Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт
Высшая школа теоретической механики и математической физики

Работа допущена к защите

Директор ВШТМиМФ

д.ф.-м. н., чл.-корр. РАН

А.М. Кривцов

«___»_____2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
МАГИСТЕРСКИЯ ДИССЕРТАЦИЯ
ОПТИМИЗАЦИЯ РАБОТЫ СИСТЕМЫ СБОРА И ТРАНСПОРТИРОВКИ НЕФТЕГАЗОВОГО МЕСТОРОЖДЕНИЯ

По направлению 01.04.03 «Механика и математическое моделирование»

Направленность 01.04.03_03 «Механика и цифровое производство»

Выполнил
студент гр.5040103/20301 — Е.Е. Анокина

Руководитель
Доцент ВШТМиМФ, к.ф.-м. н. — О.С. Лобода

Консультант
Руководитель направления аналитики
Auto, Авито (ООО «Авито Тех») — Д.С. Перец

Консультант
Главный специалист НОЦ
«Газпромнефть-Политех» — К.А. Печко

Консультант
по нормоконтролю — Е.А. Хайбулова

Санкт-Петербург 2024

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Физико-механический институт

Высшая школа теоретической механики и математической физики

УТВЕРЖДАЮ

Директор ВШТМиМФ

А. М. Кривцов

«__»_____2024 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Анокиной Екатерине Евгеньевне, гр. 5040103/20301

1. Тема работы: Оптимизация работы системы сбора и транспортировки нефтегазового месторождения

2. Срок сдачи студентом законченной работы: 30.05.2024

3. Исходные данные по работе: актуальные публикации по теме работы, исследования в области оптимизационных алгоритмов

4. Содержание работы (перечень подлежащих разработке вопросов): построить модель, описывающую физические процессы в ССиТ, интегрировать оптимизационные алгоритмы в расчетный модуль, подобрать гиперпараметры для корректной работы оптимизационных алгоритмов, протестировать и оценить работу оптимизационных алгоритмов

5. Перечень графического материала (с указанием обязательных чертежей): схемы систем сбора и транспортировки нефтегазового месторождения, таблицы с результатами работы расчетной модели и оптимизационных алгоритмов

6. Консультанты по работе: Перец Д.С. – руководитель направления аналитики Auto, Авито (ООО «Авито Тех»), Печко К.А. – главный специалист НОЦ «Газпромнефть-Политех»

7. Дата выдачи задания 26.02.2024

Руководитель ВКР _____ Лобода О.С. – доцент ВШТМиМФ, к.ф.-м.н., доцент

Задание принял к исполнению 26.02.2024

Студент _____ Анокина Е.Е.

РЕФЕРАТ

На 49 с., на 3 рисунка, 6 таблиц.

ОПТИМИЗАЦИЯ, ГИПЕРПАРАМЕТРЫ, ДРЕВОВИДНАЯ ОЦЕНКА ПАРЦЕНА, ДИФФЕРЕНЦИАЛЬНАЯ ЭВОЛЮЦИЯ, РОЙ ЧАСТИЦ, НЕФТЕГАЗ, МАГИСТРАЛЬНЫЙ ТРУБОПРОВОД

Тема выпускной квалификационной работы: «Оптимизация работы системы сбора и транспортировки нефтегазового месторождения».

Данная работа посвящена проведению оптимизации параметров системы сбора и транспорта нефтегазового месторождения. Был проведен литературный обзор существующих решений данной задачи. Для выполнения поставленной цели была построена модель на основе систем линейных уравнений, описывающая физические процессы, происходящие на скважинах при транспортировке нефти по магистральному трубопроводу, давления в узлах сети и потоки в трубах. Полученные результаты сравнили с промышленной программой для моделирования нефтяных трубопроводов.

В полученный расчетный модуль были интегрированы четыре алгоритма оптимизации: древовидная оптимизация Парцена от библиотек Optuna и HyperOpt, алгоритм дифференциальной эволюции и роя частиц. В результате работы программы на простой транспортной сети были получены оптимальные гиперпараметры алгоритмов, которые увеличивают точность и уменьшают время вычислений, и параметры труб для получения максимальной прибыли. Были проанализированы результаты расчетов и сделаны выводы о работе оптимизационных алгоритмов.

THE ABSTRACT

49 pages, 3 pictures, 6 tables.

OPTIMIZATION, HYPERPARAMETERS, TREE-STRUCTURED PARZEN ESTIMATOR, DIFFERENTIAL EVOLUTION, PARTICLE SWARM OPTIMIZATION, OIL AND GAS, MAIN PIPELINE

The subject of the graduate qualification work is "Optimization of the operation of the oil and gas field collection and transportation system".

This work is devoted to optimizing the parameters of the collection and transport system of an oil and gas field. There was a literary review of the existing solutions to this problem. To achieve this goal, a model was built based on systems of linear equations describing the physical processes occurring at wells during oil transportation through a main pipeline, pressures at network nodes and flows in pipes.

The obtained results were compared with an industrial program for modeling oil pipelines. Four optimization algorithms were integrated into the calculation module: Tree-structured Parzen Estimator from Optuna and HyperOpt libraries, Differential Evolution algorithm and Particle Swarm Optimization. As a result of the program's operation on a simple transport network, optimal hyperparameters of algorithm which increase accuracy and reduce calculation time and pipe parameters for maximum profit were obtained. The results of calculations were analyzed, and conclusions were drawn about the work of optimization algorithms.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Алгоритмы оптимизации играют ключевую роль в различных областях науки, технологии и бизнесе. Они помогают находить оптимальные решения для сложных задач, улучшая производительность и эффективность процессов. Важность алгоритмов оптимизации состоит в том, что они позволяют нам экономить время, ресурсы и деньги, а также повышать качество принимаемых решений.

Существенную роль оптимизация играет в нефтегазовой промышленности. Ее алгоритмы используются для получения максимальной выгоды от ограниченного числа ресурсов.

Применение алгоритмов оптимизации позволяет улучшить эффективность производственных процессов, снизить затраты и повысить конкурентоспособность компаний на рынке энергетики. В условиях постоянно изменяющегося рынка и растущих экологических требований использование машинного обучения и искусственного интеллекта становится необходимым инструментом для успешного функционирования нефтегазовой отрасли.

На сегодняшний день было проведено множество исследований [9, 11, 18], в которых алгоритмы использовались для оптимизации процессы бурения, поднятия нефти на поверхность, а также управления работой скважин [10].

Кроме того, алгоритмы оптимизации применяются для оптимизации транспортировки и хранения нефти и газа. Путем оптимизации маршрутов и объемов перевозок можно сократить расходы на транспортировку и увеличить скорость доставки нефти и газа до потребителей.

В данной работе будет исследована оптимизация параметров системы сбора и транспорта нефтегазового месторождения. Для этого мы построим модель, описывающую физический процессы, происходящие на скважинах и при транспортировке нефти по магистральному трубопроводу. В полученный расчетный модуль интегрируем четыре алгоритма оптимизации: Tree-structured Parzen Estimator от библиотек Optuna и HyperOpt, алгоритм дифференциальной эволюции и роя частиц. В результате работы программы

мы получим оптимальные гиперпараметры алгоритмов, которые увеличиваю точность и уменьшают время вычислений, и параметры труб для получения максимальной прибыли.

Используя полученный данные, в будущем возможно усложнение и дополнение исследуемой задачи. Система сбора и транспорта может включать в себя множество скважин и промежуточных пунктов для установки оборудования. Также сеть можно усложнять за счет прокладки трубопроводов параллельно основному пути, создавая внутренние циклы, называемые лупингами. Можно будет исследовать большее количество параметров трубопровода и их влияния на целевую функцию.

ГЛАВА 1. ЛИТЕРАТУРНЫЙ ОБЗОР

1.1 Оптимизация как инструмент решения задач

Оптимизация — это процесс поиска лучшего решения проблемы, то есть такого, которое позволяющего достичь максимальных результатов при минимальных затратах.

Методы оптимизации исследуют предположения о характере откликов на целевую функцию, варьируя параметры и предлагая наилучший способ их изменения. Существует множество методов оптимизации, таких как поисковые, градиентные и эволюционные алгоритмы, методы машинного обучения и многие другие. Каждый из них имеет свои преимущества и недостатки, и выбор конкретного метода оптимизации зависит от поставленных задач и целей.

Работа оптимизационных алгоритмов сводится к поиску экстремума (максимума или минимума) целевой функции. Такой принцип работы позволяет проводить вычисления в большом спектре областей: от минимизации физических величин на микро- и макроуровнях до максимизации прибыли или эффективности логистических цепочек.

Методология оптимизации имеет многолетний опыт успешных усовершенствований во многих технологических и научных областях и используется для решения таких задач, как проектирование, планирование, правила бизнес-контроля, управление технологическими процессами и тому подобное. Оптимизация может применяться практически в любой сфере деятельности: от производства товаров и услуг до эффективного управления ограниченными ресурсами. Также оптимизация играет важную роль в принятии решений.

В современном мире оптимизация является мощным инструментом для решения задач в различных сферах деятельности. Правильно применяемая оптимизация позволяет повысить эффективность работы, снизить затраты и принимать обоснованные решения. Поэтому она является неотъемлемой частью любого научного исследования или бизнес-проекта.

1.2 Классификации оптимизационных алгоритмов

Как и типов решаемых задач существует множество алгоритмов оптимизации. Каждый из них имеет свои принципы работы и методологическую основу, которые могут давать преимущества в определенном аспекте перед другими алгоритмами. Тем не менее все разработанные алгоритмы можно сгруппировать по определенным признакам и механизмам работы.

Например, по типу оптимизационной переменной алгоритмы могут разделяться на непрерывные, дискретные и комбинаторные. Непрерывная оптимизация применяется в областях, связанных с физикой, инженерией, экономикой, где переменные принимают такие значения. Алгоритмы такие как метод градиентного спуска решают задачи путем итеративного приближения к оптимальному решению.

При дискретной оптимизации напротив переменные могут принимать только определённые значения. С помощью алгоритмов типа метода ветвей и границ или генетических алгоритмов, решаются задачи составления расписания, управления запасами и другие.

Комбинаторные методы оптимизации объединяют в себе два первых подхода и позволяют решать задачи коммивояжера, размещения оборудования и т. п., работая с комбинацией переменных.

Также одним из интересующих нас аспектов классификации алгоритмов является деление алгоритмов на локальные и глобальные. Первые направлены на улучшение уже существующего решения в своей ограниченной области. Вторые же нацелены на нахождение решения в большом пространстве и играют важную роль в оптимизации многомерных задач.

Локальные алгоритмы применяются для нахождения решения в небольшой области пространства поиска, в окрестности начальной точки. Они обычно используются для задач с ограниченным числом решений, в которых можно предположить, что оптимум находится вблизи текущего ответа.

Примерами локальных алгоритмов являются градиентный спуск, метод Ньютона-Рафсона и другие.

Глобальные оптимизационные алгоритмы же направлены на поиск оптимального решения во всем заданном пространстве. Генетические алгоритмы, алгоритм имитации отжига, роя частиц и т. д. применяются для задач, в которых необходимо учесть все возможные варианты вычислений и найти наилучшее из них.

Основные различия между локальными и глобальными алгоритмами заключаются в области поиска оптимального решения и сложности вычислений. Локальные алгоритмы как описано выше эффективны в поиске решения в малой окрестности начальной точки, но оно может не являться самым лучшим во всем пространстве поиска, оставаясь в локальном экстремуме. Глобальные алгоритмы, напротив, исследуют все пространство, но могут быть более ресурсоемкими и требовать больших вычислительных ресурсов.

Однако, стоит отметить, что в зависимости от структуры и сложности задачи, локальные алгоритмы могут быть эффективнее глобальных, и наоборот. Например, для задач с множеством локальных оптимумов глобальные алгоритмы могут быть менее эффективны, чем методы локальной оптимизации.

Также можно разделять алгоритмы по принципу работы на поисковые и эволюционные. Поисковые алгоритмы являются наиболее простыми и распространенными. Они основаны на последовательном переборе пространства значений из пространства поиска. Примерами таких алгоритмов являются градиентный спуск, Grid Search и Random Search.

Эволюционные алгоритмы, в свою очередь, имитируют процесс естественного отбора и эволюции в природе для поиска оптимальных решений. Они работают путем генерации случайных кандидатов и их оценки на основе заданных критериев для последующих кроссинговера и мутации. Отличительной особенностью эволюционных алгоритмов является

способность к работе с большими пространствами поиска. К эволюционным алгоритмам относятся генетические алгоритмы, метод роя частиц, дифференциальная эволюция и др.

1.3 Транспортировка нефти

Развитие транспорта нефти тесно связано с историей нефтяной промышленности. Промышленная добыча нефти началась более 100 лет назад. На протяжении большей части этого периода Россия была ведущим мировым производителем нефти, а российская система поставок нефти является одной из крупнейших в мире. Зарождением нефтяной промышленности в России принято считать 1864 год [2, 4], когда в долине реки Кубань было начато бурение первой в России нефтяной скважины.

Рассмотрим один из этапов превращения нефти в готовый к использованию продукт, а именно её транспортировку. Для этого требуется тщательная координация между разными уровнями цепочки поставок, чтобы доставлять десятки миллионов баррелей в день [8] от разных скважин, между устьями которых находятся сотни тысяч километров. В настоящее время для промышленного энергообеспечения используется железнодорожный, водный и трубопроводный транспорт [4].

Железнодорожный транспорт является наиболее распространенным способом транспортировки грузов. Перевозка различных нефтепродуктов осуществляется в специальных стальных вагонах-цистернах. Благодаря развитым железнодорожным сетям транспортировка может равномерно осуществляться в течение года как в большие города, так и в сельскохозяйственные регионы.

Как и при любом производстве при перевозе нефти и нефтепродуктов по железной дороге присутствуют расходы, которые хотелось бы минимизировать. Например, происходят значительные потери нефти при погрузке и транспортировке, новые цистерны загружаются по возвращении,

на строительство новых путей, ремонт стары и обслуживание поездов также является относительно дорогим.

Водный транспорт производит перевозки по рекам на баржах или речных танкерах, по морям и океанам на специально оборудованных танкерах, способных преодолевать огромные расстояния, транспортировка на малые расстояния для этого вида транспорта нецелесообразна. Транспортировка по воде не требует построения дополнительных сооружений, но грузоподъёмность судов может быть ограничена производительностью причала и прибрежного нефтехранилища.

Наиболее интересующим нас способом транспортировки нефти является трубопроводная доставка. Магистральные трубопроводы, имеющие большие диаметры и длины, достигающие тысячи километров, - основная артерия нефтегазового бизнеса, которая работает 24 часа в сутки и семь дней в неделю.

Транспортировка жидких ресурсов с помощью трубопровода имеет очень древнюю историю, начиная бамбуковых труб в древнекитайских провинциях и продолжая медными трубами Древнего Египта [1]. Поэтому эта технология перевозки стала очевидной при разработке первых месторождений. Начиная с конца 19 века, была построена обширная сеть трубопроводов, пролегающих под землей, а также в морях и океанах [6].

Трубопроводная транспортировка не зависит от погодных условий и способна обеспечивать бесперебойные поставки. Также преимуществами являются низкая себестоимость перекачки, высокая производительность труда, универсальность труб в разновидности транспортируемых нефтепродуктов и их минимальные потери при перемещении. Еще одним достоинством является возможность увеличения пропускной способности трубопровода за счет строительства дополнительных насосных станций и прокладки параллельных путей.

В ситуации растущего спроса на нефть и нефтепродукты происходят разведки новых месторождений. Часто некоторые из них являются труднодоступными для железнодорожного или водного транспорта. Поэтому

кроме описанных выше преимуществ, магистральный трубопровод может быть единственным способом транспортировки. Следовательно, для получения наибольшей экономической эффективности месторождения, необходимо наиболее оптимально спроектировать работу добычи и транспортировки полученных ископаемых.

1.4 Применение алгоритмов оптимизации в нефтегазе

Последние несколько лет в мире наблюдалась тенденция к увеличению добычи нефти [8]. В обозримом будущем многие месторождения будут истощены, и уже сейчас возникает необходимость планирования разработки новых нефтяных месторождений.

С развитием технологий, машинного обучения и алгоритмов оптимизации подход к проектированию промышленных объектов немного поменялся. Теперь для получения максимальной выгоды от ограниченных ресурсов прибегают к помощи этих алгоритмов, которые могут играть важную роль в принятии решений.

Алгоритмы оптимизации могут применяться в разных областях нефтяной промышленности [9, 11, 18]. Например, при добыче нефти, когда необходимо расположить скважины на местности таким образом, чтобы количество получаемого сырья было максимальным, а затраты минимальными. Или также можно применять алгоритмы в логистике, оптимизируя пути транспортировки, выбирая кратчайшие и оптимальные расстояния, или параметры самих труб для улучшения их пропускной способности.

В настоящее время применение искусственного интеллекта в нефтегазовой отрасли стремительно развивается [12], поскольку концепция искусственного интеллекта постепенно проникает на различные этапы отрасли, интеллектуальным стали бурение, производство, трубопровод, нефтеперерабатывающий завод и т. д. [17]

Используя алгоритмы оптимизации, был разработан ряд технологий практического применения в разведке и разработке месторождений. В области бурения новое оборудование, такое как автоматическая буровая установка и интеллектуальная бурильная труба, позволило повысить уровень бурения и значительно снизить затраты [10]. Кроме того, искусственный интеллект предоставил более точный метод проектирования схемы гидроразрыва пласта и выбора рабочих режимов. скважины и целевые пласты [13].

Кроме решения задачи конкретным методом оптимизации, можно сравнивать алгоритмы между собой. Сравнение алгоритмов позволяет увеличивать точность нахождения решения, лучше понимать принципы прогнозирования для получения оптимального результата, также помогает определить переменную, которая оказывает наибольшее влияние на прибыльность проекта [15].

1.5 Цель и задачи исследования

Целью исследования является проведение оптимизации параметров системы сбора и транспорта нефтегазового месторождения.

Задачи:

1. Построить модель, описывающую физические процессы в ССиТ;
2. Интегрировать оптимизационные алгоритмы в расчетный модуль;
3. Подобрать гиперпараметры для корректной работы оптимизационных алгоритмов;
4. Протестировать и оценить работу оптимизационных алгоритмов.

ГЛАВА 2. МАТЕРИАЛЫ И МЕТОДЫ

2.1 Гиперпараметры и Целевая функция

Гиперпараметр — параметр алгоритма машинного обучения, значение которого используется для управления процессом обучения. Гиперпараметры похожи на специальные настройки для алгоритма, определяют его поведение и эффективность, но не могут учиться сами по себе. Выбор правильных гиперпараметров может существенно повлиять на качество и скорость обучения модели. Гиперпараметрами могут являться скорость обучения модели, насколько глубоким должно быть дерево решений, тип регуляризации и т. д.

Поиск наилучших значений для этих гиперпараметров называется оптимизацией гиперпараметров. Он включает в себя опробование различных значений и методов таких как поиск по сетке (grid search), случайный поиск (random search), байесовская оптимизация и эволюционные алгоритмы, чтобы увидеть, какие из них лучше всего подходят для модели. Каждый из этих методов имеет свои преимущества и недостатки, а выбор конкретного метода зависит от особенностей задачи и доступных вычислительных ресурсов. Правильные гиперпараметры могут существенно повлиять на то, насколько хорошо работает модель и насколько точны ее прогнозы.

В общем и целом, выбор правильных гиперпараметров является важной частью создания успешной модели машинного обучения. Грамотное управление гиперпараметрами позволяет повысить качество модели, улучшить ее производительность и способность к обобщению на новые данные. Поэтому оптимизация гиперпараметров является важным этапом в процессе создания эффективных и точных моделей машинного обучения.

Целевая функция — вещественная или целочисленная функция нескольких переменных, подлежащая оптимизации в целях решения некоторой оптимизационной задачи. Целевая функция обычно представляется в виде уравнения или неравенства, ее переменные становятся

оптимизируемыми параметрами, а ее значение – показателем эффективности решения. Задача оптимизации заключается в поиске такого значения переменных, при котором значение целевой функции будет минимальным или максимальным, в зависимости от постановки задачи.

2.2 Байесовская оптимизация

Одним из итерационных методов, позволяющих осуществить подбор гиперпараметров, является байесовская оптимизация. Она позволяет найти оптимум функции, не вычисляя ее производной. Также пользуясь этим методом, мы на каждом шаге сможем определить в какой следующей точке мы с наибольшей вероятностью найдем лучшее решение для заданного оптимума. Это позволяет сократить количество вычислений целевой функции, каждое из которых может быть довольно затратным по времени.

Алгоритм инициализирует работу двух моделей: exploaration и exploitation models, чтобы найти компромисс между степенью исследования и эксплуатации. Сутью первой является исследование малоизученных областей на данной итерации. Это позволяет с меньшей вероятностью пропустить оптимальное значение в областях, которые могли быть первоначально откинуты. Вторая модель работает с уже хорошо изученными данными, где вероятность нахождения оптимального решения выше.

Отношение оценок первой модели ко второй позволяет балансировать между exploaration и exploitation, так как гиперпараметры, семплируемые из него, близки к оптимуму, но это же привносит элемент exploration, так как они не равны оптимуму в точности.

2.3 Tree-structured Parzen Estimator

Алгоритм Tree-structured Parzen Estimator (TPE) так же, как и байесовский алгоритм является итерационным: на каждом шаге принимается решение о

том, какие следующие значения оптимизируемых гиперпараметров будут выбраны, исходя из решений, полученных ранее. Но идейно работа алгоритмов отличается.

Для начала реализации алгоритма необходимо задать диапазон гиперпараметра, на котором мы сможем оценить качество работы модели, то есть произвести «разогрев». Если такую группу задать трудно или невозможно, то можно применить Random Search для сужения начальной выборки.

Для работы этих моделей на следующем шаге все полученные данные разбиваются на две группы. В первую группу попадают лучшие значения гиперпараметра (заранее заданный процент от общего числа решений), а во вторую – все остальные. Далее для каждой группы строятся оценки и на их основе формируются новые начальные данные для следующей итерации. С помощью оценки из первой группы семплируюся, то есть отбираются наиболее репрезентативные, значения-кандидаты, число которых пользователь может задать заранее. Из насемплированных кандидатов отбираются те, которые с большей вероятностью окажутся в первой группе, чем во второй. Для этого вычисляется Expected Improvement (EI), которое является частным функции оценки первой группы к функции оценки второй группы для данного кандидата. Затем мы отбираем значение с максимальным EI, чтобы обучить модель с этим значением гиперпараметра.

Другим распространенным методом для функции сбора данных является Probability of Improvement (PI). В то время как PI склоняется к использованию знаний, EI в большей мере исследует невидимые области.

Далее качество работы модели проверяется на валидационной выборке и на основе полученных результатов снова формируются две группы с новыми оценками: снова ранжируются все кандидаты с учетом качества модели, процент лучших значений формирует первую группу, все остальные вторую. И так повторяется столько количество раз, сколько итераций было задано.

Если расширять задачу на общий случай, когда требуется нахождение нескольких гиперпараметров, то мы придем к сути названия алгоритма. Алгоритм работает с гиперпараметрами, представляя их в форме дерева.

Для начала также требуется произвести «разогрев» с помощью Random Search на первоначальном распределении значений, если нет возможности изначально сузить выборку. Далее обновление дерева на каждой итерации происходит в два этапа: движение от корня к листу (максимизация EI) и от листа к корню (обновление оценок).

Так же как при работе с одним гиперпараметром при движении от корня к листу на каждой вершине отбираются семплирование кандидаты, максимизирующие EI. Так алгоритм идет от одной вершины к другой выбирая путь, соответствующий максимизации EI, пока не дойдет до листа, где поиск закончится. Весь пройденный путь составляет полный набор гиперпараметров для модели, и ее с этими значениями можно провалидировать.

Далее после валидации модели движение идет в обратном направлении от листа наверх и в каждой вершине обновляется распределение согласно с информацией о полученном качестве. Постороение новых распределений происходят так же, как и прежде: имеющиеся значения гиперпараметров переранжируются по качеству с учётом результата последнего кандидата (этот результат общий для всех вершин вдоль пути), строятся оценка на группе из лучших значений (заданный процент) и оценка на группе из всех оставшихся.

На выходе работы алгоритма мы получаем полный набор гиперпараметров, на которых было получено лучшее качество работы модели за все итерации.

При том, что алгоритм TPE может быть эффективным в оптимизации сложных и многомерных функций благодаря использованию дерева для оценки и выбора лучших точек для дальнейшей оптимизации, использование дерева для оценки функций может привести к высокой вычислительной сложности алгоритма, особенно при работе с большими объемами данных или высокой размерности [3]. Также TPE требует тщательной настройки

параметров для достижения оптимальной производительности, что может потребовать времени и усилий исследователя.

Еще одним недостатком TPE является его неумение выявлять внутренние зависимости между гиперпараметрами. По смыслу работы алгоритма гиперпараметры, на каждой итерации лежащие в разных путях на дереве, являются независимыми. Это суждение может оказаться неверными, гиперпараметры могу зависеть друг от друга, и если связь между ними пользователю неизвестна, то алгоритм сам ее не найдет. Поэтому для некоторых задач, решаемых с помощью TPE, необходимо задавать регуляризацию для предотвращения переобучения модели [3].

2.4 Библиотека Optuna

Optuna — это библиотека настройки гиперпараметров, специально разработанная для работы с различными платформами. Она поддерживает различные алгоритмы оптимизации и предоставляет простой и гибкий интерфейс для определения пространства поиска, целевой функции и гиперпараметров, которые могут быть автоматически подобраны для моделей машинного обучения [22]. Optuna разработана таким образом, чтобы быть масштабируемой и возможностью легкого интегрирования с другими библиотеками машинного обучения [21].

Optuna проста в использовании: она полностью написана на Python и имеет мало зависимостей, что позволяет быстро перенести теорию на реальные расчеты.

Для выборки гиперпараметров Optuna предоставляет функции, позволяющие четко разграничить область поиска решений. В соответствии с требованиями разработчика может происходить автоматический поиск оптимальных гиперпараметров на целочисленной, непрерывной, логарифмической и др. сетках [22]. Также пространство поиска можно

настроить, используя знакомый синтаксис Python, включая условные выражения и циклы.

В Optuna реализовано большое количество современных алгоритмов, позволяющих производить эффективную оптимизацию и сокращать число бесперспективных испытаний для получения более быстрых результатов. Реализованные семплеры постоянно сужают пространство поиска, используя записи предложенных значений параметров и оцененных целевых значений, что приводит к оптимальному пространству поиска, которое выдает параметры, приводящие к лучшим целевым значениям [21].

Еще одним достоинством, реализующим быструю работу алгоритмов, является распараллеливание поиска по гиперпараметрам в нескольких потоках или процессах без изменения кода [22]. Также в Optuna реализованы функции визуализации для анализа гиперпараметрической оптимизации.

Таким образом, ключевыми особенностями Optuna, которые делают ее оптимальной платформой для оптимизации гиперпараметров, являются:

- Лёгкая, универсальная архитектура, независящая от платформы;
- Вариативность определения пространства поиска;
- Современные алгоритмы;
- Простое распараллеливание;
- Быстрая визуализация.

Недостатки Optuna связаны с оптимизационными алгоритмами, которые она использует. При неосторожном подходе оптимизации гиперпараметров с помощью автоматизированных методов модель может переобучиться под конкретный набор данных, что может снизить ее обобщающую способность. С этим же сложность оптимизации возрастает экспоненциально в зависимости от количества параметров. То есть количество необходимых испытаний увеличивается экспоненциально при увеличении количества параметров, поэтому рекомендуется не добавлять несущественные параметры.

2.5 Библиотека HyperOpt

HyperOpt — это библиотека Python с открытым исходным кодом для байесовской оптимизации. Она предназначена для крупномасштабной оптимизации моделей с сотнями гиперпараметров, включая глубокие нейронные сети, регрессионные модели и другие [7]. Библиотека была явно использована для оптимизации конвейеров машинного обучения, включая подготовку данных, выбор модели и гиперпараметры модели.

Работа HyperOpt заключается в поиске наилучшего значения скалярной, возможно, стохастической функции по набору вероятных аргументов этой функции. В то время как многие пакеты оптимизации предполагают, что эти входные данные взяты из векторного пространства, HyperOpt отличается тем, что он побуждает описывать пространство поиска более подробно [7]. Предоставляя больше информации о том, где определена функция и где возможно находятся наилучшие значения, алгоритмы в HyperOpt могут выполнять поиск более эффективно [19].

В данной библиотеке реализовано три алгоритма оптимизации: Random Search, Simulated Annealing (метод имитации отжига) и Tree of Parzen Estimators [20].

Так как библиотека может работать с большим количеством гиперпараметров, в ней также реализованы механизмы, которые позволяют масштабировать процедуру оптимизации на нескольких ядрах и нескольких машинах. Также некоторым плюсом может считаться относительная простота и читаемость кода при оптимизации несложной целевой функции.

Если сравнивать работу пользователя с библиотеками Optuna и HyperOpt, то вторая уступает в легкости использования интерфейса. Например, вся информация о гиперпараметрах, которые были протестированы, и соответствующая оценка хранятся в отдельном объекте, и пользователю необходимо создавать отдельный объект Trials. Также в отличии от Optuna в HyperOpt реализовано очень мало функций для визуализации результатов

работы алгоритмов, что может затруднить или снизить точность анализа полученных значений гиперпараметров.

Ко всему вышесказанному также накладывается сложность в использовании библиотеки, так как некоторые пользователи указывают на то, что документация может быть недостаточно подробной и не всегда содержит достаточно информации для более сложных задач. Настройка HyperOpt может потребовать значительных усилий и времени, особенно для новичков в области машинного обучения и оптимизации.

2.6 Differential Evolution из библиотеки scipy.optimize

SciPy – одна из самых популярных библиотек Python, содержащая в себе множество математических методов и функций, созданных с помощью модуля NumPy.

Благодаря тому, что SciPy реализован на Python, это позволяет создавать сложные программы и специализированные приложения. SciPy содержит множество подпакетов, которые помогают решить наиболее распространенные проблемы, связанные со сложными вычислениями. Библиотека также простота в использовании.

Модуль scipy.optimize позволяет осуществлять работу с разного рода алгоритмами машинного обучения:

• Условная и безусловная минимизация скалярных функций нескольких переменных, реализующаяся с помощью алгоритмов симплекс Нелдера-Мида, Бройдена-Флетчера-Гольдфарба-Шанно (BFGS), сопряженных градиентов Ньютона, Constrained Optimization By Linear Approximation (COBYLA) и Алгоритм последовательного программирования методом наименьших квадратов (SLSQP);

• Глобальная оптимизация: basin-hopping и дифференциальная эволюция;

• Минимизация остатков: Метод наименьших квадратов (МНК) (least_squares) и алгоритмы подгонки кривых нелинейным МНК (curve_fit);

• Минимизации скалярной функций одной переменной (minim_scalar) и поиск корней (root_scalar);

• Многомерные решатели системы уравнений (root) с использованием различных алгоритмов.

Так как в контексте нашей задачи предметом интереса является нахождение глобального оптимума целевой функции, то подробнее рассмотрим работу алгоритма дифференциальной эволюции.

Differential Evolution (DE) – алгоритм оптимизации, основанный на принципе эволюции, где путем итеративных попыток возможное решение улучшается с учетом заданного показателя качества.

DE применяется в задачах, где необходимо найти оптимальные значения для нескольких параметров, в многомерных вещественных функциях. Так как алгоритм не использует информацию о градиенте при поиске, как того требуют классические методы оптимизации, DE хорошо подходит для недифференцируемых нелинейных целевых функций, а также работает шумами [16].

Дифференциальная эволюция работает за счет поддержания совокупности возможных решений, представленных в виде веществennозначных векторов, и создает новые возможные решения, объединяя уже существующие, а затем сохраняя лучший набор параметров.

Базовый вариант алгоритма DE работает за счет наличия совокупности возможных решений, называемых агентами. Эти агенты перемещаются в пространстве поиска с помощью простых математических формул для объединения позиций существующих агентов из популяции. Если новое положение агента является улучшением, то оно принимается и становится частью популяции, в противном случае новое положение просто отбрасывается. Весь процесс поиска повторяется заново.

Существует несколько стратегий для создания пробных кандидатов [14]. На основе выбранной стратегии отбирается заданное число случайных или

лучших наборов параметров и строится вектор разности, который и используется для получения нового кандидата.

Эти операции позволяют исследовать пространство поиска оптимального решения и сходиться к нему. Дифференциальная эволюция позволяет находить глобальные оптимумы целевой функции, что делает его полезным для задач оптимизации с множеством локальных экстремумов.

Однако такие метаэвристические методы, как DE, которые делают мало предположений об оптимизируемой задаче или не делают вовсе, не всегда гарантирует сходимость к глобальному оптимуму, особенно на сложных и многоэкстремальных функциях, а на некоторых сложных задачах может работать менее эффективно, чем другие алгоритмы. Таким образом, оптимальное решение может быть вообще не найдено.

2.7 Particle Swarm Optimization из библиотеки PySwarms

Библиотека PySwarms полностью посвящена реализации алгоритма Particle Swarm Optimization (PSO) и инструментов, позволяющих реализовать удобное взаимодействие с ним.

PSO или алгоритм роя частиц – также является эвристическим методом поиска оптимальных параметров, вдохновленным поведением косяков рыб, стаи птиц и колоний муравьев. Алгоритм оптимизирует задачу путем итеративных попыток улучшить возможное решение с учетом заданного показателя качества. Роем называют набор потенциальных решений, а частицы являются самими возможными решениями задачи.

Рой частиц перемещается по пространству поиска, обновляя свои позиции на основе данных от ранее разведанных территорий и значений, полученный из новой области. PSO работает по методу exploaration и exploitation, балансируя между поиском новых областей пространства и уточнением текущих наилучших решений.

Движение частиц по пространству решений происходит под влиянием друг на друга на основе их собственного опыта и опыта наилучшей частицы в окрестности. Каждая частица в PSO имеет свою текущую позицию и скорость, и изменяет их в соответствии с двумя основными величинами: лучшей позицией, которую она сама достигла (локальное оптимальное решение) и лучшей позицией, достигнутой любой частицей в окрестности (глобальное оптимальное решение).

Так как PSO — это метод оптимизации без производных, то о не использует градиент функции для поиска решений, а следовательно алгоритм может применяться для решения задач оптимизации с недифференцируемыми, прерывистыми или зашумленными целевыми функциями.

Также PSO выполняет поиск в пространстве решений, используя несколько частиц одновременно, что позволяет лучше исследовать пространство поиска и потенциально ускорить конвергенцию. Преимущества метода роя частиц включают его способность к поиску глобального оптимума, адаптивность к изменяющейся среде и относительную простоту реализации.

Метод Роя частиц предназначен для поиска глобального оптимума задачи. В этом заключается преимущество метода и его недостатки. Как и во всех эвристических методах, в PSO есть вероятность застрять в локальном оптимуме из-за ограниченной способности исследования пространства поиска решений [23]. Метод exploaration и exploitation должен бороться с этой проблемой, но все равно остается вероятность не прийти к оптимальному решению.

К тому же скорость сходимости метода роя частиц может быть недостаточно быстрой для некоторых сложных задач оптимизации. В случае большого количества частиц метод PSO может потреблять большое количество вычислительных ресурсов, особенно при работе с высокоразмерными пространствами поиска решений [23].

Также проблемой может являться настройка параметров алгоритма таких как коэффициент инерции, локальный и глобальный коэффициенты усиления. Неправильная настройка параметров может привести к ухудшению результатов, а поскольку поведение метода роя частиц зависит от случайного движения частиц, результаты могут варьироваться между разными запусками, что может затруднить повторяемость результатов.

2.8 Описание расчетной модели работы сети

Обозначения, используемые для описания модели поведения системы сбора и транспортировки нефти:

- Матрица смежности H;
- Число узлов сети k;
- Число связей (ребер) между узлами сети e;
- Число скважин b;
- Список связей между узлами $edges$;
- Давления на стоке p_{st};
- Набор давлений p_i на узлах сети;
- Набор потоков Q_i на ребрах сети;
- Константы $const$ и α для расчета кривой притока на скважинах;
- Матрица длин труб X;
- Точность расчета для приближения $epsilon$;
- Матрица диаметров труб d;
- Матрица шероховатостей труб μ.

На вход модели подается матрица смежностей, а на выходе получаем значения давлений на узлах, потоков в трубах и величину целевой функции.

Расчет целевой функции производится по формуле (2.1).

$$K_{tar} = m_c - k \sum_{1}^{e} \frac{X_i \cdot d_i}{\mu_i}, \qquad (2.1)$$

где – m_c сумма дебетов на скважинах, k – коэффициент удельной стоимости (в нашей задаче $k = 1$).

Для того, чтобы рассчитать искомую функцию и найти физические величины в сети, необходимо из матрицы смежностей извлечь информацию о топологии системы: количестве ребер, узлов, связях между ними и какие из них являются скважинами.

Матрица смежностей состоит из нулей с единицами на тех местах, где номера связанных узлов и номера строки и столбца совпадают. Следовательно, мы имеем симметричную матрицу относительно главной диагонали. Количество узлов k равно количеству столбцов матрицы смежностей минус единица, так как давление на стоке нам уже известно, и мы не будем считать этот узел, а количество ребер e – числу ненулевых элементов в верхнем треугольнике.

Далее получим список связей между узлами. Мы проходимся по верхнему треугольнику матрицы и ищем ненулевые элементы. Когда число, неравное нулю найдено, мы записываем номер строки в первый столбец $edges$, а номер столбца - во второй, обозначая тем самым направление движения потока. Таким образом, мы проходим через все элементы и получаем полный набор связей нашей сети.

Для того, чтобы посчитать скважины, мы снова воспользуемся симметричностью матрицы и будем двигаться только по верхнему треугольнику. Число скважин равно количеству строк, в которых только одна единица, что эквивалентно тому, что скважины не соединены между собой, а только с внутренними узлами.

Так же в целях подготовки к расчетам, обозначим размерности величин и введем константы:

- Матрица давлений p размером $1 \times k$;
- Набор давлений p_1 на узлах первого приближения размером $k \times 1$ (набор случайных величин в заданном диапазоне);
- Список констант α длиной b;

- Список констант $const$ длиной b;
- Давления на стоке p_{st};
- Матрица длин труб X размером $e \times e$, у которой на главной диагонали стоят значения для каждого ребра;
- Точность расчета для приближения $epsilon$.

Затем для составления системы линейных уравнений мы узнали какие потоки входят в узел, а какие выходят. Для этого написали функцию, которая проходится по каждому внутреннему узлу и присваивает соответствующие значения в зависимости от того потоки от каких узлов входят (положительные), а каких выходят (отрицательные). На основе элементов, описывающих связи, матрицы $edges$ переписали значения новой матрицы $sys_{e-b \times e}$, состоящей из любых дробных чисел. Каждой строке матрицы sys соответствуют коэффициенты вершин графа для одного внутреннего узла.

Система линейных уравнений (СЛУ) в общем виде без учета циклов внутри сети (то есть $k = e$) представлена формулой (2.2):

$$\begin{cases} Q_{1\dots b} = \alpha_{1\dots b} p_{1\dots b} + const_{1\dots b} \\ \sum_{l=1}^{n} Q_{b+1\dots k_l} = 0 \\ p_i - p_j = \dfrac{X_{ii}\mu_{ii}}{d_{ii}} Q_{1\dots e-1_i} \\ p_k - p_{st} = \dfrac{X_{ee}\mu_{ee}}{d_{ee}} Q_e \end{cases} \qquad (2.2)$$

где n – количество ребер (потоков), пересекающихся в узле.

Первые b уравнений описывают приток на скважинах. Следующие $e - b$ гласят, что алгебраическая сумма сходящихся в узле потоков должна равняться нулю. Далее $e - 1$ уравнений показывают изменение давлений на ребрах, которых зависят от физических параметров труб. Последнее уравнение также говорит о падении давления на ребре, но с учетом того, что конечное давление на стоке нам известно.

В матричном виде СЛУ принимает вид формулы (2.3):

$$L_{k+e\times k+e} \cdot PQ_{k+e\times 1} = R_{k+e\times 1} \quad (2.3)$$

Для случая, когда в сети отсутствуют циклы то есть, количество узлов равно количеству ребер, можно разбить СЛУ на две подсистемы: нахождение потоков (формула (2.4)) и давлений (формула (2.5)).

$$C_{k\times k} \cdot Q_{e\times 1} = D_{k\times 1} \quad (2.4)$$

$$A_{e\times e} \cdot P_{k\times 1} = B_{e\times 1} \quad (2.5)$$

Такое обозначение размерностей матрицы может быть приемлемым, так как в такой сети $k = e$. Если в системе появляются лупинги, то разделять систему (3) на подсистемы (4) и (5) нельзя, тогда размерности матриц и неизвестных нарушаются.

Для записи всех коэффициентов в матрицу L ввели матрицы диаметров труб d и шероховатостей μ размерами $e \times e$, заполненных нулями, а на главной диагонали находятся значения физических величин соответствующей номеру строки или столбца трубы.

Далее записали коэффициенты левой части СЛУ в матрицу L, заполненную нулями. Первые b строк соответствуют уравнениям, описывающим кривые притока, эти строки заполняются единицами по диагонали. Следующие $e - b$ строк заполняются на основе матрицы sys, в которой мы описали поведение сходящихся в узле потоков. Мы искали все целочисленные элементы матрицы sys и в столбец, соответствующий абсолютному значению элемента, вписываем 1 или -1 в зависимости от того положительным или отрицательным было число.

Затем заполнили оставшиеся e строк. Первые e столбцов отвечают за коэффициенты перед Q_i и равны $-\frac{X_{ii}\mu_{ii}}{d_{ii}}$. Оставшиеся k столбцов отвечают за коэффициенты перед p_i и записываются на основе матрицы sys так же, как и для потоков в предыдущем абзаце.

Перед тем, как начать заполнять матрицу R ввели константы для начала расчетного цикла, так как значения R будут изменяться на каждом шаге цикла, пока решение не сойдется к удовлетворяющему нас ответу. Создали вектор

$raz\_p_{1\times k}$, в который будем записывать разность между давлениями, найденными на текущем шаге и предыдущем, и переменную $delta$, которой перед входом в цикл присвоили любое ненулевое значение, а во время будем переписывать ее значение на максимальное число из вектора raz_p.

Запустили цикл и начали запись коэффициентов в R. Первые b строк равны $\alpha p_{1_i} + const$ и на каждом шаге цикла их значения менялось так как вектор p_1 заполнялся новыми значениями давлений, полученными после расчета. Все остальные значения матрицы равны нулю, кроме последнего элемента, который равен p_{st}.

После того как правая и левая матрица СЛУ заполнены, решили эту систему уравнений и получили значения потоков (первые e значений) и давлений (следующие k значений) PQ. Нашли разницу между старыми и новыми значениями давлений, записали новую $delta$ и присвоили новые значения давлений p_1. Цикл завершился, когда $delta$ стало меньше $epsilon$, и мы получили потоки и давления в сети с заданной точностью $epsilon$.

В конце расчетного модуля вычислили целевую функцию сети. Для этого просуммировали все потоки, выходящие из скважин. Это общий дебет m_c. Далее вычислили целевую функцию по уже известной формуле (2.1).

Таким образом, подав на вход функции только матрицу смежностей, мы можем рассчитать физические процессы, которые происходят при транспорте нефти, найти давление на узлах, потоки на ребрах и на выходе получить значение целевой функции для данной сети.

2.9 Описание интеграции оптимизационных алгоритмов

Для начала оптимизации транспортной сети нам было необходимо написать вспомогательные программы для интеграции алгоритмов в расчетный модуль. Нам нужно было унифицировать вид подаваемых параметров для оптимизации целевой функции, гиперпараметров алгоритмов оптимизации и функций для настройки и запуска этих алгоритмов.

Мы начали с подготовки конфигурационного файла параметров для оптимизации целевой функции. Файл представляет собой словарь, где ключами являются имена параметров, а значения каждого ключа – описание ограничений этого параметра. По кодовым словам «min», «max», «step» и «type» указываются минимальное и максимальное значение переменной, шаг поиска решения и её тип соответственно.

Далее создали конфигурационный json-файл для гиперпараметров алгоритмов. Он также представляет собой словарь, где ключами являются названия оптимизационных алгоритмов. По каждому ключу хранятся названия гиперпараметров и список их возможных значений Эти записи тоже представлены в виде словаря.

Затем описали большой модуль, где будет происходить настройка работы алгоритмов. Для начала мы написали несколько вспомогательных операций, которые включают в себя логирование, открытие и чтение конфигурационных файлов с параметрами и гиперпараметрами, их запись в новые переменные и присоединили расчетную модель через exe-файл. Написали декоратор, который отслеживает время работы алгоритма, и функцию, которая создает список словарей, содержащих все возможные комбинации гиперпараметров, что эквивалентно grid-search.

Потом создали функцию для каждого алгоритма, где происходит встраивание в них прочитанных оптимизационных параметров, а на выходах функций мы получаем значение целевой функции, расчет которой присоединили ранее. Затем также для каждого алгоритма написали функцию запуска оптимизации. Здесь происходит загрузка оптимизируемых параметров и их ограничений, создается экземпляр, задается режим работы и присваиваются все необходимы параметры алгоритма. Функции обернуты в декоратор, который отслеживает их время работы. На выходе мы получаем лучшее значение целевой функции и параметров.

Таким образом, с помощью написанной программы можно легко задать оптимизируемые параметры сразу для всех алгоритмов, а единый вид для

гиперпараметров и способ чтения позволяет не прописывать их вручную для каждой функции. Также на выходе из каждой функции мы получили одинаковые типы искомых решений, что очень удобно для представления ответов в едином формате.

2.10 Описание параметров и гиперпараметров оптимизационных алгоритмов

Теперь перейдем к настройке параметров и гиперпараметров. В процессе работы программа сама определяет наиболее эффективные гиперпареметры для каждого алгоритма, а параметры мы задаем сами. Поэтому начнём с описания параметров и их значений, которые были выбраны для исследования.

Для запуска алгоритма TPE из библиотеки Optuna мы создали объект Study, который управляет оптимизацией. Мы задали только два параметра: sampler и direction. Первый отвечает за реализацию выбранного алгоритма оптимизации. Мы выбрали TPESampler, у которого тоже есть свои настройки относительно выборки кандидатов, и передали ему информацию об оптимизируемых параметрах: границы, шаг и тип. Второй определяет направление функции. Так как мы решаем задачу минимизации, то передаём значение «minimize». Все остальные параметры, отвечающие за хранение, имена и остановку алгоритма остаются по умолчанию. В optimize мы передали интеграционную функцию для optuna, которая возвращает значение целевой функции, и количество испытаний, равное 100.

Optuna имеет три логических гиперпараметра, которые мы искали для эффективной работы алгоритма. Первый гиперпараметр – multivariate, указывает на то, как оптимизируются переменные: по очереди (False) или все одновременно (True). Второй - consider_magic_clip управляет тем, будет ли алгоритм избегать экстремальных значений таких как inf, -inf или nan, полученных из-за некорректных оценок. Если гиперпараметр имеет значение

True, то при поиске решений такие магические числа будут учитываться. И последний - consider_endpoints отвечает за то, учитываются ли краевые значения параметров при поиске (True) или нет (False).

Для настройки TPE из библиотеки HyperOpt мы создали дополнительную переменную, куда сложили границы и шаг оптимизируемых параметров. Затем мы создали объект класса Trials, чтобы хранить информацию о каждом испытании и брать из нее необходимые данные. В оптимизацию fmin мы загружаем fn, которой присваиваем вспомогательную функцию, возвращающую целевую, space, равную новой переменной с данными об оптимизируемых параметрах, algo, где выбираем реализуемым алгоритмом TPE. Так же записываем количество выполнений функции в max_evals равным 100. В параметр хранения оценок trials записываем экземпляр класса Trials. Последний параметр early_stop_fn говорит о том сколько испытаний без изменения значения целевой функции должно пройти, чтобы можно было прервать вычисления и назвать ее значение оптимальным. В нашей задаче он равен 20.

В библиотеке HyperOpt мы находим значение одного гиперпараметра – allow_trials_fmin. Он указывает на то, выбирать ли испытания для функции fmin() только из имеющихся результатов (True) или же использовать только новые значения, исключая результаты, полученные ранее (False).

Для работы с алгоритмом дифференциальной эволюции мы создали переменную bounds, которой присвоили граничные значения параметров. Далее идет запуск алгоритма, и в функцию расчета передаются параметры. Первым идет вспомогательная функция для дифференциальной эволюции, которая возвращает значение целевой функции. Следующий параметр maxiter имеет значение 40 и характеризует максимальное число поколений, за которое эволюционирует вся популяция. Popsize – это множитель для определения общей численности популяции. Популяция содержит popsize * len(x) особей, где x – это набор оптимизируемых параметров в виде одномерного массива. В

нашей задаче popsize = 15. Disp = True выводит вычисленную функцию на печать на каждой итерации.

Значения остальных параметров используются по умолчанию. Strategy, который отвечает за выбор пути мутации, равен best1bin. Это означает, что в стратегии для генерации новых кандидатов путем комбинации двух других особей будет использоваться лучший индивид из текущего поколения и случайно выбранный из популяции. Новая особь сравнивается с лучшей, и если она таковой является, то заменяет ее.

Алгоритм DE имеет два гиперпараметра: mutation и recombination. Когда кандидат выбирается для мутации, в соответствии с вероятностью мутации изменяется значение одного из параметров, содержавшегося в векторе решений. Вероятность мутации не фиксирована. Процесс начинается с высокой вероятности мутации, которая будет постепенно уменьшаться, пока не достигнет значений вероятности, близких к обратным значениям численности популяции в конце запланированного числа поколений. В нашем исследовании мы задали два промежутка для выбора коэффициента мутации: [0,5, 1] и [0,6, 1.2]. Первой число в паре обозначает нижнюю границу вероятности, а второе – верхнюю.

Далее идет гиперпараметр рекомбинации, который определяет вероятность перемешивания векторов кандидатов при генерации новых потомков. Чем больше значение рекомбинации, тем интенсивнее идет смешивание информации родителей, тем более разнообразным получается новой потомство. В нашей задаче мы выбрали 0,7 и 0,8 как возможные вероятности рекомбинации.

Перед работой с алгоритмом роя частиц, в переменную bounds также загрузили границы наших оптимизируемых параметров. Далее создали объект оптимизации ps.single.GlobalBestPSO. Ps.single означает, что оптимизация происходит с использованием одной частицы независимо от всех остальных, а GlobalBestPSO обозначает метод, в котором все частицы в рое обмениваются информацией о найденных решениях (глобальные решения), и их

перемещение в пространстве определяется как комбинация их собственного лучшего значения и глобального лучшего значения среди всех частиц.

Параметр n_particles = 40, определяющий количество частиц в рое, является первым в ps.single.GlobalBestPSO. Далее задается значение dimensions, которое указывает на размерность пространства решений, оптимизируемого в задаче. Мы задали его как длину вектора ключей из файла с оптимизируемыми параметрами. В options мы передаем param, где содержатся дополнительные сведения о параметрах. И в последним передаем в bounds значение переменной, которую мы объявили ранее, определяющую границы изменения параметров.

К полученному объекту применим функцию, которая запускает в работу алгоритм PSO. Она принимает в себя вспомогательную программу pyswarms, которая возвращает целевую функцию, число итераций iters = 25.

Для настройки поведения частиц используются гиперпараметры c1, c2 и w. Число c1, принимающее в нашей задаче значение 0,5 или 0,6, называется коэффициентом социального влияния и определяет важность личного опыта, полученного частицей. Чем выше его значение, тем более важную роль играет информация о лучшем решении, найденном самой частицей. Число c2 – коэффициент ускорения когнитивного влияния, который определяет, насколько частицы склонны двигаться в направлении лучшего положения в группе. Мы присвоили ему значения 0,3 или 0,4. Величина c2 аналогична c1: чем больше значение, тем сильнее влияние свойства коэффициента, но в случае с c2 уже имеется в виду важность глобального лучшего решения в рое.

W – это коэффициент инерции, который управляет влиянием предыдущей скорости частицы на ее будущую скорость. Низкое значение w означает уменьшение влияния инерции, что приводит к более медленному движению частиц. В нашем исследовании коэффициент равен 0,8 или 0,9.

Исходя из всего вышесказанного, мы получили описание настроек каждого алгоритма, типы, способы и значения принимаемых ими параметров

и гиперпараметров, что дает полную картину для понятия свойств и принципов работы выбранных алгоритмов.

ГЛАВА 3. РЕЗУЛЬТАТЫ

3.1 Оценка работы модели системы сбора и транспортировки нефти

В предыдущей главе был описан принцип работы расчетной модели системы сбора и транспорта нефти. Теперь мы применим ее для расчета давлений, потоков и целевой функции простой сети.

Топология исследуемой сети представлена на рисунке (3.1):

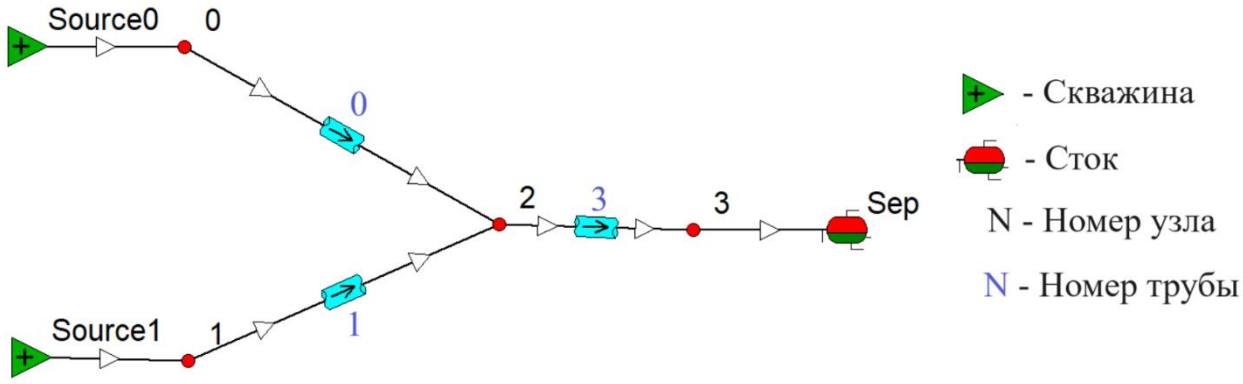

Рис. 3.1 Схема топологии ССиТ нефти

Матрица смежности такой системы выглядит так:

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Данные, которые мы получили из H, равны:

- Число узлов $k = 3$;
- Число ребер $e = 3$;
- Число скважин $b = 2$;
- Список связей между узлами $edges = [(0, 2), (1, 2), (2, 3)]$.

Матрица sys, описывающая направление движения потоков в узлах, имеет вид:

$$sys = [[0, 1, -2]]$$

Далее мы задали константы и физические свойства труб для начала расчета:

- Набор давлений p_1 на узлах первого приближения примем 15 атм;

- Константы $\alpha_i = -1$ и $const_i = 80$ для всех скважин;
- Давления на стоке $p_{st} = 10$ атм;
- Все длины труб $X_{ii} = 1$ км;
- Все диаметры $d_{ii} = 159$ мм;
- Все шероховатости $\mu_{ii} = 0{,}15$ мм;
- Точность расчета для приближения примем $epsilon = 0{,}0001$.

Система линейных уравнений для данной топологии представляется формулой (3.1):

$$\begin{cases} Q_0 = \alpha_0 p_0 + const_0 \\ Q_1 = \alpha_1 p_1 + const_1 \\ Q_0 + Q_1 - Q_2 = 0 \\ p_0 - p_2 = \dfrac{X_{00}\mu_{00}}{d_{00}} Q_0 \\ p_1 - p_2 = \dfrac{X_{11}\mu_{11}}{d_{11}} Q_1 \\ p_2 - p_{st} = \dfrac{X_{22}\mu_{22}}{d_2} Q_2 \end{cases} \qquad (3.1)$$

Далее мы заполнили левую L и правую R части матричного уравнения, представленного формулой (2.3). Матрицы коэффициентов СЛУ имеют вид:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 \\ -0.001 & 0 & 0 & 1 & 0 & -1 \\ 0 & -0.001 & 0 & 0 & 1 & -1 \\ 0 & 0 & -0.001 & 0 & 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 65 \\ 65 \\ 0 \\ 0 \\ 0 \\ 10 \end{pmatrix}$$

Мы построили все необходимые матрицы и решили систему уравнений (2.3). Найденные значение искомых величин представлены в таблице (3.1).

Таблица 3.1

Давления p на узлах и потоки Q на ребрах в ССиТ нефти

№ узла/ребра	Давление p, атм	Поток Q, кг $\cdot \frac{\text{м}}{\text{с}}$
0	10,20	69,8
1	10,20	69,8
2	10,13	139,6
3	10,00	

Для проверки точности реализованного алгоритма сеть, представленная на рисунке (3.1), была смоделирована в программе Gap.exe, которая используется для расчета сетей в промышленных масштабах. Результаты, полученные путем моделирования в Gap, представлены в таблице (3.2).

Таблица 3.2

Давления p на узлах и потоки Q на ребрах в ССиТ нефти, полученные в программе Gap

№ узла/ребра	Давление p, атм	Поток Q, кг $\cdot \frac{\text{м}}{\text{с}}$
0	10,25	70
1	10,25	70
2	10,13	140
3	10,00	

Сравнивая величины из таблиц (3.1) и (3.2), можно сказать, что построенная нами модель может описывать свойства системы сбора и транспортировки нефти с точностью допустимой погрешности.

После сравнения результатов мы рассчитали значение целевой функции по формуле (2.1) в исследуемой топологии равно:

$$K_{tar} = -2900{,}8$$

3.2 Оценка работы оптимизационных алгоритмов и полученные гиперпараметры

Для топологии, представленной на рисунке 3.1, мы нашли оптимальные параметры диаметров d и шероховатостей μ для получения наилучшего значения целевой функции.

Краевые значения параметров были взяты согласно ГОСТу [5], где указаны правила технологического проектирования магистрального трубопроводного транспорта нефти и нефтепродуктов. Их значения представлены в таблице (3.3).

Таблица 3.3

Минимальное, максимальное значения и шаг дискретизации для нахождения оптимальных значений параметров

Параметр	Минимальное значение, мм	Максимальное значение, мм	Шаг исследования, мм
d	159	1220	10
μ	0,10	0,15	0,025

В таблице (3.4) отражены результаты работы оптимизационных алгоритмов и нахождения гиперпаметров.

Таблица 3.4

Алгоритм, оптимальные параметры, значение целевой функции, время работы алгоритма и полученные гиперпараметры

Алгоритм	Набор параметров $[d_0, d_1, d_2]$, $[\mu_0, \mu_1, \mu_2]$	Целевая функция, K_{tar}	Полное время работы, ч	Гиперпараметры
Optuna	[159, 159, 159], [0,15, 0,15, 0,15]	- 2900,8	3,15	multivariate: false, consider_magic_clip: false, consider_endpoints: true

Алгоритм	Набор параметров $[d_0, d_1, d_2]$, $[\mu_0, \mu_1, \mu_2]$	Целевая функция, K_{tar}	Полное время работы, ч	Гиперпараметры
HyperOpt	[159, 159, 159], [0,15, 0,15, 0,15]	- 2900,8	0,26	allow_trials_fmin: true
Difevolution	[159, 159, 160], [0,15, 0,15, 0,15]	- 2907,5	18,4	mutation: [0,5, 1,0], recombination: 0,8
PSO	[170, 214, 265], [0,15, 0,15, 0,15]	- 4047,2	17,0	c1: 0,5, c2: 0,3, w: 0,8

В таблице (3.4) представлены гиперпараметры, используя которые алгоритмы рассчитали оптимальное значение для целевой функции. Применение полученных гиперпараметров позволяет сократить время работы алгоритмов.

Таким образом, исходя из таблицы (3.4), Optuna представляет собой алгоритм, поочередно оптимизирующий точки, включающий краевые значения, но избегающий экстремальных чисел, а HyperOpt для новых испытаний выбирает кандидаты из старых значений. Лучшими значениями для работы DE стали меньшие значение мутации и большее рекомбинации. В рое частиц личный опыт играет не самую важную роль, но всё же больше, чем информация от глобального лучшего решения в рое, а влияние инерции на скорость движения в последующих итерация не такое высокое, каким могло бы быть.

Из полученных результатов мы можем видеть, что практически все алгоритмы, кроме PSO, оптимальным значением находят минимальный размер диаметра и максимальную величину шероховатости, что согласно формуле (2.1) логично, ведь чем больше диаметр и меньше шероховатость трубы, тем больше расходы, следовательно меньше целевая функция.

Так как величина целевой функции, полученная после работы алгоритма роя частиц, отличается от значений, рассчитанных другими алгоритмами, аппроксимируем точки, полученные в ходе расчета степенной функцией, чтобы проследить тренд движения вычислений.

Значения целевой функции в зависимости от времени расчета при работе алгоритма PSO, линия тренда и оптимальное значение, полученное вследствие работы остальных алгоритмов, представлены на рисунке (3.2).

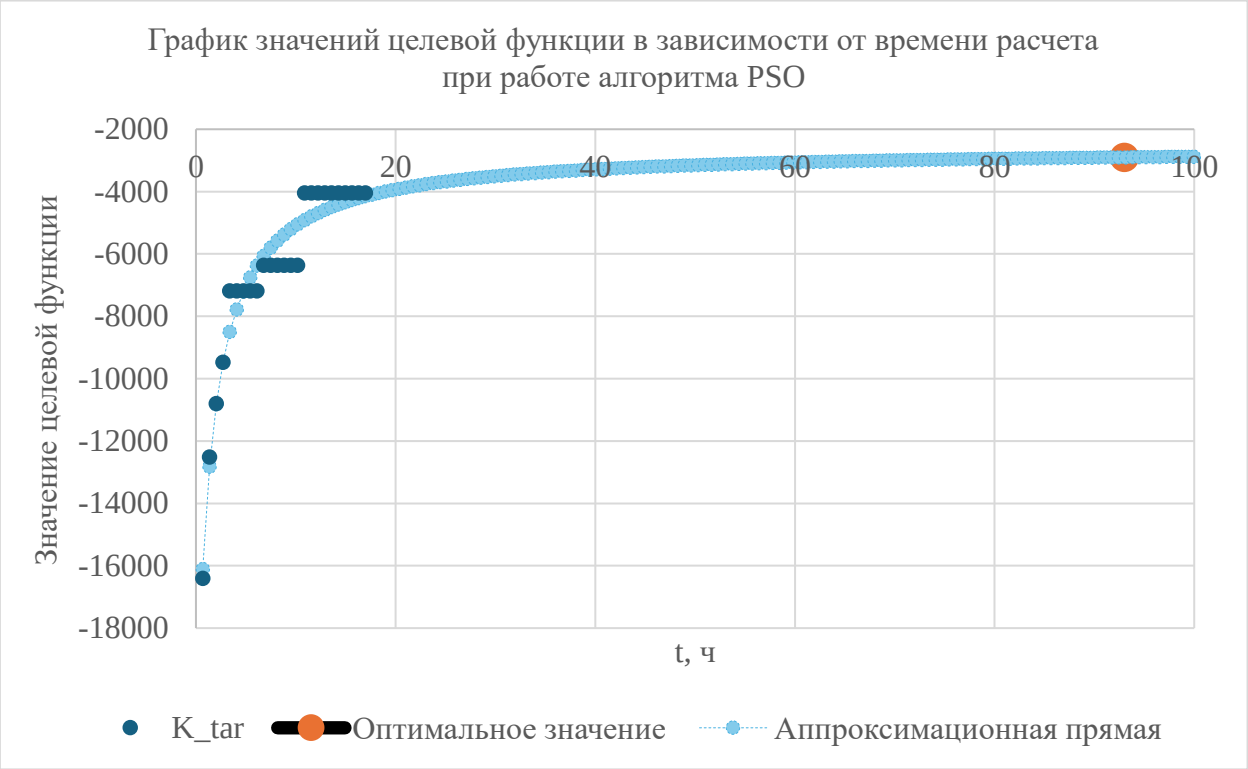

Рис. 3.2 Значения целевой функции в зависимости от времени расчета при работе алгоритма PSO

Из Рисунка (3.2) и отображенной на нем аппроксимации видно, что промежуточные значения целевой функции, подчиняясь степенному закону, стремятся к максимальному значению. Также, согласно аппроксимации, мы

можем оценить, что оптимальное значение целевой функции, которое мы получили в результате работы других алгоритмов, возможно будет достигнуто после 93 часов работы вследствие увеличения либо количества частиц начального роя, либо числа итераций.

Согласно ГОСТу [5] в алгоритмы, которые работают с дискретными значениями переменной, заданными списком, для оптимизации были поданы диаметры труб, которые могут быть изготовлены.

$d = [159, 219, 273, 325, 377, 426, 530, 630, 720, 820, 1020, 1220]$ мм

В таблице (3.5) отражены результаты работы оптимизационных алгоритмов и найденных гиперпаметров.

Таблица 3.5

Алгоритм, оптимальные параметры, значение целевой функции, время работы алгоритма и полученные гиперпараметры

Алгоритм	Набор параметров $[d_0, d_1, d_2]$, $[\mu_0, \mu_1, \mu_2]$	Целевая функция, K_{tar}	Время работы, с	Гиперпараметры
Optuna	[159, 159, 159], [0,15, 0,15, 0,15]	- 2900,8	1,26	multivariate: false, consider_magic_clip: false, consider_endpoints: true
HyperOpt	[159, 159, 159], [0,15, 0,15, 0,15]	- 2900,8	2,30	allow_trials_fmin: true

Из таблицы (3.5) мы можем увидеть, что алгоритмы, которые могут с оптимизируемыми параметрами, заданными списком, выполняют вычисления за несколько секунд и получают то же самое оптимальное значение целевой функции.

Такое преимущество в скорости вычислений связано с тем, что, задавая только верхнюю и нижнюю границы, алгоритмы вынуждены были искать решение среди множества чисел этого промежутка. Подаваемый список сократил количество перебираемы параметров в 88, тем самым дав выигрыш по времени в 9000 и 400 раз для Optuna и HyperOpt соответственно.

Для проверки эффективности алгоритмов найдем оптимальные параметры для более сложной сети. Топология новой сети представлена на рисунке (3.3):

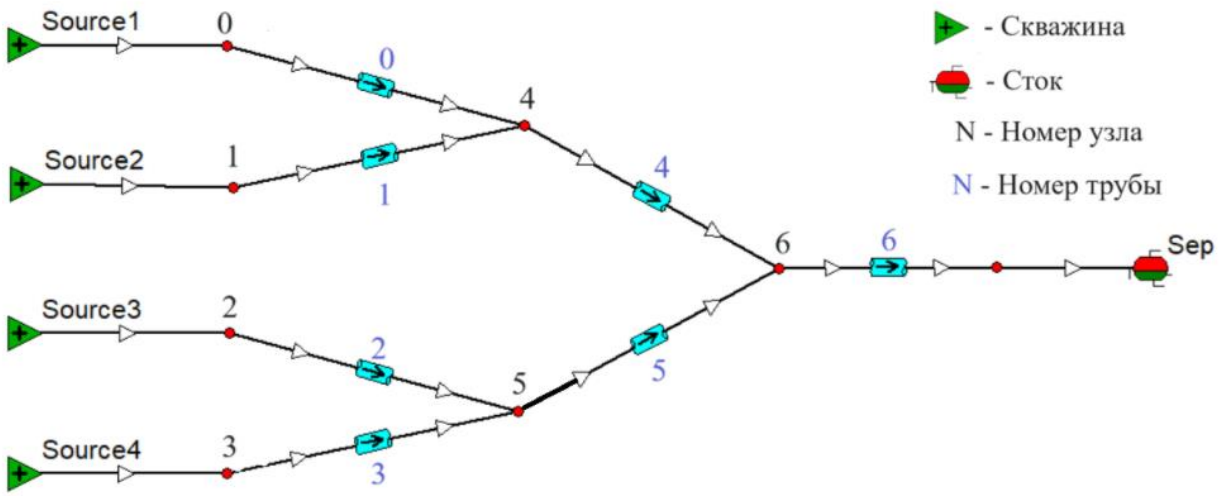

Рис. 3.3 Схема топологии ССиТ нефти

В таблице (3.6) отражены результаты работы оптимизационных алгоритмов и найденные параметры.

Таблица 3.6

Алгоритм, оптимальные параметры, значение целевой функции, время работы алгоритма и гиперпараметры

Алгоритм	Набор параметров $[d_0, d_1, d_2, d_3, d_4, d_5, d_6]$, $[\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6]$	Целевая функция, K_{tar}	Время работы, ч	Гиперпараметры
Optuna	[159, 159, 159, 159, 159, 159, 159],	- 6585,5	1,37	multivariate: false, consider_magic_clip: false, consider_endpoints: true

Алгоритм	Набор параметров $[d_0, d_1, d_2, d_3, d_4, d_5, d_6]$, $[\mu_0, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6]$	Целевая функция, K_{tar}	Время работы, ч	Гиперпараметры
	[0,15, 0,15, 0,15, 0,15, 0,15, 0,15, 0,15]			
HyperOpt	[159, 159, 159, 159, 159, 159, 159], [0,15, 0,15, 0,15, 0,15, 0,15, 0,15, 0,15]	- 6585,5	9,20	allow_trials_fmin: true

Таблица (3.6) демонстрирует, что с Optuna и HyperOpt можно оптимизировать более сложные системы сбора и транспортировки в пределах допустимого времени расчета.

ЗАКЛЮЧЕНИЕ

Целью данной работы является проведение оптимизации параметров системы сбора и транспортировки нефтегазового месторождения.

В ходе проведенного исследования был написан программный код, который используя только матрицу смежностей, получал полную информацию о системе, а именно число ребер, количество узлов и скважин и связи между ними. С помощью полученной информации и констант, описывающих физические свойства скважин, была решена система линейных уравнений, которая включала в себя законы перепада давления и движения потоков, и найдены давления в узлах сети и потоки на ее ребрах. Для проверки корректности работы написанной программы было проведено сравнение с моделью системы, построенной в промышленном симуляторе Gap. Итогом сравнения являлось утверждение, что наша модель описывает ССиТ нефти с точностью допустимой погрешности. Далее мы вычислили целевую функцию.

Для выполнения задачи интеграции оптимизационных алгоритмов в расчетный модуль были написаны вспомогательные программы, которые структурировали и унифицировали подаваемые оптимизационные параметры и гиперпараметры. Свойства параметров были записаны с помощью ключевых слов «min», «max», «step» и «type». Гиперпараметры были представлены в виде json-файла с ключами в виде имен алгоритмов. Для каждого алгоритма написали функции запуска алгоритма, которую обернули в декоратор отслеживания времени работы, и вычисления целевой функции. Все вышеперечисленные операции позволили упростить ввод данных: он производится в едином формате для всех алгоритмов, и результаты работы алгоритмов также в унифицированном виде записываются в json-файл, что позволяет эффективно сравнивать полученные значения.

После проведения расчётов мы получили оптимальные значения гиперпагаметров для каждого алгоритма, которые позволяют оптимизаторам быстрее сходиться к лучшему решению. Также по значениям гиперпараметров

мы смогли описать принципы работы алгоритмов и аспекты, влияющие на поиск оптимальных решений.

Также после проведения расчетов мы получили оптимальные значения параметров от каждого алгоритма, максимальное значение целевой функции и время их работы: μ=0,15 мм для всех труб и алгоритмов; d=159 мм для всех труб, K_tar=-2900,8 от Optuna (3,15 ч) и HyperOpt (0,26 ч), d= [159, 159, 160] мм, K_tar=-2907,5 для дифференциальной эволюции (18,4 ч) и d= [170, 214, 265] мм, K_tar=-4047,2 для PSO (17,0 ч). Алгоритмы Optuna и HyperOpt сходятся к одинаковому максимальному значению, значение целевой функции в DE очень близко к оптимальному, рассчитанному двумя предыдущими алгоритмами. Так как значение функции в PSO было отличным от всех остальных значений, была построена аппроксимационная кривая на основе данных, полученных в процессе оптимизации, которая дает возможность оценить вероятное время работы до нахождения максимума. До 93 ч время расчета увеличивается с помощью увеличения начального числа частиц или количества итераций.

В целях ускорения работы алгоритмов, которые могут работать с параметрами, представленными в виде списка, в Optuna и HyperOpt были поданы списки значений возможных параметров, взятые из ГОСТа. Значение целевой функции осталось равным -2900,8, но уменьшение пространства поиска в 88 дало нам преимущество в скорости счета: для Optuna расчет выполняется за 1,26 секунды, что в 9000 раз быстрее, а для HyperOpt за 2,3 секунды, что в 400 раз меньше, чем поиск в старом пространстве решений.

Так как Optuna и HyperOpt успешно справились с простой сетью, была оптимизирована более сложная топология, состоящая из 7 узлов, 4 из которых скважины, и 7 ребер. У обоих алгоритмов получились значения μ=0,15 мм и d=159 мм для всех труб, K_tar=-6585,5. Время расчета 1,37 ч и 9,2 ч для Optuna и HyperOpt соответственно.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коршак А. История нефтегазового дела. – Litres, 2022.

2. Cherniaev V. D. et al. Oil transportation //Oil Industry of the Former Soviet Union. – CRC Press, 2019. – С. 185-278.

3. Watanabe S. Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance //arXiv preprint arXiv:2304.11127. – 2023.

4. Ашпов А. С., Димитриади Ю. К., Мурадханов И. В., Черненко К. И., Основы нефтегазового дела. Introduction to Oil-and-Gas Engineering, — Северо-Кавказский федеральный университет, 2017.

5. ГОСТ 34563-2019 Магистральный трубопроводный транспорт нефти и нефтепродуктов. Правила технологического проектирования, 2019.

6. СП 36.13330.2012 Магистральные трубопроводы. Актуализированная редакция СНиП 2.05.06-85, 2013.

7. Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. TProc. of the 30th International Conference on Machine Learning (ICML 2013), June 2013, pp. I-115 to I-23.

8. International Energy Agency, World Energy Balances Database, 2022.

9. Giuliani M. et al. Hybrid artificial intelligence techniques for automatic simulation models matching with field data //Abu Dhabi International Petroleum Exhibition and Conference. – SPE, 2018. – С. D032S175R001.

10. Hojageldiyev D. Artificial intelligence in HSE //Abu Dhabi International Petroleum Exhibition and Conference. – SPE, 2018. – С. D012S120R002.

11. Li D. W., Shi G. R. Optimization of common data mining algorithms for petroleum exploration and development //Acta Petrolei Sinica. – 2018. – Т. 39. – №. 2. – С. 240-246.

12. Li H. et al. Applications of artificial intelligence in oil and gas development //Archives of Computational Methods in Engineering. – 2021. – Т. 28. – С. 937-949.

13. Liu B. J. Construction conception of intelligent oilfield //J Shengli Oilfield Party Sch. – 2015. – Т. 6. – С. 99-101.

14. Qiang J. I. A unified differential evolution algorithm for global optimization. – 2014.

15. Razghandi M., Dehghan A., Yousefzadeh R. Application of particle swarm optimization and genetic algorithm for optimization of a southern Iranian oilfield //Journal of Petroleum Exploration and Production. – 2021. – Т. 11. – С. 1781-1796.

16. Rocca P., Oliveri G., Massa A. Differential evolution as applied to electromagnetics //IEEE Antennas and Propagation Magazine. – 2011. – Т. 53. – №. 1. – С. 38-49.

17. Sircar A. et al. Application of machine learning and artificial intelligence in oil and gas industry //Petroleum Research. – 2021. – Т. 6. – №. 4. – С. 379-391.

18. Wei L., Na Y. Application and influence of artificial intelligence in petroleum engineering area //Oil Forum. – 2018. – Т. 37. – №. 4. – С. 32.

19. FMin - hyperopt, GitHub, 2018. – URL: https://github.com/hyperopt/hyperopt/wiki/FMin. – (дата обращения: 23.10.2023).

20. Hyperopt Documentation, 2018. – URL: https://hyperopt.github.io/hyperopt. – (дата обращения: 23.10.2023) .

21. Optuna: A hyperparameter optimization framework — Optuna 3.6.0 documentation, 2018. – URL: https://optuna.readthedocs.io/en/stable. – (дата обращения: 23.10.2023).

22. Python Optuna: A Guide to Hyperparameter Optimization, datagy, 2023. – URL: https://datagy.io/python-optuna. – (дата обращения: 23.10.2023).

23. PySwarms 1.3.0 documentation, 2017. – URL: https://pyswarms.readthedocs.io/en/latest. – (дата обращения: 25.10.2023).