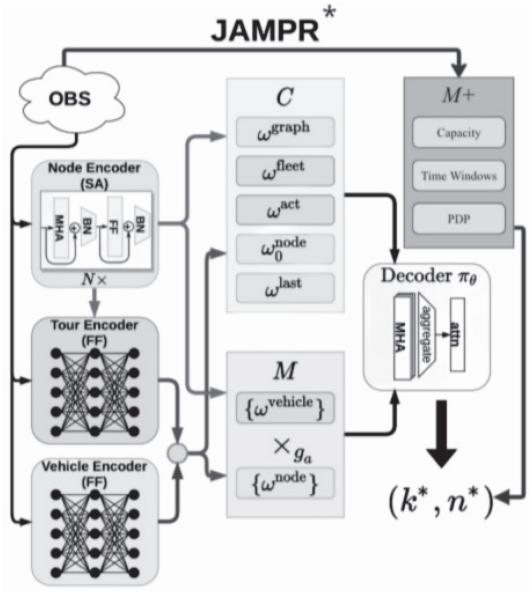


Рис. 1. Архитектура модели JAMPR, модифицированная для решения проблем CPDPTW.

Рабочий процесс модели: сначала кодировщик получает характеристики $x_i$ каждого узла $i$ (координаты, вес грузов, временные окна и т.д.) и кодирует их в скрытый линейный вектор $\tilde{x_i} \in R^{d_{emb}}$ размером $d_{emb}$. Затем модель декодера вычисляет запрос внимания для каждого $\tilde{x_i}$ по отношению к конкретному контексту $C^{(t)}$ на шаге декодирования $t$ для получения оценок для всех узлов, которые могут быть добавлены к текущему маршруту. Здесь контекст включает неявное встраивание графа задачи и дополнительную информацию о проблеме, такую как индекс узла депо, последний узел, добавленный к текущему маршруту, и оставшуюся пропускную способность. Полученные оценки затем либо используются в жадной процедуре выбора, т.е. всегда выбирается узел с наивысшей оценкой, либо используется softmax для преобразования его в распределение, применяемое для выборки. В общем случае модель кодер–декодер представляет собой политику $\pi(i^{(t+1)}|C^{(t)}, x; \theta)$ с обучаемыми параметрами $\theta$. Полная архитектура $JAMPR^*$ из [11] представлена на рис. 1. Здесь OBS — текущее состояние среды, Node Encoder — кодировщик состояния вершин графа, Tour Encoder — кодировщик текущего состояния маршрута, Vehicle Encoder — кодировщик состояния грузовика. Блок $C$ — контекст пути, $M$ — контекст Масок, каждый из весов $w$ внутри ассоциируется с разными показателями текущего состояния среды: graph — граф, fleet — флот, act — последнее действие, node — вершины, last — последняя вершина, vehicle — текущий грузовик. Блок $M+$ обозначает обучаемые маски, каждый из Capacity, Time Windows и PDP относится к совему типу ограничений, Объем, Временные Окна и Вывоз и Доставка соответственно.

## 4. Система *Smart Routes*

### *4.1. Требования*

Система должна соответствовать следующим функциональным **основным требованиям**.

— **Решение задачи VRP с популярными ограничениями.** Это включает в себя решение VRP с общими и важными ограничениями, такими как вместимость транспортных средств и временные окна для доставки. С расширением системы будут добавлены новые ограничения.

— **Решение задач большого масштаба (∼1000 точек).** Решение VRP большого масштаба представляет собой вызовы, поскольку увеличение количества клиентов для посещения затрудняет поиск решений в алгоритмах и приводит к увеличению времени поиска.

— **Использование данных реального мира.** Так как данная работа сравнивает алгоритмы с использованием искусственно сгенерированных данных, предлагаемая система также включает функцию обработки данных реального мира в заданном формате.

— **Проведение экспериментов с классическими эвристическими, точными и глубокими методами обучения с подкреплением.** Задача определения лучшего алгоритма для решения VRP остается актуальной. Поэтому предлагаемая система предоставляет возможность экспериментировать с алгоритмами из разных классов.

— **Интерактивное использование системы.** Для упрощения использования встроенных алгоритмов предусмотрен интуитивно понятный интерфейс для настройки параметров и взаимодействия с системой.

— **Визуальная интерпретация решений.** Реализация алгоритмов иногда может затруднить определение корректности и оптимальности решений. Визуализация решения упрощает этот процесс верификации, предоставляя визуализацию построенного маршрута.

— **Сравнение показателей качества.** При сравнении нескольких алгоритмов полезно анализировать различия в качестве полученных решений. Графики, демонстрирующие уменьшение или увеличение стоимости маршрута за определенный период времени, особенно полезны и удобны для проведения этих экспериментов.

### *4.2. Архитектура*

Архитектура системы представлена на рис. 2. Архитектура системы Smart Routes включает три блока: 1) Модуль данных (Dataset) для обработки данных и загрузки в различных форматах; 2) Модуль алгоритма (Algorithm) занимается обучением и оценкой алгоритмов, позволяя добавлять пользовательские алгоритмы; 3) Модуль решения (Solution) формирует метрики, визуализирует результаты и предоставляет веб-интерфейс для взаимодействия. Значения на рис. 2: Input Dataset — Входной набор данных, Generation

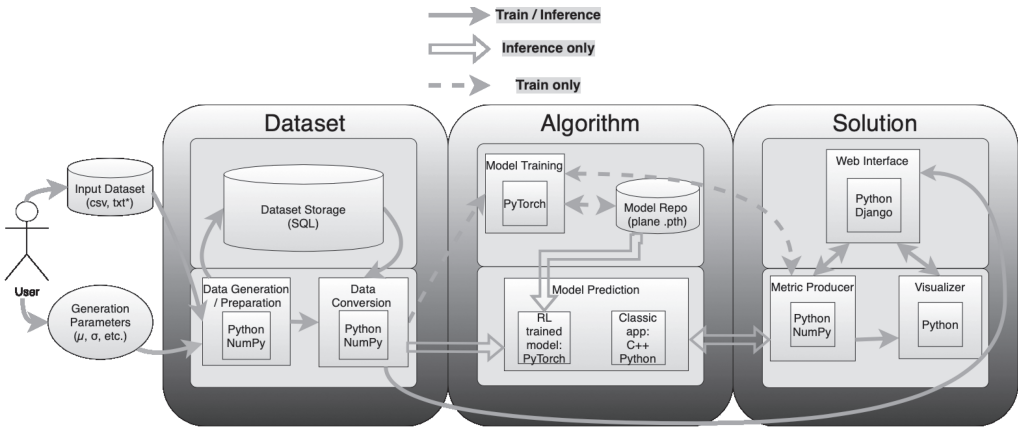


Рис. 2. Архитектура системы Smart Routes.

Parameters — Параметры генерации распределения, Dataset — Набор данных, Dataset Storage — Хранение набора данных, Data Generation — Генерация набора данных, Data Convertion — Преобразование набора данных, Algorithm — Алгоритм, Model Training — Обучение модели, Model Repo — Репозиторий модели, Model Prediction — Предсказание модели, RL Trained Model — Обученная модель обучения с подкреплением, Solution — Решение, Web Intefrace — Веб-интерфейс, Metric Producer — Предиктор метрик, Visualizer — Визуализатор.

Опишем основные шаги использования системы. Пользователь устанавливает систему, строго следуя документации. При запуске его перенаправляют в веб-браузер, где отображается веб-страница с предложением ввести необходимые параметры.

1. **Данные и параметры.**
   a. пользователь выбирает проблему, которую он собирается решить: $CVRP$, $VRPTW$, $CVRPTW$;
   b. выбирается имя алгоритма решения;
   c. устанавливается временное ограничение для решения одной задачи;
   d. выбирается вариант решения задач последовательно или параллельно на доступном количестве процессоров;
   e. загружается набор данных в определенном формате;
   f. указываются параметры для генерации данных, если это необходимо: 1) $\mu$ — математическое ожидание; 2) $\delta$ — дисперсия; 3) $capacity_{\min}$ — минимальное значение объема груза потребителя; 4) $capacity_{\max}$ — максимальное значение объема груза потребителя; 5) $n_{samples}$ — количество примеров задач; 6) $n_{customer}$ — количество клиентов в каждой задаче; 7) $service_{window}$ — правое временное окно для депо; 8) $service_{duration}$ — время обслуживания клиента.
2. **Набор данных.** Пользовательские параметры и загруженные данные передаются блоку *Генерации/Подготовки данных*, где данные либо ис-

кусственно генерируются на основе указанных параметров пользователя, либо сохраняются в переменных. Затем пользовательские наборы данных сохраняются в базе данных или немедленно передаются блоку *Преобразование данных*, где данные преобразуются и приводятся к необходимым типам.

3. **Алгоритм.** Этот блок получает предобработанные данные и параметры, указанные пользователем. Система определяет, к какому классу проблем принадлежит выбранный алгоритм: *classic_heuristics*, *exact_methods*, *neural_methods*.

   В результате для каждой представленной задачи пользователь получает конечный маршрут, т.е. список клиентов и его стоимость.

4. **Решение.** Результаты, полученные из предыдущего блока, передаются блоку *Генерации метрик*, который создает общий граф с метриками для выбранных пользователем алгоритмов, демонстрируя уменьшение стоимости маршрута в пределах указанного времени. Результаты также передаются блоку *Визуализатор*, который визуализирует полный графический маршрут на основе полученного списка клиентов.

По завершении работы система отображает сгенерированные графики, конечную стоимость маршрута и список клиентов в порядке их посещения на начальной веб-странице.

### *4.3. Преимущества*

Предложенная платформа обладает ключевой особенностью, позволяющей пользователям добавлять собственные алгоритмы и модели, что придает ей значительное преимущество перед другими фреймворками, поскольку:

1. Понимание инструментария фреймворка может занять много времени. Например, реализация модели CVRPTW с использованием программного обеспечения SCIP требовала значительного времени для разбора того, как правильно реализовать ограничения, определить архитектуру модели и предобработать входные данные.
2. Пакеты программного обеспечения, такие как OR-Tools, SCIP, Gurobi, CPLEX, позволяют создавать модели только в качестве точных методов. По мере увеличения размеров экземпляров проблемы, время поиска решения также растет, что может стать критичным, требуется инструмент реализации эвристик, удобный для сравнения результатов.

В описанной системе пользователи могут интегрировать свои алгоритмы, используя базовые классы. Встроенные алгоритмы подходят для всех методов решения проблем VRP. Также доступны удобные функции: визуализация результатов для оценки эффективности разработанного подхода и API, упрощающий использование встроенных подходов без интеграции собственных алгоритмов.

Таким образом, функционал платформы упрощает разработку, тестирование и экспериментальное сравнение подходов решения VRP, а также демон-

стрирует значительное преимущество по сравнению с существующим программным обеспечением.

### *4.4. Сравнение Smart Routes и Классических фреймворков*

В настоящее время существуют фреймворки, предоставляющие инструменты для решения проблемы маршрутизации транспортных средств (VRP) с различными ограничениями. Самые популярные из них: OR-Tools, Gurobi и SCIP.

В таблице представлены типы алгоритмов, поддерживаемые этими фреймворками (обозначено знаком +, в противном случае знаком —).

| Тип алгоритма | Фреймворк | | | |
|---|---|---|---|---|
| | SCIP | Gurobi | OR-Tools | Smart Routes |
| Точные методы | + | + | + | + |
| Эвристические методы | + | + | + | + |
| Глубокое обучение с подкреплением | — | — | — | + |
| Визуализация результатов | — | — | — | + |
| Добавление собственных алгоритмов | — | — | — | + |

Важно отметить, что OR-Tools и SCIP могут не предоставлять тот же уровень производительности и масштабируемости, что и Gurobi, при решении очень крупных задач оптимизации.

Предложенная система отличается от упомянутого программного обеспечения в нескольких аспектах. Помимо охвата всех типов алгоритмов для решения VRP, Smart Routes позволяет интегрировать собственные модели, демонстрируя гибкость, не подвергая риску общую архитектуру. В ней реализована визуализация для создания графиков производительности и представления конечных маршрутов. Кроме того, при установке системы OR-Tools и SCIP устанавливаются автоматически, что упрощает проведение экспериментов сравнения моделей пользователей с моделями, реализованными на наборах инструментов, предоставленных этими фреймворками. Нет необходимости разрабатывать алгоритмы для решения индивидуальных логистических задач, поскольку предложенная система уже дает встроенные реализации. Более того, система предоставляет интерфейс, который упрощает рабочий процесс и делает его удобным инструментом для пользователей.

## 5. Данные

Рассматриваемыми алгоритмами решалась задача *CVRPTW*, так как наибольший интерес представляет поведение рассмотренных подходов с несколькими ограничениями одновременно, и рассматриваемая задача решалась в *Soft-постановке*, т.е. могут быть точки, которые невозможно посетить. В таком случае они исключаются из итогового маршрута и выносятся в штраф, а

точнее, в формуле стоимости увеличивается переменная *missed_ nodes*. В работе для *CVRPTW* выбраны подходящие экземпляры задач из распределения на основе статистики R201, известного эталонного набора Соломона [21]. Объемы грузовиков заданы как $Q_{50} = 750$, $Q_{100} = 1000$ для задач размеров 50 и 100 соответственно. Общий временной горизонт равен $[a_0, b_0]$, где $a_0 = 0$ для всех примеров, а правая граница $b_0$ меняется в зависимости от размера задачи: 1000 для 50 и 100 точек. При этом продолжительность обслуживания $h_i$ равномерно устанавливается равной 10.

## 6. Эксперименты

Для проведения экспериментов были сгенерированы искусственные наборы данных, состоящие из 100 примеров с размерами 50 и 100 точек. Каждой задаче был назначен определенный временной лимит для решения с использованием алгоритмов: 100 с для задач с 50 точками и 200 с для задач со 100 точками. Поскольку точные методы включают полный поиск возможных решений, им требовалось в 10 раз больше времени для каждой задачи, чтобы получить любое решение, даже если оно не оптимальное. Поэтому точному методу было выделено 1000 и 2000 с для задач с 50 и 100 точками соответственно.

Эксперименты проводились с использованием разработанной системы Smart Routes, которая упростила процесс настройки параметров при запуске различных алгоритмов и задач. Для экономии времени при экспериментах задачи были распараллелены на все доступные процессоры машины.

Все эксперименты выполнялись на сервере с графическим процессором NVIDIA Tesla A40 и процессором Intel(R) Xeon(R) Gold 6226R CPU@2.90GHz с 16 виртуальными ядрами.

### *6.1. Основные результаты*

Основные результаты экспериментов отображены на рис. 3, где каждый алгоритм представлен в виде точки. Прямые линии различных цветов соединяют точки, показывая, что общая стоимость маршрутов увеличивается с увеличением размеров проблемы. Результаты, полученные с помощью SCIP, соединены наклонной сплошной линией, а пунктиры указывают на потенциальное геометрическое положение результатов SCIP. Ось $x$ представляет среднее время решения для одной задачи, а ось $y$ представляет среднюю стоимость маршрута для одной задачи. На рис. 3 горизонтальная линия указывает время, необходимое для алгоритма для достижения наилучшей стоимости пути, вертикальная линия указывает наилучшую достижимую алгоритмом стоимость. Наклонная сплошная линия показывает предел применимости точных методов: слева недостаточно времени для алгоритма, чтобы предоставить решение, зона справа представляет собой благоприятную область для выбора точных методов. Обозначения на рис. 3: Cost —

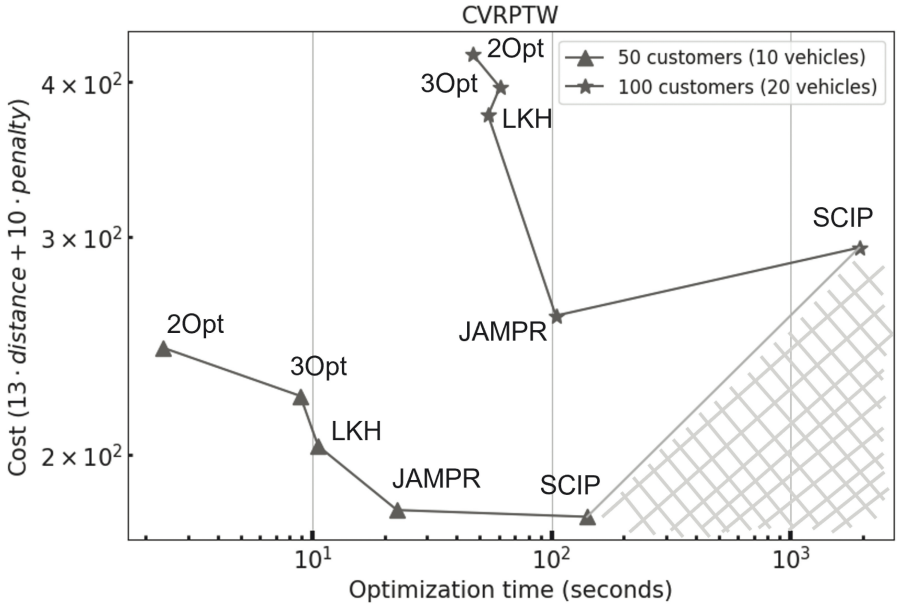


Рис. 3. Итоговый результат классических подходов.

стоимость пути, 50 customers (10 vehicles) — задача с 50 точками и 10 грузовиками, 100 customers (20 vehicles) — задача со 100 точками и 20 грузовиками, Optimization Time — время оптимизации.

На основе полученных результатов можно сделать следующие выводы.

1. Увеличение значения $\lambda$ в алгоритме LKH ($\lambda$ Opt) улучшает качество решений, но увеличивает время поиска.
2. По мере увеличения размерности задачи и количества транспортных средств время решения SCIP значительно увеличивается. Удвоение размерности задачи приводит к примерно 14-кратному увеличению времени решения SCIP. В отличие от этого нейронная сеть JAMPR проявляет себя лучше, чем точные методы, как по качеству решения, так и по времени поиска для экземпляров задач с размерностью 100 точек. Это подтверждает, что точные методы имеют ограниченную применимость для решения задач с размерностью 100 точек и более.

Две последующие главы предоставляют более подробное описание проведенных экспериментов на экземплярах задач с размерностью 50 и 100 точек.

### *6.2. Результаты для экземпляров задач с 50 точками*

Для диаграммы с метриками GAP было рассчитано процентное отклонение алгоритмов в каждой точке по сравнению с окончательным лучшим результатом, достигнутым в течение 1000 секунд оптимизации (SCIP). Общая метрика представляет среднее значение среди всех решенных экземпляров задач.

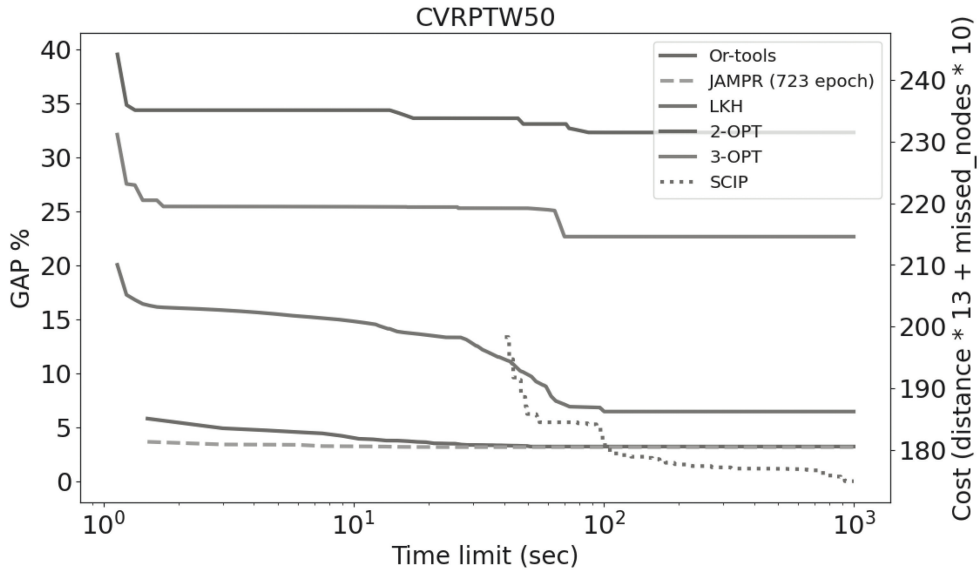


Рис. 4. Эффективность моделей на CVRPTW с 50 точками.

На рис. 4 отражено изменение качества окончательных стоимостей маршрутов по мере оптимизации времени. Сплошные линии — результаты эвристик. Пунктир — лучший результат SCIP за максимальное время. Пунктирная линия — JAMPR после 700 эпох — быстрое субоптимальное решение, сходящееся к эвристике. Две шкалы: общая стоимость (справа) и отклонение от лучшего решения (слева). Для задач с 50 точками как классические эвристики, так и глубокое обучение с подкреплением быстро предоставляют субоптимальные решения. Результаты методов 2-Opt, 3-Opt, LKH, OR-Tools и JAMPR достигают плато, но в начале оптимизации классические эвристики предоставляют начальные жадные решения, с LKH, постепенно приближающимся к результатам JAMPR и OR-Tools. Однако после примерно 10 с SCIP становится менее эффективным, чем LKH, OR-Tools и JAMPR, но после 100 с SCIP превосходит их. Несмотря на то, что точные методы обеспечивают лучшее решение, они требуют значительно больше времени, чтобы превзойти LKH, JAMPR и OR-Tools по качеству решения, при этом конечная разница между моделью глубокого обучения и SCIP составляет менее 5%. Обозначения на рис. 4: Cost — стоимость пути, GAP — метрика, процентное отставание от лучшего решения, Time limit (sec) — время оптимизации в секундах.

Таким образом, для задач с 50 точками, несмотря на то, что точные методы демонстрируют лучшее решение, их превосходят классические эвристики и методы глубокого обучения с подкреплением по времени на порядок. Важно отметить, что нейронные сети и классические эвристические подходы могут предоставить субоптимальное решение для CVRPTW уже в первые несколько с.

На рис. 5 показано, как меняется качество стоимости маршрута во времени для задач со 100 точками. Сплошные линии — результаты эвристик. Пунктир — результаты SCIP. Пунктирная линия — JAMPR после 23 эпох — быстрое субоптимальное решение, аппроксимирующее эвристику. Две шкалы: общая стоимость (справа) и отклонение от лучшего решения (слева). Как и в случае с 50 точками, была вычислена процентная стоимость для каждого алгоритма относительно лучшего результата, показанного SCIP, в пределах максимального времени оптимизации (200 с для классических эвристик и метода глубокого обучения, и 2000 с для SCIP). Обозначения на рис. 5: Cost — стоимость пути, GAP — метрика, процентное отставание от лучшего решения, Time limit (sec) — время оптимизации в секундах.

На графике видно, что классические эвристические методы предоставляют более быстрое субоптимальное решение, но уступают по качеству JAMPR и OR-Tools. Несмотря на то, что JAMPR достигает плато примерно на 100-й с, его начальное жадное решение превосходит LKH примерно на 10%, а конечный результат превосходит LKH GAP примерно на 50%. При увеличении размерности задачи увеличивается разница между начальными решениями в модели глубокого обучения и эвристиками.

Также на рис. 5 видно, что для SCIP потребовалось более 900 с для первого решения, что в 13 раз дольше, чем JAMPR и OR-Tools. В отведенное время SCIP не предложил оптимальное решение, и его первое решение было примерно на 50% дороже, чем решения, предоставленные эвристиками OR-Tools и JAMPR. Это указывает на то, что с увеличением размерности задачи точ-

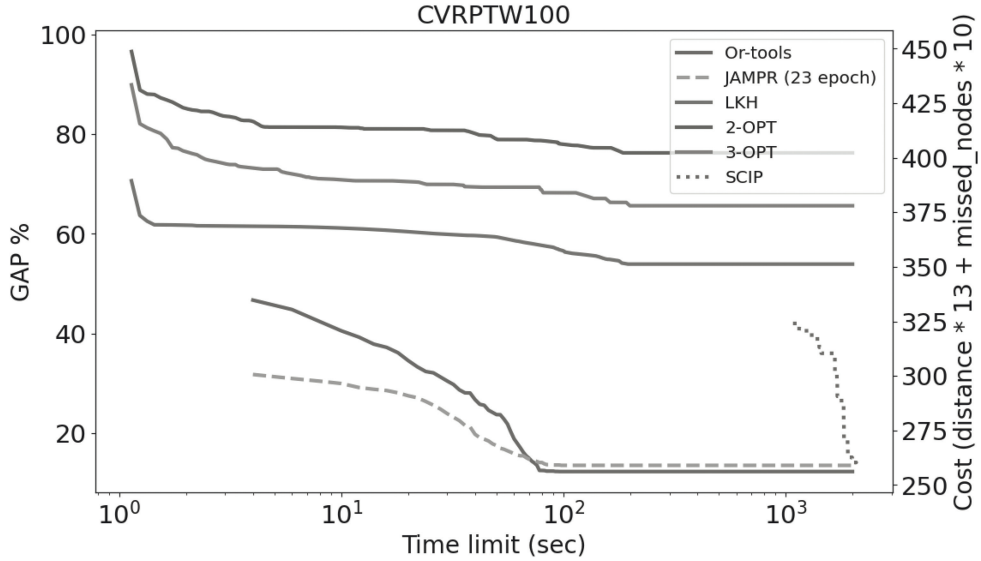


Рис. 5. Эффективность моделей на задаче CVRPTW с 100 точками.

ные методы становятся менее применимыми из-за экспоненциального роста времени.

Таким образом, в случае задач со 100 точками методы глубокого обучения с подкреплением могут предоставить быстрое субоптимальное решение, превосходящее классические эвристики по качеству из-за более длительного времени обучения, и превосходящее точные методы по времени, необходимому для нахождения решения.

## 7. Заключение

Проблема оптимизации маршрутов с учетом реалистичных ограничений становится чрезвычайно актуальной в свете глобального роста городского населения. Несмотря на наличие подходов, которые теоретически обеспечивают точное оптимальное решение, их применение становится трудным с увеличением размера задачи из-за экспоненциальной сложности. Здесь исследуется задача оптимизации маршрутов с ограничением по времени и вместимости транспортного средства (CVRPTW) и сравниваются решения, полученные с использованием точного решателя SCIP [5] с эвристическими алгоритмами, такими как LKH, 2-OPT, 3-OPT [17], фреймворком OR-Tools [20] и моделью глубокого обучения JAMPR [11].

Все метрики, представленные в статье, были получены с использованием платформы Smart Routes. Эта система обладает возможностью решения задачи оптимизации маршрута с использованием различных подходов, что облегчает исследования и сравнение результатов.

На основе экспериментальных результатов можно сделать следующие выводы:

— Для задач с 50 точками как классические эвристические методы, так и методы обучения с подкреплением демонстрируют свою эффективность, предоставляя быстрые субоптимальные решения. Общие результаты близки к результатам, полученным точным методом (SCIP), с разницей менее 5%. Это свидетельствует о том, что для задач с 50 точками классические эвристические методы и методы, основанные на нейронных сетях, могут предложить хороший баланс между качеством решения и временем поиска.

— Для задач со 100 точками точные методы требовали в 13 раз больше времени для нахождения начального решения. Они не смогли предоставить оптимальное решение в заданный срок, что привело к увеличению затрат на маршрут до 50%. Классические эвристические методы предоставляют быстрые субоптимальные решения, но начинают отставать от более продвинутых методов, таких как JAMPR и OR-Tools, по качеству решения. Эти результаты свидетельствуют о полной нецелесообразности точных методов для решения задачи оптимизации маршрута со 100 (и более) точками.

Разработанная платформа является важным компонентом для будущих исследований в области оптимизации маршрутов с различными ограничениями.

## СПИСОК ЛИТЕРАТУРЫ


1. *Laporte G., Nobert Y.* A branch and bound algorithm for the capacitated vehicle routing problem // Operations-Research-Spektrum. 1983. V. 5. P. 77–85.
2. *Cook W., Rich J.L.* A parallel cutting-plane algorithm for the vehicle routing problem with time windows // Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Housten. 1999.
3. *Augerat P., Naddef D., Belenguer J.M., et al.* Computational results with a branch and cut code for the capacitated vehicle routing problem / Research report — IMAG. 1995.
4. *IBM ILOG Cplex* V12. 1: User's Manual for CPLEX // Int. Busin. Machin. Corporat. 2009. V. 46. No. 53. P. 157.
5. *Bestuzheva K., Besancon M., Chen W.-K., et al.* The SCIP Optimization Suite 8.0 // Technical Report, Optimization Online. 2021. http://www.optimization-online.org/DB_HTML/2021/12/8728.html
6. *Gurobi Optimization, LLC* Gurobi Optimizer Reference Manual // 2023. https://www.gurobi.com
7. *Nazari M., Oroojlooy A., Snyder L., et al.* Reinforcement learning for solving the vehicle routing problem // Conf. Advances Neural Inform. Proc. Syst. 2018. V. 31.
8. *Vinyals O., Fortunato M., Jaitly N.* Pointer networks // Conf. Advances Neural Inform. Proc. Syst. 2015. V. 28.
9. *Kool W., Hoof H.V., Welling M.* Attention, learn to solve routing problems! // 2018. arXiv preprint arXiv:1803.08475.
10. *Vaswani A., Shazeer N., Parmar N., et al.* Attention is all you need // Conf. Advances Neural Inform. Proc. Syst. 2017. V. 30.
11. *Falkner J.K., Schmidt-Thieme L.* Learning to solve vehicle routing problems with time windows through joint attention // arXiv preprint arXiv:2006.09100. 2020.
12. *Chen X., Tian Y.* Learning to perform local rewriting for combinatorial optimization // Conf. Advances Neural Inform. Proc. Syst. 2019. V. 32.
13. *Lu H., Zhang X., Yang S.* A learning-based iterative method for solving vehicle routing problems // International conference on learning representations. 2019.
14. *Li S., Yan Z., Wu C.* Learning to delegate for large-scale vehicle routing // Conf. Advances Neural Inform. Proc. Syst. 2021. V. 34.
15. *Soroka A.G., Meshcheryakov A.V., Gerasimov S.V.* Deep Reinforcement Learning for the Capacitated Pickup and Delivery Problem with Time Windows // Patt. Recognit. Imag. Anal. 2023. V. 33. No. 2. P. 169–178.
16. *Groër C., Golden B., Wasil E.* A library of local search heuristics for the vehicle routing problem // Math. Program. Comput. 2010. V. 2. P. 79–101.
17. *Helsgaun K.* An effective implementation of the Lin–Kernighan traveling salesman heuristic // Eur. J. Oper. Res. 2000. V. 126. No. 1. P. 106–130.
18. *Çetinkaya C., Karaoglan I., Gökcen H.* Two-stage vehicle routing problem with arc time windows: A mixed integer programming formulation and a heuristic approach // Eur. J. Oper. Res. 2013. V. 230. No. 3. P. 539–550.
19. *Tahernejad S., Ralphs T.K., DeNegre S.T.* A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation // Math. Program. Comput. 2020. V. 12. P. 529–568.

20. *Perron L.* Operations research and constraint programming at google // International Conference on Principles and Practice of Constraint Programming. 2011. P. 2–2.
21. *Solomon M.M.* Algorithms for the vehicle routing and scheduling problems with time window constraints // Oper. Res. Inf. 1987. V. 35. No. 2. P. 254–265.

*Статья представлена к публикации членом редколлегии А.А. Галяевым.*

Поступила в редакцию 08.07.2023

После доработки 10.10.2023

Принята к публикации 20.01.2024