Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

КУРСОВАЯ РАБОТА

Машинное обучение для настройки параметров блокчейна

Название темы

по направлению подготовки Прикладная математика и информатика

образовательная программа «Науки о данных»

Студент группы

мНОД21_ИССА

Гатиятуллин Эрнест

Ринатович

Ф.И.О.

Руководитель:

Доцент к .ф-м. н.

Янович Юрий Александрович

степень, звание, должность Ф.И.О.

Москва 2022

Содержание

Введение

1.1 Актуальность

Блокчейн обладает рядом настраиваемых параметров, которые задаются при запуске первого блока (генезис блока) и определяют конфигурацию создаваемой сети. Для оценки производительности блокчейна используются такие метрики, как TPS (Transactions-Per-Second), характеризующая пропускную способность сети, и droprate, показывающий долю отклоненных (неуспешных) транзакций. Таким образом, блокчейн можно рассматривать как функцию черного ящика, который на вход принимает вектор параметров X, а на выходе выдает двумерный вектор y с TPS и droprate. Это естественным образом приводит к рассмотрению возможности подбора параметров блокчейна с помощью методов машинного обучения, рассматривая настройку наилучших параметров как задачу регрессии.

В качестве тестовой сети была выбрана Solana – высокопроизводительный блокчейн, архитектура которого построена на гибридном консенсусе, соединяющий модели Proof-of-History (PoH) и Proof-of-Stake (PoS). PoS-консенсус позволяет генерировать новый блок сети тем участникам, которые владеют наибольшим количеством монет (coins). Solana обладает 89 контролируемыми параметрами, такими как размер блока, параметр синхронизации транзакций (размер кучи) и другие [1]. Однако не все параметры одинаково влияют на выходные переменные блокчейна, поэтому появилась идея снижения размерности задачи. Для этого с помощью алгоритма SHAP каждому параметру была присвоена значимость, на основе которой были отобраны 6 наиболее важных:
1) NUM_THREADS; 2) DEFAULT_TICKS_PER_SLOT; 3) ITER_BATCH_SIZE,
4) RECV_BATCH_MAX_CPU, 5) DEFAULT_HASHES_PER_SECOND,
6) DEFAULT_TICKS_PER_SECOND (рис.1).

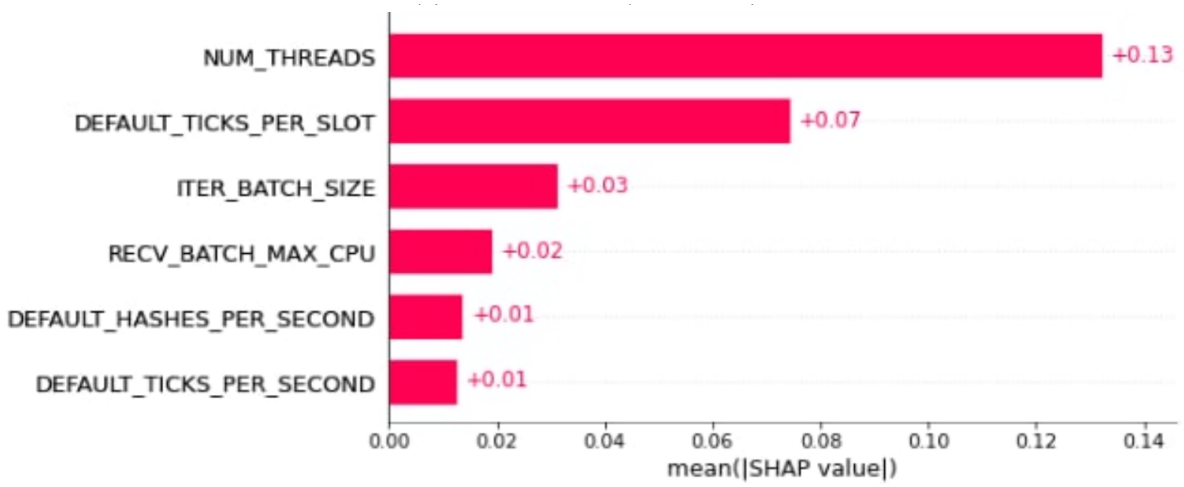

Рис.1. Значимость параметров блокчейна, полученная с помощью алгоритма SHAP

Таким образом, обозначенная задача оптимизации сводится к подбору такого 6-мерного вектора параметров $\theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ при котором на выходе блокчейна будет оптимальный двумерный вектор $y = (y_1, y_2)$.

Суррогатная оптимизация

2.1 Основные принципы

Одним из перспективных методов нахождения оптимальных точек целевой функции, применяемый в данной работе, является суррогатная оптимизация. Ее основная идея заключается в том, что вместо зачастую дорогостоящей оптимизации функции черного ящика использовать заранее обученную регрессионную (суррогатную) модель, которая аппроксимирует целевую функцию [2]. Так как обученная модель является аналитической функцией, то, как правило, оценка суррогатной модели имеет на порядок меньшую вычислительную сложность, позволяя исследовать только те области существования функции, которые с наибольшей вероятностью содержат глобальный оптимум. Сначала суррогатная модель обучается на нескольких точках из области определения функции, в которых также вычисляется значения целевой функции. Одним из самых распространенных выборов модельной функции является гауссовский процесс. Это набор случайных величин, совместное распределение вероятностей любого конечного набора которых является гауссовым (нормальным) распределением (рис.1) [3].

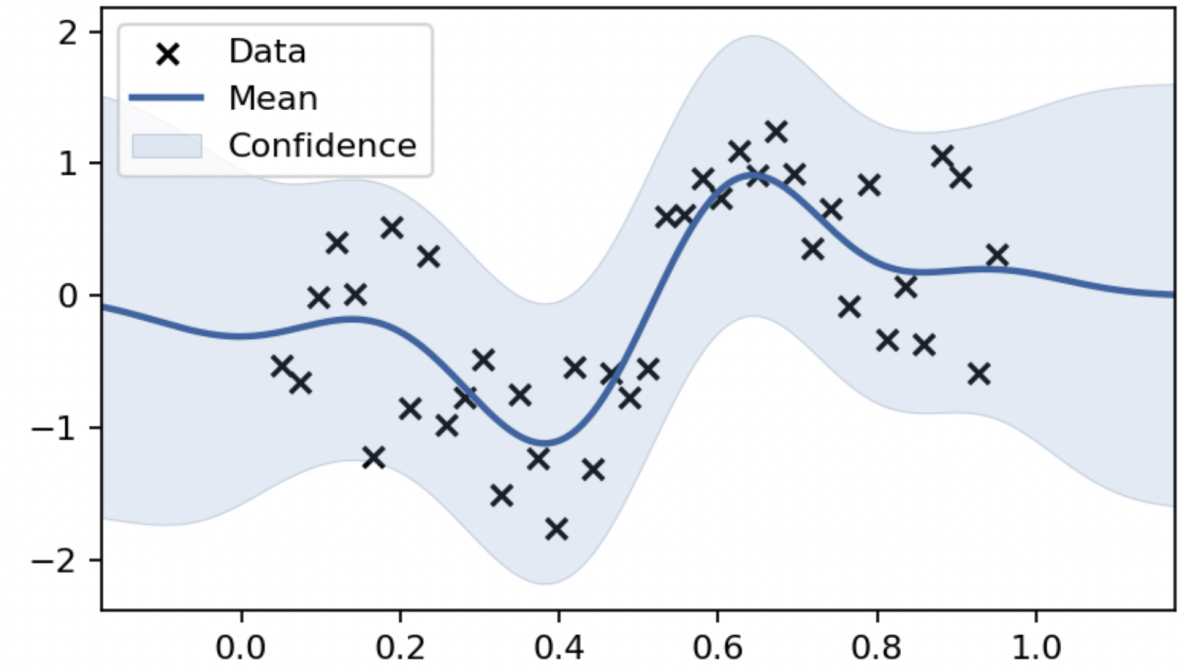

Рис.1. Пример обученного гауссовского процесса. Сплошной синей линией обозначена гауссовская регрессия,черными крестиками – точки, распределенные по синусоидальному закону с наложенным шумом

Следующий этап – это активное обучение, во время которого регрессионная модель дополняется точками, которые имеют наибольшую вероятность улучшить предсказание оптимума. Для определения этих перспективных точек применяется специальная обучающая функция. Отбор новых точек для добавления в обучающую выборку происходит из тех областей, где обучающая функция достигает экстремума. В гауссовском процессе такой функцией является функция ожидаемого улучшения (рис 2):

$$E[I(x)] = \int I(x)p(y)dy\,,$$

где $I(x) = y_{min} - y(x),\ if\ y_{min} < y(x);\ I(x) = 0,\ otherwise,$

$$p(y) \sim N(\mu(x),\ \sigma^2(x))$$

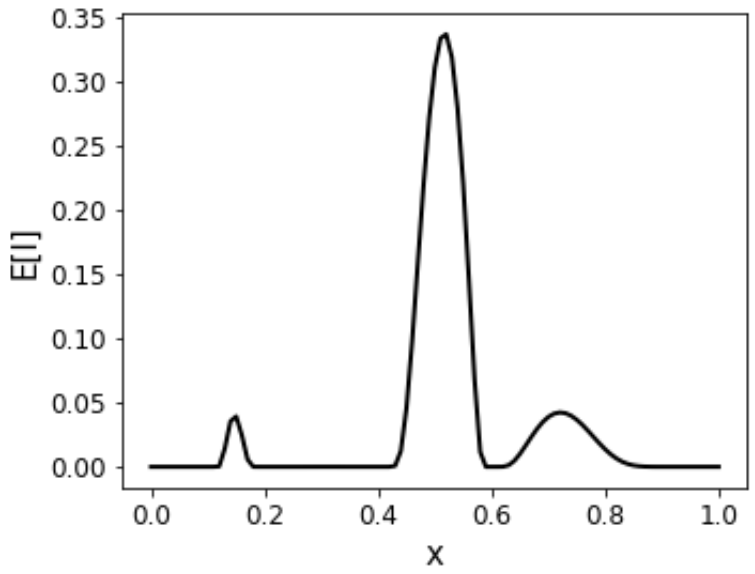

Рис.2. Вид обучающей функции после 1 итерации суррогатной оптимизации. Поскольку она имеет ярко выраженный пик при x = 0.52, то набор обучающих точек должен быть расширен новой точкой, взятой из этой области определения

Подводя итоги, суррогатная оптимизация использует статистическое приближение целевой функции некоторой моделью, такой как, например, гауссовский процесс, для снижения стоимости вычислений и ищет ее оптимальное значение. Для более точной аппроксимации производится оценка ее значений в тренировочных точках из области определения, которые затем дополняются точками только из тех областей, в которых

нахождение экстремума целевой функции имеет наибольшую вероятность. Для нахождения перспективных точек производится поиск максимального значения обучающей функции.

2.2 Планирование эксперимента

Перед началом процесса оптимизации необходимо сгенерировать набор стартовых точек. Начальные точки для оценки целевой функции определяют планирование эксперимента. Хорошо подобранный план имеет решающее значение для того, чтобы суррогатная модель наилучшим образом уловила поведение базовой целевой функции. Одним из методов генерации случайной выборки для многомерных данных является выборка с помощью Латинского гиперкуба [4]. В отличие от обычного случайного семплирования, когда каждое новая сгенерированная случайная величина не зависит от предыдущей, метод Латинского гиперкуба относится к запоминающим методам. Каждое новое значение генерируется так, что в любой гиперплоскости, проходящей вдоль осей гиперкуба, содержалось ровно одно значение. В двумерном случае метод генерирует случайные значения в виде Латинского квадрата, на пересечении любых строк и столбцов которого содержится единственное значение (рис.3).

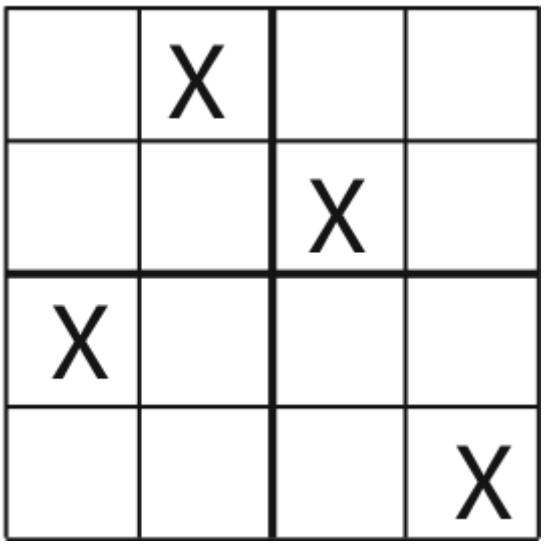

Рис.3. Латинский квадрат.

Это достигается благодаря разделению многомерной функции распределения на равные участки, после чего выбирается с каждого участка выбирается только одна случайная точка. Для того, чтобы избежать генерации гиперкуба с точками, расположенными вдоль

его диагонали, среди случайных наборов выбирают наборы с наиболее равномерном распределением в пространстве куба (рис.4).

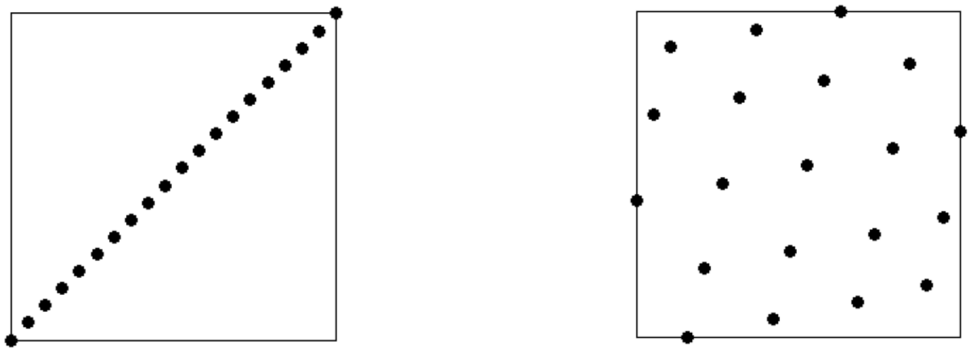

Рис.4. Латинские гиперкубы размерностью d = 2, заполненные 20-ю точками. При этом гиперкуб слева с точками, расположенными по диагонали, имеет плохие свойства заполняемости пространства, в то время как заполняемость точками в гиперкубе справа более равномерная.

Благодаря такому подходу удается генерировать репрезентативную с точки зрения плотности расположения точек случайную выборку с достаточно равномерной заполняемостью многомерного пространства [4]. Такие достоинства Латинского гиперкуба позволяют применять его для построения хорошей суррогатной модели, поскольку для нее критическое значение имеет первоначальная конфигурация случайных точек.

2.3 Библиотека PySOT

PySOT это библиотека для глобальной суррогатной оптимизации на Python. Она предназначена для целевых функций черного ящика, являющихся вычислительно дорогими и, в частности, имеющих целочисленную область задания параметров, которая, кроме того, должна быть ограниченной по всем измерениям [5] .

Реализация суррогатной оптимизации в PySOT состоит из 4 этапов:

1) Стратегия: Алгоритм выбора новых оценок после задания дизайна экспериментов. Это основная часть алгоритма оптимизации, поскольку она ответственна за новые оценивания функции и прекращение оптимизационного процесса при достижении критерия остановки;
2) Дизайн эксперимента: инициализация первоначального входного набора точек для построения суррогатной модели;
3) Суррогатная модель: выбор модели, аппроксимирующей целевую функцию (RBF, GP и тд.);

4) Задача оптимизации: вся доступная информация о задаче, например размерность, типы переменных, целевая функция и т. д. В качестве критерия остановки обычно выбирается максимальное число оценок функции.

В данной работе была применена стратегия DYCORS, которая является расширением метода стохастического радиального базиса (SRBF). Основная идея метода заключается в том, что многие целевые функции зависят только от нескольких направлений, поэтому выгодно двигаться только вдоль наиболее перспективных направлений. Для задания дизайна эксперимента используется симметричный Латинский гиперкуб, который определяется путем задания размерностью и количеством стартовых точек. При этом рекомендуется выбирать число начальных точек по формуле 2*размерность + 1, так как в таком случае дизайн будет обладать полным рангом. В качестве суррогатной модели была взята радиальная базисная функция RBFInterpolant. Процесс оптимизации запускается с помощью контроллера встроенным методом run. Контроллер ответственен за принятие или отклонение предложений, поступающих от объекта класса стратегии. Если контроллер принимает предложение, например, по оценки функции, то он обновляет записи по вызовам функции и соответствующим точкам, которые сохраняются в логах оптимизации.

Solana

Блокчейн – это децентрализованная распределенная система передачи и хранения данных, где добавление каждого нового блока в цепи основано на алгоритмах консенсуса (согласия). Одним из распространенных типов консенсуса является Proof-of-Work (PoW), используемых такими криптовалютами, как Биткойн и Эфириум. Для добавления нового блока майнерам необходимо решить сложную криптографическую задачу, называемую хеш-пазлом, используя свои вычислительные мощности [6] . Майнер, успешно решивший задачу, получает право расширить блокчейн блоком, состоящий из действительных транзакций. При этом он получает вознаграждение в виде монеты, записанной в добавленный блок.

Архитектура сети блокчейна Solana, в свою очередь, имеет в основе консенсус Proof-of-Stake. Как количество затраченных вычислительных ресурсов увеличивает вероятность генерации нового блока в PoW, так и число поставленных на кон (stake) монет увеличивает вероятность валидатора быть выбранным для продолжения сети в PoS. Сравнение этих двух типов консенсуса представлены на рис.5:

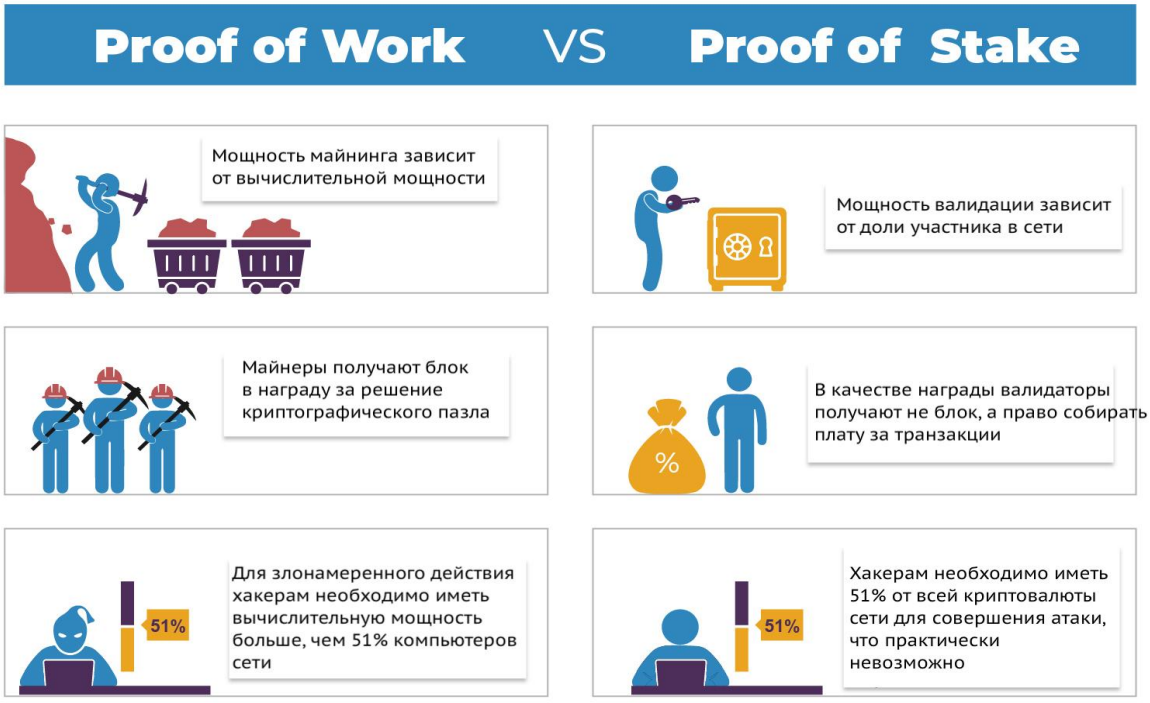

Рис.5. Сравнение отличительных черт консенсусов PoW and PoS

Для увеличения пропускной способности за счет записи временных отметок всех событий сети Solana дополняется протоколом Proof-of-History (PoH). PoH выступает в виде глобальных криптографических часов, обеспечивая согласованность чтения сети и поддающееся проверке течение времени. Протокол использует криптографически защищенную функцию, написанную таким образом, что вывод нельзя предсказать на основе ввода, так как каждое следующее значение функции принимает на вход значение предыдущего [7]. Путем последовательных вычислений мы делаем записи состояния, данных и индекса, соответствующему i-му вызову хеш-функции. Тем самым можно гарантировать, что данные были созданы в момент времени, предшествующему определенному хэшу в последовательности. Это позволяет использовать последовательность хешей как отметки времени и, следовательно, опираться на эту последовательность для предотвращения злонамеренных действий, направленных на изменения последовательности цепочки событий (рис.6).

PoH Sequence A

Index	Data	Output Hash
10		`hash10a`
20	Event1 = append(event1 data, `hash10a`)	`hash20a`
30	Event2 = append(event2 data, `hash20a`)	`hash30a`
40	Event3 = append(event3 data, `hash30a`)	`hash40a`

Рис.6. Пример PoH последовательности генератора A. Каждое новое событие сети содержит хеш, предшествующий во времени этому событию. Если событие Event3 к времени своего появления будет содержать хеш, отличный от hash30a, то это свидетельствует об перестановочной атаке и, следовательно, такая последовательность недействительна.

Кроме того, это обеспечивает согласованность действий всех валидаторов сети, увеличивая ее пропускную способность, не жертвуя при этом ее защищенностью. Таким образом, в блокчейне Solana валидаторы путем большинства голосов выбирают участника, который сгенерирует следующую последовательность PoH. Чтобы мотивировать участников голосовать только за валидные хэши, валидаторы, в случае голосования за недопустимый с точки зрения протокола PoH хэш, наказываются путем изъятия монет. Это делает экономически невыгодным злонамеренные действия и стимулирует валидаторов поддерживать только определенные форки сети.

Экспериментальная часть

Перейдем непосредственно к процессу подбора параметров блокчейна с целью получения наилучших выходных характеристик, которыми являются TPS и droprate. Команда разработчиков стартапа Z-union предоставили доступ к запуску тестовой сети блокчейна Solana, который разворачивается на виртуальных машинах инфраструктуры Amazon. Декларативное управление параметрами создаваемых AWS виртуальных машин осуществляется через Terraform. Запуск блокчейна на виртуальной машине происходит посредством скрипта на Python, вызывающий Terraform. В свою очередь в его коде находится docker image, внутри которого содержится команда создания генезисного блока, реестр которых расположен на AWS. Варьирование параметров разворачиваемой тестовой сети осуществляется путем внесения изменений в toml - файл с параметрами запуска. После того, как тестовая сеть блокчейна развернута, можно осуществлять проведение транзакций командой docker run, в процессе которого в командную строку выводятся логи, среди которых можно выделить TPS и droprate.

Итогом экспериментальной части данной работы является скрипт на Python, который последовательно реализует две основные задачи:

1) вызов функции черного ящика, представляющая собой:
   - запуск блокчейна с начальными параметрами;
   - проведение транзакций;
   - парсинг значений TPS и droprate;
   - сворачивание блокчейна.
2) оптимизация аргументов вызываемой функции.

Работа скрипта осуществляется после загрузки его на виртуальную машину, на которой предварительно установлен Terraform, папка со скриптами для запуска блокчейна и виртуальное окружение для управление зависимости Python-библиотек. После запуска скрипта функция оптимизатора вызывает функцию черного ящика, которая разворачивает блокчейн с первоначальными параметрами. В дальнейшем i-ый шаг итерации состоит из следующих операций:

- целевая функция принимает от оптимизатора целочисленные параметры блокчейна и, внося изменения в toml-файл, запускает блокчейн с шестью новыми параметрами и неизменными остальными;

- далее проводятся транзакции, и по их окончанию получаем текстовый файл, из которого извлекаются выходные параметры: TPS и droprate.

Целью оптимизации является максимизация взвешенной суммы данных параметров, в которую TPS входит с положительным весом 0.8, а droprate – с отрицательным весом -0.2. Тогда тестовая сеть блокчейна будет обладать наибольшей пропускной способностью и наименьшей долей отклоненных транзакций. Целевая функция черного ящика является вычислительно дорогой, а время вызова функции составляет порядка нескольких минут. Поэтому целесообразно ограничить бюджет вычислений максимальным числом вызовов функции, то есть ввести параметр max_fevals = 100, а также число итераций max_iters = 100. Все методы осуществляли поиск оптимальных параметров из ограниченной по всем измерениям области, которая составляла ±10% относительно начальных значений параметров. Границы области округлялись до ближайшего целого числа, поскольку все изменяемые параметры блокчейна являются целочисленными.

Были выбраны 3 метода оптимизации:

1) Наивная оптимизация. Наивный метод служит бейзлайном для сравнения работы 2 других оптимизаторов. Он заключается в генерации 100 (по величине бюджета) случайных выборок параметров из области ограничения и выборе такого набора, подстановка которого дает наименьшее значение целевой функции.
2) Оптимизация с ограничениями. Использует Multi-verse optimizer (MVO) библиотеки mealpy.
3) Суррогатная оптимизация. Использует библиотеку PySOT, в которой строится суррогатный оптимизатор согласно гл.1 п.3.

В соответствии с ограничением на число вызовов функции для каждого метода были проведены эксперименты на 100 точках. Затем, среди 100 измерений выбирался тот набор параметров, при котором целевая функция достигала максимума. Результаты экспериментов приведены в таблице:

тип оптими-зации	шаг итерации	NUM_ THREADS	DEFAULT_ TICKS_ PER_SLOT	ITER_ BATCH _SIZE	RECV _BATCH_ MAX_ CPU	DEFAULT_HASHES_PER_SECOND	DEFAULT_ TICKS _PER_ SECOND	AVERAGE_ TPS	AVERAGE_DROPRATE
Начальные параметры		4	1033	900	65	1866666	152	23.60	0.0
Наивная	22	3	980	859	68	2007222	136	27.13	0.0
Ограни-ченная	57	4	1082	929	67	1754909	136	39.54	0.0
Сурро-гатная	13	3	980	859	68	2007222	136	39.83	0.0

Таблица 1. Результаты 3 видов оптимизации. Представлены оптимальные входные параметры, максимизирующие выходной параметр блокчейна – среднее значение TPS (AVERAGE_TPS), а также шаг итерации, на котором данная схема оптимизации достигает максимума целевой функции блокчейна. Для сравнения в первой строке приведены начальные параметры блокчейна, являющиеся центром многомерной области ограничений.

Из приведенной таблицы можно сделать следующие выводы:

1) Наивная оптимизация на 22 шаге привела к росту среднего TPS на 13%;
2) Ограниченная оптимизация на 57 шаге увеличила средний TPS на 31% по сравнению с наивной оптимизации, взятой в качестве бейзлайна;
3) Суррогатная оптимизация позволила получить на 32% больший средний TPS, чем наивная оптимизация. При этом она достигла максимального значения на 77% быстрее, чем ограниченная оптимизация.

Таким образом, суррогатная оптимизация не только показала рост значения выходного параметра блокчейна (среднего TPS) по сравнению с бейзлайном, но и показала значительный рост скорости нахождения максимума по сравнению с ограниченной оптимизацией. Это обусловлено тем, что суррогатная оптимизация лучше подходит для оптимизации вычислительно дорогих функций черного ящика, поскольку оптимизирует не саму целевую функцию, а ее аналитическую модель.

Заключение

Сети блокчейна, такие как Solana, обладают рядом настраиваемых параметров, которые задаются перед запуском сети. В дальнейшем они определяют режим работы сети, ее производительность и защищенность. Рассматривая блокчейн как функцию черного ящика, получающий на вход 6-мерный вектор и выдающий 2-х мерный, мы можем рассматривать поиск оптимальной конфигурации сети как задачу регрессии. Целью является получение наилучших выходных характеристик блокчейна, к которым относятся TPS и droprate. В условиях ограниченного бюджета вычислений и значительной вычислительной сложности целевой функции, являющейся блокчейн-сетью, наиболее привлекательным вариантом представляется использование схемы суррогатной оптимизации. Ее преимущество в том, что она не пытается оптимизировать исходную функцию напрямую, а сводит задачу к оптимизации суррогатной модели, которая является аналитическим представлением целевой функции. Для сравнения были взяты две оптимизационные схемы: суррогатная и оптимизация с ограничениями, результаты которых сравнивались с бейзлайном – наивным оптимизатором, который на каждом шаге итерации подставляет случайную выборку параметров из ограниченной области. Результаты экспериментов с ограниченным бюджетом показали, что суррогатная оптимизация приходит к оптимальным выходным значениям всего на 13 итерации, что на 77% быстрее конкурирующей схемы оптимизации, получая максимальное значение TPS на 32% большее по сравнению с бейзлайном.

Список литературы

1. Amelin V. et al. Machine Learning View on Blockchain Parameter Adjustment // 2021 3rd Blockchain and Internet of Things Conference. New York, NY, USA: Association for Computing Machinery, 2021. P. 38–43.
2. Guo S. An Introduction To Surrogate Optimization: Intuition, illustration, case study, and the code [Electronic resource] // Towards Data Science. 2020. URL: https://towardsdatascience.com/an-introduction-to-surrogate-optimization-intuition-illustration-case-study-and-the-code-5d9364aed51b (accessed: 23.06.2022).
3. Seeger M. Gaussian processes for machine learning // Int. J. Neural Syst. 2004. Vol. 14, № 2. P. 69–106.
4. Viana. Things you wanted to know about the Latin hypercube design and were afraid to ask // 10th World Congress on Structural and Multidisciplinary Optimization.
5. David E. et al. pySOT Documentation [Electronic resource] // 2020. URL: https://pysot.readthedocs.io/_/downloads/en/stable/pdf/ (accessed: 23.06.2022).
6. Wang T., Liew S.C., Zhang S. When Blockchain Meets AI: Optimal Mining Strategy Achieved By Machine Learning // arXiv [cs.CR]. 2019.
7. Yakovenko A. Solana: a new architecture for a high performance blockchain v0. 8.13. Solana cryptocurrency whitepaper (2020).