**Министерство образования и науки Российской Федерации**

**Московский физико-технический институт (государственный университет)**

Физтех-школа прикладной математики и информатики

Кафедра проблем передачи информации и анализа данных

Выпускная квалификационная работа бакалавра

# Суррогатная оптимизация параметров систем распределенного реестра

**Автор:**

студент Б05-904в группы

Щербаков Андрей Сергеевич

**Научный руководитель:**

канд. физ.-мат. наук

Зайцев Алексей Алексеевич


Москва 2023

**Аннотация**

# Суррогатная оптимизация параметров систем распределенного реестра

*Щербаков Андрей Сергеевич*

В последние годы распределенные реестры, такие как блокчейн, стали популярными инструментами для обеспечения безопасности и прозрачности в различных областях, включая финансы, индустрия, здравоохранение и тд. Однако, в связи с ростом их популярности возникла потребность в увеличении эффективности их работы. Оптимизация параметров распределенного реестра является сложной задачей, так как она требует обширных вычислительных ресурсов и времени. В этом контексте суррогатная оптимизация представляет собой мощный инструмент для эффективного поиска оптимальных значений параметров. Данная работа рассматривает различные механизмы суррогатной оптимизации, сравнивает их, а также подготавливает почву для экспериментального исследования в облаке. Отдельное внимание в работе посвящено выбору целевой функции для проведения оптимизации.

**Оглавление**

# Глава 1: Введение

## 1.1 Блокчейн как Чёрный ящик

Блокчейны являются децентрализованными сетями, где несколько участников должны согласовывать изменения в хранилище данных [1]. Для достижения согласия между участниками используются протоколы консенсуса, такие как proof-of-work [2]. Они необходимы для корректной работы, но весьма энергозатратны [3].

Вопрос эффективности работы блокчейна также довольно объемен. Существует несколько основных моментов, важных для корректной и бесперебойной работы блокчейна [4]:

- быстрое подтверждение транзакций;
- уменьшение задержки (latency);
- качество цепочки блоков;

Высокая производительность блокчейна выражается в терминах количества транзакций в секунду (tps). При этом задача настройки параметров блокчейна довольно сложна [5] и вычислительно затратна.Таким образом, нашей целью является максимизация пропускной способности (tps) при ограниченном вычислительном бюджете и суррогатная оптимизация наиболее подходит для выполнения данной задачи [6].

Для её реализации предлагался подход черного ящика для оценки производительности блокчейна в купе с оптимизацией наиболее значимых параметров [5]. Подход Черного ящика используется блокчейн в качестве замкнутой системы с заданными параметрами. В этом подходе параметры блокчейна изменяются, а производительность оценивается на основе результатов работы виртуальной сети блокчейна. Он позволяет исследо-

вателям изучать и настраивать параметры блокчейна для максимизации пропускной способности и других показателей эффективности.

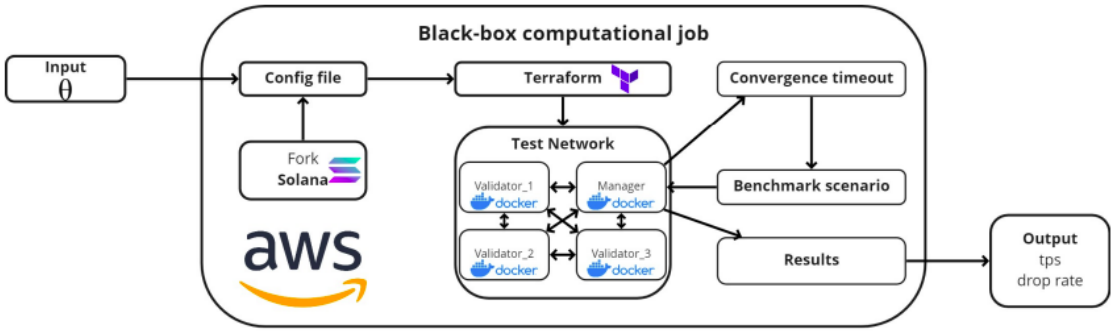



Рис. 1.1: Устройство ЧЯ[5]

Необходимо заметить, что у меня не было физического доступа к Черному ящику во время написания работы, но зато были данные, полученные из первоначального эксперимента [5]. В связи с этим эта работа планировалась как отладка алгоритма суррогатной оптимизации с использованием "искуственного"Черного ящика, для дальнейшего эксперимента с реальным Чёрным ящиком.

## 1.2 Параметры распределенного реестра

Это исследование основано на статье 2022 года [5], в которой в качестве Черного ящика рассматривался конфигурируемый блокчейн в облаке на основе Solana. Мы рассматриваем весь рабочий процесс как функцию $y = F(x, \theta, \xi)$ [5], где:

x - вектор наблюдаемых неуправляемых параметров блокчейна. $\theta$ - вектор (наблюдаемых) управляемых параметров блокчейна. $\xi$ - скрытые (неуправляемые) параметры блокчейна.

y - режима работы блокчейна с заданными параметрами (x, $\theta$, $\xi$). В нашем случае, tps (транзакций в секунду).

Поскольку x и $\xi$ -неуправляемые параметры (случайные величины), то Таким образом в данной статье мы рассматриваем $f(\theta)$ как матожида-

ние $F(x, \theta, \xi)$, а наша задача превращается в

$$\max_{\theta} f(\theta)$$

при N максимально возможном числе вызовов f.

В в ходе эксперимента было выделено 6 наиболее значимых параметров:

| Parameter | Default value | Description |
|---|---|---|
| NUM_THREADS | 4 | To improve the throughput to the database, Solana supports connection pooling using multiple threads, each maintaining a connection to the PostgreSQL database. |
| TICKS_PER_SLOT | 63 | The period for which each leader ingests transactions and produces a block is called a slot. Each slot is divided into logical units called ticks. |
| RECV_BATCH_MAX_CPU | 1000 | Transaction signature verification is a repetitive and time-consuming operation. Solana sends a batch of transactions for the verification to schedule the load. By default, a CPU verifies transactions, and RECV_BATCH_MAX_CPU defines the maximum batch size. |
| ITER_BATCH_SIZE | 1000 | Batch inserts into a database reduce the round trips to the database and, hence, improve performance under high load. |
| HASHES_PER_SECOND | 2000000 | Solana documentation states Google Cloud Platform's n1-standard hardware to be the standard hardware and claims it to have HASHES_PER_SECOND hash computations per second. |
| TICKS_PER_SECOND | 160 | A ledger entry that estimates wallclock duration is called a tick. |

Рис. 1.2: Наиболее значимые параметры и их описание[5]

Именно эти параметры мы и попытаемся оптимизировать. Для отладки алгоритма (low-fi модели) мы создадим "искуственный"Чёрный ящик , апроксимирующий нашу систему с добавлением нормального белого шума, не требующий дорогостоющих экспериментов.

## 1.3 Задача суррогатной оптимизации

Суррогатная оптимизация - это метод оптимизации, который использует аппроксимационную (суррогатную) модель, чтобы оценивать и выбирать лучшие значения для неизвестной целевой функции (в нашем случае это функция $f(\theta)$), которая может быть дорогостоящей или сложной для вычисления. Вместо прямого вычисления целевой функции, суррогатная модель строится на основе известных значений функции в некоторых точках. Затем модель используется для предсказания значений функции в других точках, позволяя выбирать наиболее перспективные точки для дальнейшего исследования.

Суррогатная оптимизация особенно полезна в случаях, когда целевая функция не может быть аналитически описана или является вычислительно сложной: в нашем случае как раз имеются ограничения по количеству проведенных экспериментов-запусков черного ящика. Этот подход позволяет сократить количество вычислений целевой функции, ускоряя процесс оптимизации и экономя ресурсы. Однако, точность и надежность суррогатной модели являются важными факторами, и требуются тщательные методы выбора и обновления модели для достижения оптимальных результатов [7].

Для лучшего понимания сравним простые схемы классической и суррогатной оптимизации (Рис. 1.3). Baseline отвечает за априорные вероятности в нашей Байесовской (суррогатной)оптимизации, суррогатная модель дает возможность дёшево оценивать целевую функцию и находить кандидаты на точку оптимума (оптимизировать так называемую acquisition function - выбор этой функции равносилен выбору алгоритма оптимизации) , в этой точке мы получаем значение целевой функции до тех пор пока не достигнем желаемого результата или воспользуемся всеми запусками, предусмотренными в бюджете. При этом с каждой новой точкой мы можем корректировать нашу суррогатную модель (переобучать её), поскольку мы получили дополнительные данные о нашей системе. В нашем случае baseline - это данные, полученные на эксперименте [5], а остальные элементы (суррогатная модель, алгоритм оптимизации и тд) необходимо выбирать исходя из них.

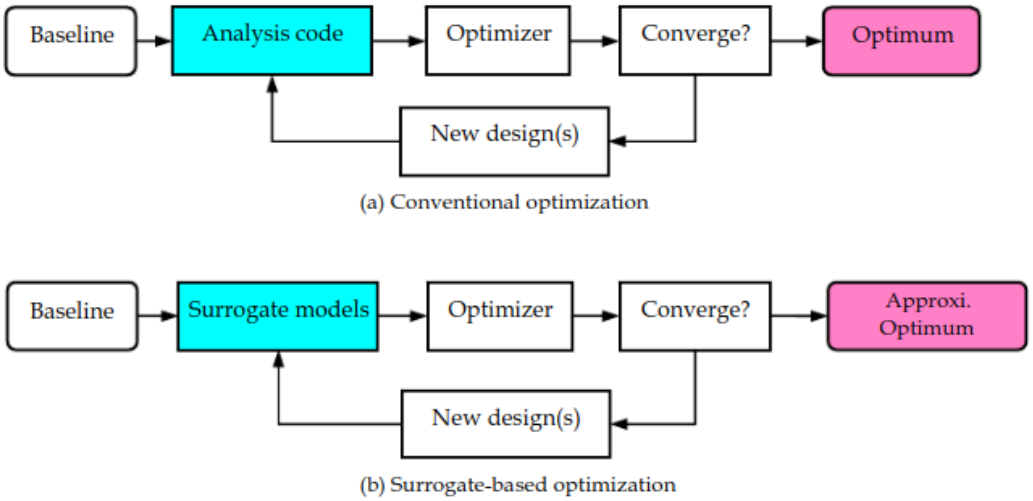



Рис. 1.3: Сравнение схемы для классической оптимизации (a) и оптимизации на основе суррогатных моделей(b) [8].

## Глава 2: Цель и задачи

Целью работы является суррогатная оптимизация параметров распределенного реестра.

Для достижения поставленной цели были сформулированы следующие задачи:

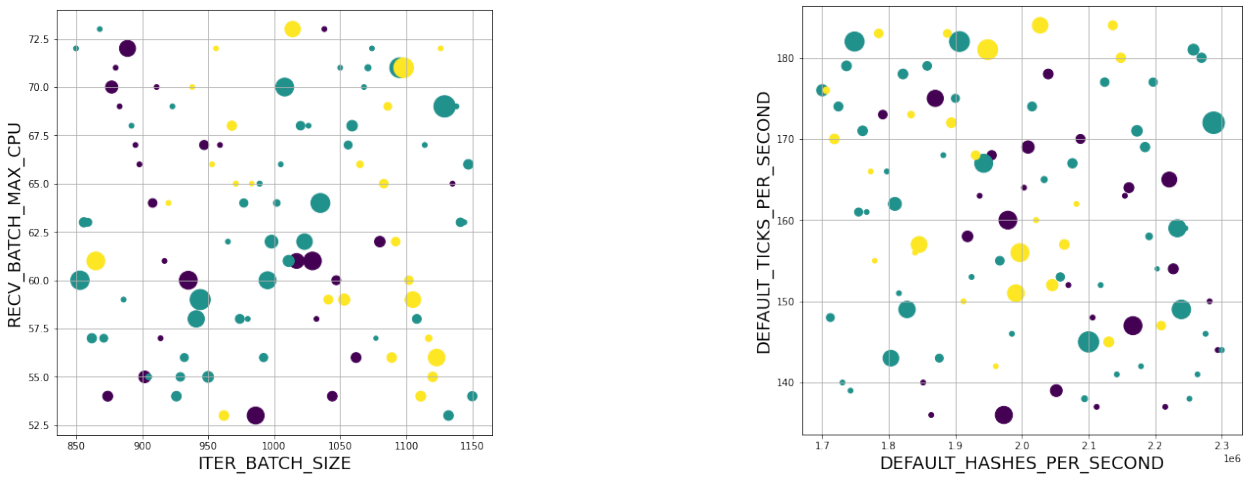
1. проанализировать данные, полученные в ходе эксперимента в облаке;
2. выбрать наулучшую модель аппроксимации, чтобы оценивать целевую функцию;
3. реализовать "искусственный"Чёрный ящик для получения low-fi модели и целевой функции;
4. выбрать суррогатную модель;
5. сравнить различные алгоритмы суррогатной оптимизации на различных бюджетах и их устойчивость к шумам;
6. сформулировать вид эксперимента в облаке на реальном Чёрном ящике (далее ЧЯ);

Первая задача была уже реализована в статье [5] и мы воспользуемся результатами из неё для нашей следующей задачи.

# Глава 3: Анализ данных эксперимента и создание "искуственного"ЧЯ

## 3.1 Анализ предоставленных данных

Как было показано ранее 1, мы имеем дело с 6-мерным набором параметров среди которых есть, как дискретные, так и непрерывные переменные. В ходе ранего эксперимента lанные были получены с использованием метода Симметричного Латинского Гиперкуба (SLHC) [5], изобразим их двумерные срезы по парам параметров 3.1a и 4.1b. На данных срезах цвет точки обозначает параметр количество threads (3 - фиолетовый, 4 - голубой и 5 жёлтый), а размер точки прямопропорционален целевой функции (количеству транзакций в секунду).




(a) (b)

Рис. 3.1: Срезы данных по соответствующим параметрам

## 3.2 Критерии подбора модели

Мы хотим выбрать модель, хорошо апроксимирующую наш Чёрный ящик, чтобы на её основе создать "Искуственый"Чёрный ящик. При сложной функции и невыпуклых данных следующие критерии могут быть рас-

смотрены при выборе модели:

- Гибкость модели: Сложная функция может требовать модели с высокой гибкостью, способной адаптироваться к сложным нелинейным зависимостям в данных. В этом случае, модели с высокой гибкостью, такие как, например, случайный лес или градиентный бустинг или методы ядерной регрессии (например, метод опорных векторов с ядром) могут быть более подходящими.

- Нелинейные зависимости: Если данные содержат нелинейные зависимости или взаимодействия между переменными, то модели, способные улавливать их, могут быть предпочтительными. Например, полиномиальная регрессия, сплайн-регрессия

- Регуляризация: Если данные невыпуклые или имеют большое количество признаков, модели с регуляризацией могут быть предпочтительными. Регуляризация помогает справиться с проблемой переобучения и улучшить обобщающую способность модели. Например, L1 или L2 регуляризация может использоваться в линейной регрессии, а методы регуляризации градиентного бустинга могут быть применены к ансамблям деревьев.[9]

- Кросс-валидация и сравнение моделей: В случае сложной функции и невыпуклых данных, важно провести кросс-валидацию и сравнить различные модели для выбора оптимального варианта. Сравнение моделей на основе метрик оценки, таких как средняя квадратичная ошибка или R2, поможет определить, какая модель лучше соответствует данным.[6]

В итоге, выбор модели при сложной функции и невыпуклых данных зависит от специфики проблемы, доступных данных и требований к точности

и интерпретируемости модели. Экспериментирование с различными моделями и их адаптация к конкретному контексту может быть полезным для достижения наилучших результатов.

### 3.3 Подбор модели

Данные, полученные в ходе эксперимента [5],довольно сложно интерпертируются и требуют модели с высокой гибкостью и возможностью регуляризации. Нам также важно провести кросс-валидацию и сравнить различные модели для выбора оптимального варианта. Проведём сравнение моделей по метрике R2. Метрика R2 (коэффициент детерминации) определяется следующей формулой:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

где:

- n : количество примеров в выборке;
- $y_i$ : истинное значение целевой переменной для i-го примера;
- $\hat{y}_i$ : предсказанное значение целевой переменной для i-го примера;
- $\bar{y}$ : среднее значение целевой переменной в выборке.

В нашем случае мы уже знаем, что модель на основе Гаусовских процессов лучше подходит под нашу задачу, чем Метаевристическая модель ([5]), поэтому для того чтобы выбрать модель из предложенных вариантов я воспользовался кросс валидацией с количество сплитов = 10 и с метрикой R2. В качестве основных моделей возмем (исходя из вышеизложенных критериев) метод опорных векторов(svr), метод градиентного бустинга и гаусовские процессы (регрессия). Кроссвалидация показала, что наилучшие результаты дает градиентный бустинг. С небольшим отрывам от него идут гауссовские процессы с РБФ ядром (3.1).

Таким образом мы получаем, что наиулчшим выбором для "искус-

Таблица 3.1: Результаты кросс-валидации

| Model | Best Scoring (R2) | Parameters |
|---|---|---|
| svr | 0.0546 | {'kernel': 'linear', 'degree': 5.0} |
| gradient boosting | 0.320 | {'n estimators': 100, 'max depth': 3, 'learning rate': 0.05} |
| gaussian process | 0.152 | {'kernel': 'rbf', 'length scale': 2.64} |

ственного"ЧЯ является градиентный бустинг с соответствующими параметрами. Напомним, что градиентный бустинг[10] можно задать таким образом, это нам поможет в понимании следуещего раздела.

$$\text{Градиентный бустинг} = \arg\min_{F} \sum_{i=1}^{n} L(y_i, F(x_i)) + \sum_{k=1}^{K} \Omega(f_k) \tag{3.1}$$

где: - $F$ - ансамбль моделей,

- $L(y_i, F(x_i))$ - функция потерь,

- $\Omega(f_k)$ - регуляризация моделей,

- $y_i$ - истинное значение,

- $x_i$ - входные данные,

- $K$ - количество моделей в ансамбле.

### 3.4 Анализ сходимости модели полученого "искуственного"ЧЯ

Проверим насколько хорошо сходится наша модель. Для этого построим график квадратичной функции потерь [6]: Квадратичная функция потерь (MSE) определяется следующей формулой:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

где: - n : количество примеров в выборке;

- $y_i$ : истинное значение целевой переменной для i-го примера;

- $\hat{y}_i$ : предсказанное значение целевой переменной для i-го примера.

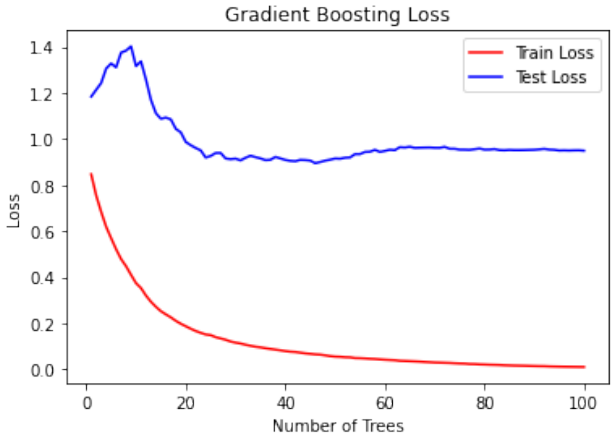



Рис. 3.2: Квадратичная Функция потерь для "искуственного"ЧЯ

По графику видим 3.2, что в целом наша модель сходится , что достаточно для того, чтобы применить её для отладки нашей суррогатной оптимизации и подготовки соответствующего эксперимента. Чтобы протестировать устойчивость к шумам мы добавим 2% белый шум (нормально распределенная случайная величина с 0-ым средним и вариацией в 2 процента от максимального значения целевой функции) в целевую функцию, чтобы приблизить условия к более реальным.

# Глава 4: Алгоритмы суррогатной оптимизации

## 4.1 Обзор существующих реализованных решений по суррогатной оптимизации

Одной из наиболее распространненых реализаций суррогатной оптимизации является Байесовская оптимизация: Это метод оптимизации, основанный на гауссовских процессах. Он использует байесовский подход для моделирования неизвестной целевой функции и выбора следующей точки для оценки [11]. Популярные реализации включают scikit-optimize, GPyOpt, PySOT, Botorch и Spearmint.

Каждое из этих решений имеет свои особенности, преимущества и недостатки. Выбор конкретного решения зависит от требуемых возможностей, характеристик задачи и доступных ресурсов.

Применение байесовской оптимизации включает следующие шаги [12]:

- Определение априорной модели целевой функции. В нашем случае обучение суррогатной модели на предоставленных данных (метод SLHC см. 3), оцененых с помощью целевой функции, т.е. "искуственного"ЧЯ.
- Выбор следующей точки для оценки с использованием алгоритма выбора (acquisition function)
- Вычисление целевой функции в выбранной точке.
- Обновление апостериорной модели с учетом полученных данных.
- Повторение шагов 2-4 до достижения условия остановки (пока не превышен бюджет).

Важно заметить, что существуют также другие методы оптимизации опирающиеся на суррогатную модель, такие как Эволюционные алгоритмы

(Surrogate Model-Assisted Evolutionary Algorithms), алгоритмы многокритериальной оптимизации (Surrogate-Assisted Multi-Objective Optimization), а также генетические(Surrogate-Assisted Genetic Programming ) и алгоритмы роя частиц (Surrogate-Assisted Particle Swarm Optimization ), но по объективным причинам, они не рассматриваются в данной работе, поскольку имеют немного другие специфические задачи, отличающиеся от нашей.

## 4.2 Критерии выбора суррогатной модели

При выборе суррогатной модели для оптимизации рекомендуется учитывать следующие критерии [7]:

- Предсказательная точность: Суррогатная модель должна обеспечивать довольно высокую точность предсказания целевой функции или метрики качества.
- Вычислительная эффективность: Модель должна быть вычислительно эффективной, чтобы обеспечить быструю оценку целевой функции и сократить время оптимизации.
- Гибкость: Модель должна быть гибкой и способной адаптироваться к различным типам данных и структурам проблемы оптимизации.
- Интерпретируемость: В некоторых случаях важно иметь возможность интерпретировать суррогатную модель и понять, какие факторы влияют на результаты оптимизации.
- Устойчивость к шуму: Модель должна быть устойчивой к шуму в данных и способной обработать выбросы или неоднородности в данных.
- Ресурсоемкость: Модель должна требовать разумного объема ресурсов (памяти и вычислительной мощности) для обучения и примене-

ния.

Под наши критерии подходят Гауссовские процессы: как было показано в 3 они хоть и уступают в предсказательной точности Градиентному бустингу, но зато не имеют проблем с гибкостью, ресурсоёмкостью. Также методы, основанные на гаусовских процессах имеют под собой хорошую теоретическую базу [11]. В связи с этим была выбрана библиотека готовых решений botorch, как наиболее оптимальная в нашей ситуации. В ней был использован метод SingleTaskGP (botorch SingleTaskGP), который лучше всего работает при единичной матрице ковариации, а результаты стандартизированы (с нулевым средним и единичной дисперсией). В связи с этим мы будем использовать Стандартизацию входных данных, для лучших результатов.

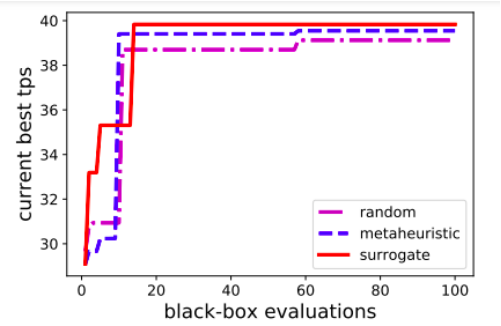
### 4.3 Baseline оптимизации

Из [5] мы имеем baseline оптимизации ЧЯ. В Качестве baseline были выбраны рандомная оптимизация, оптимизация ,сонованная на Метаевристических методах, а также байесовская оптимизация с RBF ядром (для суррогатной модели) и DYCORS стратегий выбора следующих точек. Ниже приведены результаты данных методов.

Именно исходя из превосходства суррогатной оптимизации, над данными методами мы и рассматриваем её, как один из наиболее перспективных методов оптимизации параметров систем распределеного реестра.

### 4.4 Алгоритмы выбора

Существует несколько зарекомендовавших себя алгоритмов выбора для байесовской оптимизации (BO) для определения следующей точки для оценки целевой функции. Некоторые из основных алгоритмов выбора (реализовынные при этом в botorch) это:




(a) Сравнение сходимости различных методов при небольших бюджетах

| Method | $\theta$ | tps |
|---|---|---|
| Default | (4, 1000, 1000, 63, 2000000, 160) | 29.5 ± 0.4 |
| Random | (3, 1033, 867, 67, 1951151, 142) | 38.1 ± 1.3 |
| Metaheuristic | (4, 1082, 929, 67, 1754909, 136) | 38.6 ± 0.9 |
| Surrogate | (3, 980, 859, 68, 2007222, 136) | 38.9 ± 0.8 |

(b) Результаты оптимизации

Рис. 4.1: Baseline

- Upper Confidence Bound (UCB): Этот алгоритм основан на оценке верхней границы доверия и стремится найти баланс между исследованием и использованием уже известной информации.

$$\text{Upper Confidence Bound (UCB)} = \mu(x) + \sqrt{\frac{\beta}{2}\log(\frac{n}{n(x)})} \tag{4.1}$$

где: - $\mu(x)$ - среднее значение модели в точке $x$, - $\beta$ - коэффициент баланса исследования и использования, - $n$ - общее количество выборов, - $n(x)$ - количество выборов в точке $x$.

- Expected Improvement (EI): Этот алгоритм оптимизирует ожидаемое улучшение путем выбора точки, которая имеет наибольшее ожидаемое улучшение по сравнению с текущим наилучшим значением.

$$EI = \mathbb{E}[\max(f(x) - f(x^*), 0)] \tag{4.2}$$

где: - $f(x)$ - целевая функция, которую необходимо оптимизировать,

- $x^*$ - текущая наилучшая точка, - $\mathbb{E}$ - оператор математического ожидания.

- Probability of Improvement (PI): В этом алгоритме выбирается точка с наибольшей вероятностью превышения текущего наилучшего значения.

$$\text{Probability of Improvement (PI)} = P(f(x) \geq f(x^*)) \tag{4.3}$$

где: - $f(x)$ - целевая функция в точке $x$, - $x^*$ - текущая наилучшая точка, - $P$ - вероятность превышения текущего наилучшего значения.

- Lower Confidence Bound (LCB): Этот алгоритм выбирает точку с наименьшей ожидаемой стоимостью исследования.

$$\text{Lower Confidence Bound (LCB)} = \mu(x) - \sqrt{\frac{\beta}{2}\log(\frac{n}{n(x)})} \tag{4.4}$$

где: - $\mu(x)$ - среднее значение модели в точке $x$, - $\beta$ - коэффициент баланса исследования и использования, - $n$ - общее количество выборов, - $n(x)$ - количество выборов в точке $x$.

- Thompson Sampling: Этот алгоритм основан на сэмплировании из апостериорного распределения параметров модели и выборе действия с наибольшим сэмплированным значением награды. Алгоритм обеспечивает баланс между исследованием и использованием уже полученной информации.

$$\text{Thompson Sampling} = \arg\max_x P(f(x) \geq f(x^*)) \tag{4.5}$$

где: - $f(x)$ - целевая функция в точке $x$, - $x^*$ - текущая наилучшая точка, - $P$ - распределение вероятности.

В связи с тем, что стоимость исследования у нас одинаковая, то из данного списка исключим LCB. Для остальных алгоритмов мы проведём

эксперимент с оптимизацией "искуственного"ЧЯ, сравним между собой и с baseline для определения наиболее успешного алгоритма.

## Глава 5: Исследование различных алгоритмов выбора

В данной главе приводятся результаты экспериментов по суррогатной оптимизации парамеров распределенного реестра с использованием "искуственного"ЧЯ. Рассматриваемые алгоритмы перечислены ниже.

1. Алгоритм Upper Confidence Bound (далее UCB)
2. Алгоритм Expected Improvement (далее EI)
3. Алгоритм Probability of Improvement (далее PI)
4. Алгоритм Thompson Sampling(далее TS)

Все данные алгоритмы были имплементированы, используя схему Байесовской оптимизации (см. 4).

### 5.0.1 Проверка работы алгоритма с помощью функции Розенброка

Для начал необходимо убедиться в правильности работы алгоритмов суррогатной оптимизации. Для этогоВ качестве проверики мы рассматриваем их работу, заменив целевую функцию: вместо "искуственного"ЧЯ мы подставляем 6-мерную функцию Розенброка.

$$f(\mathbf{x}) = \sum_{i=1}^{5} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \tag{5.1}$$

где $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$ - вектор переменных.

Функция Розенброка является популярным тестовым случаем для суррогатной оптимизации тк содержит несколько локальных минимумов, что делает ее многомодальной. Это позволяет оценить способность алгоритмов суррогатной оптимизации находить и улучшать различные оптимальные решения в присутствии множества локальных экстремумов. Дополнительно она является Гладкой и достаточно простой в вычислении.

По результатам теста все алгоритмы нашли глобальный оптимум (задача минимизации, глобальный оптимум у 6-мерной функции Розенброка равен 0.0), в связи с этим можем их использовать на "искуственном"ЧЯ.

### 5.0.2 Полученные результаты

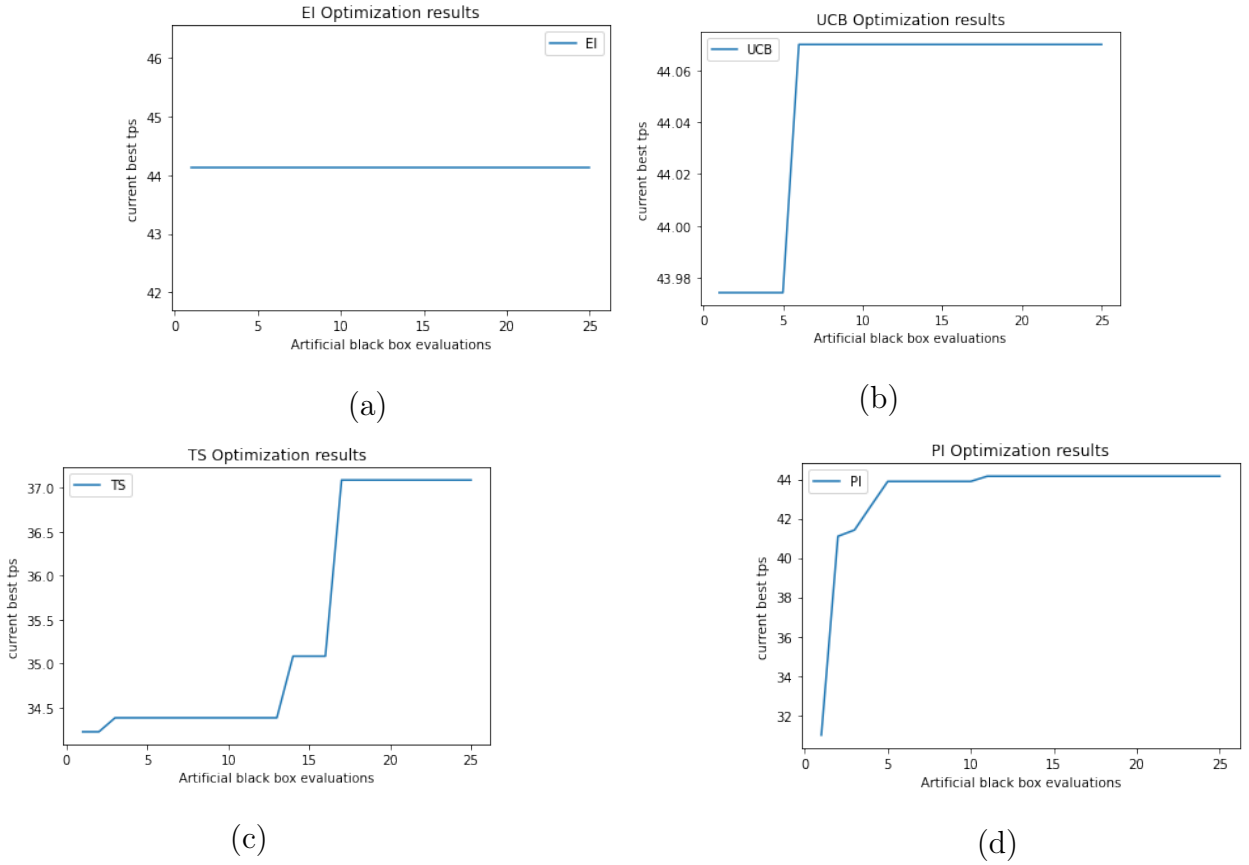



Рис. 5.1: Максимальный tps на каждом этапе итерации для методов (EIa), UCB(b), TS (c)и PI(d)

Результаты оптимизации приведены в таблице 5.1, а зависимость максимально полученного tps на каждой итерации приведена в графиках 6.1 для каждого из методов в отдельности.

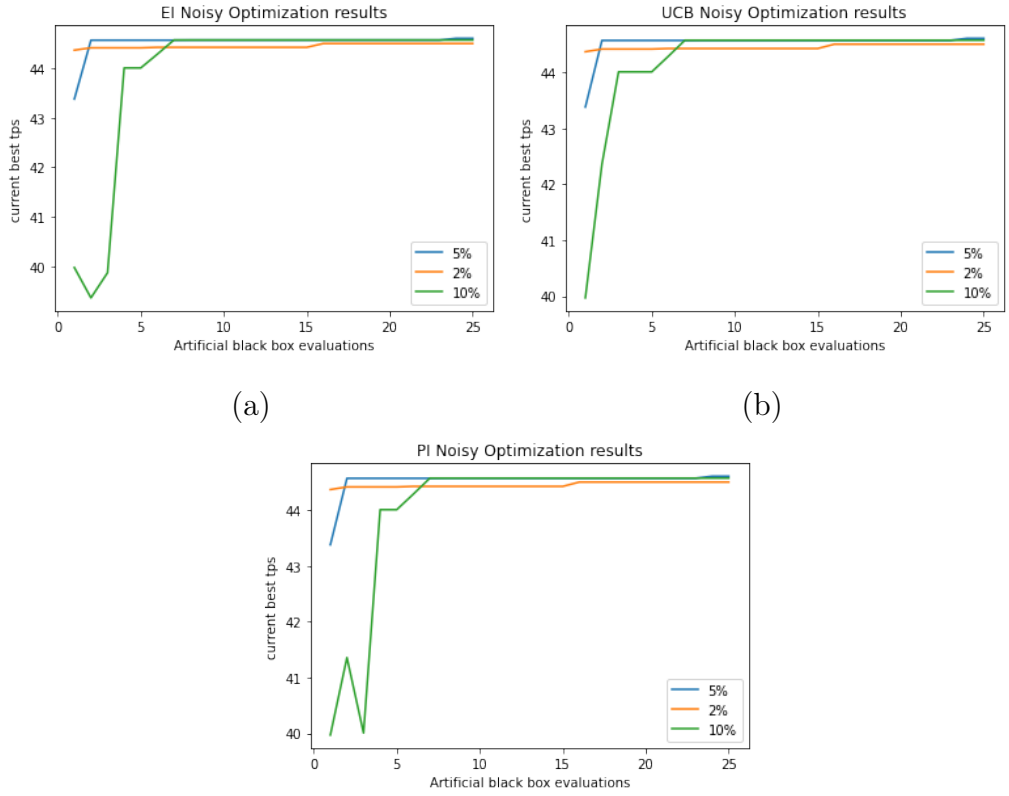
### 5.0.3 Устойчивость к небольшому уровню шума

Для анализа устойчивости к уровню внешнего шума для нашей оптимизации была проведена оптимизация для различных уровней шума.6.2

| Method | $\theta$ | tps |
|---|---|---|
| Baseline Surrogate | [3, 980, 859, 68, 2007222, 136] | 38.9 ± 0.8 |
| PI | [4, 976, 1128, 67, 2286946, 172] | 44.1 ± 0.9 |
| TS | [4, 1012, 1035, 68, 2003988, 174] | 37.0 ± 0.7 |
| EI | [4, 976, 1129, 69,2287593, 172] | 44.1 ± 0.9 |
| UCB | [4, 973, 1129, 69,2291512, 172] | 44.1 ± 1.0 |

Таблица 5.1: Значения алгоритмов

Шум в данном случае - нормальный с нулевым срдним и дисперсией = количеству процентов от максимального значения целевой функции.




(a) (b)

(c)

Рис. 5.2: Устойчивость к шуму для методов (EIa), UCB(b), и PI(c)

Как можно выдеть из полученных графиках, при мылых уровнях шума UCB, PI и EI всё равно сходятся примерно к одной и той же точке. Таким образом мы получаем устойчивость данных методов к небольшим

внешним шумам.

### 5.0.4 Анализ полученных результатов

Исходя из данных результатов и 6.1 мы можем видеть, что TS не находит тот же оптимум, что и все остальные из чего можно сделать вывод, что он всё же не так сильно устойчив к локальным точкам оптимума. В придачу, доверительные интервалы TS и Baseline моделей пересекаются, что может говорить о том, что они находят один и тот же локальный оптимум, упуская глобальный.

Оба алгоритма EI и UCB находят точку оптимума довольно быстро и точно, PI немного дольше, но также сходится к той же самой точке. Все три данных метода применимы в нашей задаче. При этом, стоить заметить небольшую разницу в скорости работы данных методов. Учитывая, насколько сильно стала больше оптимальноеь значение tps мы однозначно улучшили baseline модель.

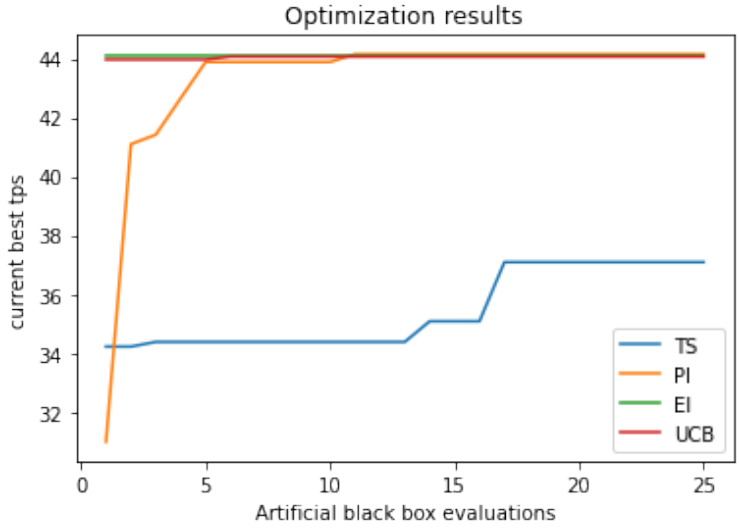



Рис. 5.3: Сведенные вместе результаты оптимизации

Как мы можем видеть из сравнительного графика 5.3, все алгоритмы, кроме TS, находят оптимальную точку (которая отличается от baseline модели), но при этом все алгоритмы кроме TS находят одну и ту же точку,

что подчеркивает надёжность и корректность работы метода суррогатной оптимизации. Заметим, что разница несмотря на добавленный шум в работу "искуственного"Ч.Я. методы UCB, EI и PI (сходятся к одним и тем же точкам в рамках шумовой погрешности) Иными словами, наша суррогатная оптимизация довольно устойчиа перед помехами при небольшом бюджете.

## Глава 6: Заключение

В рамках проведенной работы были выполнены следующие задачи:

1. Были проведены анализ данных, полученных в ходе эксперимента в облаке, включающий изучение статистических характеристик и визуализацию данных.

2. Была выбрана наилучшая модель аппроксимации для оценки целевой функции на основе проведенного анализа данных.

3. Был реализован "искусственный"Чёрный ящик, который эмулирует поведение реального Чёрного ящика и позволяет получить lo-fi модель и целевую функцию для отладки алгоритма оптимизации.

4. Была выбрана суррогатная модель, наиболее подходящая по нашим критериям.

5. Были сравнены различные алгоритмы суррогатной оптимизации, такие как UCB, EI, PI и TS на различных бюджетах, что позволило определить их производительность и эффективность.

6. Был проведен анализ устойчивости к зашумленным данным для UCB, EI и PI

7. Был сформулирован вид эксперимента в облаке на реальном Чёрном ящике, определены его параметры, цели и ожидаемые результаты.

Полученные результаты экспериментов подробно описаны в п. 5. Подводя итоги lo-fi экспериментов, можно отметить, что предложенные модификации позволили найти новую точку оптимума для "искуственного"ЧЯ при достаточно низких бюджетах. При этом все алгоритмы, кроме TS находят точку лучше точки baseline модели. Также заметим, что именно метод

"искуственного"ЧЯ позволил нам заранее отсеить заведомо менее эффективные модели, что заметно сократило затраты на реальные эксперименты. Это позволяет считать цель работы выполненной.

По итогам проведенной работы можно сформулировать следующие дальнейшие направления исследования:

- исследование методов EI, UCB и PI на реальном ЧЯ;
- изучение более сложных алгоритмов суррогатной оптимизации (например, при использование градиентного бустинга в качестве суррогатной модели)
- переход от low-fidelity ("искуственный"ЧЯ) к multi-fidelity (реальный ЧЯ) модели;
- проведение эксперимента на реальном ЧЯ;
- изучение возможностей комбинирования различных алгоритмов суррогатной оптимизации для улучшения качества оптимизации;
- использование метода "искуственного"ЧЯ для других задач суррогатной оптимизации для уменьшения затрат при выборе алгоритмов оптимизации;

Дальнейшие исследования в этих направлениях могут привести к развитию более эффективных и мощных методов суррогатной оптимизации, которые могут быть применены для решения сложных оптимизационных задач в различных областях.

## Список литературы


1. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System. — 2009. — Май. — URL: http://www.bitcoin.org/bitcoin.pdf.
2. *Ursache C.*, *Sammeth M.*, *Alboaie S.* OpenDSU: digital sovereignty in PharmaLedger // Frontiers in Blockchain. — 2022.
3. *Wisessing K.* [и др.]. The Prototype of Thai Blockchain-based Voting System // International Journal of Advanced Computer Science and Applications. — 2020. — Янв. — Т. 11. — DOI: 10.14569/IJACSA.2020.0110510.
4. *Thakkar P.*, *Nathan S.*, *Viswanathan B.* Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform // 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). — 2018. — С. 264—276. — DOI: 10.1109/MASCOTS.2018.00034.
5. *Amelin V.* [и др.]. Black-Box for Blockchain Parameters Adjustment // IEEE Access. — 2022. — Т. 10. — С. 101795—101802. — DOI: 10.1109/ACCESS.2022.3208702.
6. *Koziel S.*, *Leifsson L.* Surrogate-Based Modeling and Optimization: Applications in Engineering. — 08.2013. — ISBN 978-1-4614-7550-7. — DOI: 10.1007/978-1-4614-7551-4.
7. *Garnett R.* Bayesian Optimization. — Cambridge University Press, 2023. — С. 19—31.
8. *Han Z.-H.*, *Zhang s.* Surrogate-Based Optimization //. — 03.2012. — ISBN 978-953-51-0146-8. — DOI: 10.5772/36125.
9. *Kuhn M.*, *Johnson K.* Applied Predictive Modeling. — Springer, 2013. — ISBN 978-1-4614-6848-6.

10. *Friedman J.* Greedy Function Approximation: A Gradient Boosting Machine // The Annals of Statistics. — 2000. — Нояб. — Т. 29. — DOI: 10.1214/aos/1013203451.

11. *Snoek J.*, *Larochelle H.*, *Adams R.* Practical Bayesian Optimization of Machine Learning Algorithms // Advances in Neural Information Processing Systems. — 2012. — Июнь. — Т. 4.

12. *Mockus J.* Application of Bayesian approach to numerical methods of global and stochastic optimization // Journal of Global Optimization. — 1994. — Т. 4. — С. 347—365.

# Приложение

## Приложение 1. Результаты работы алгоритмов суррогатной оптимизации

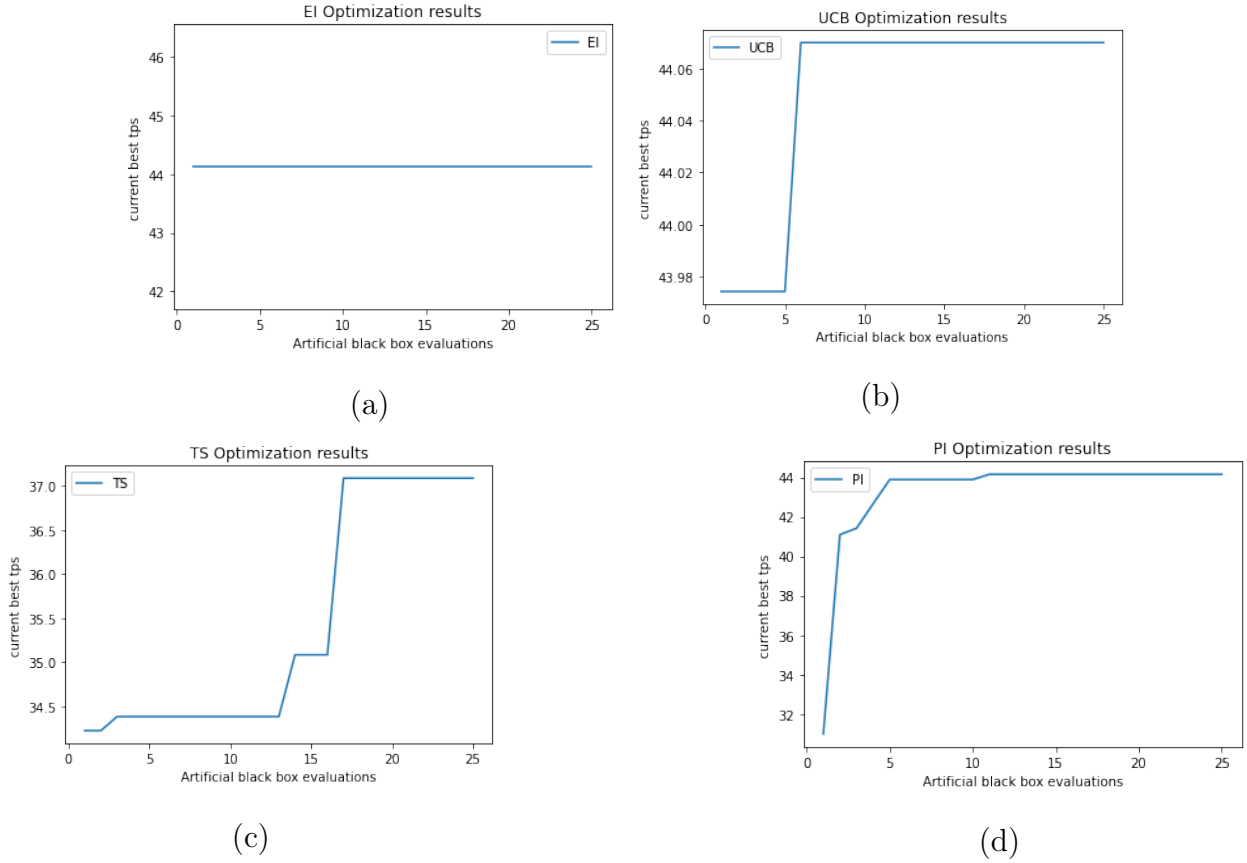
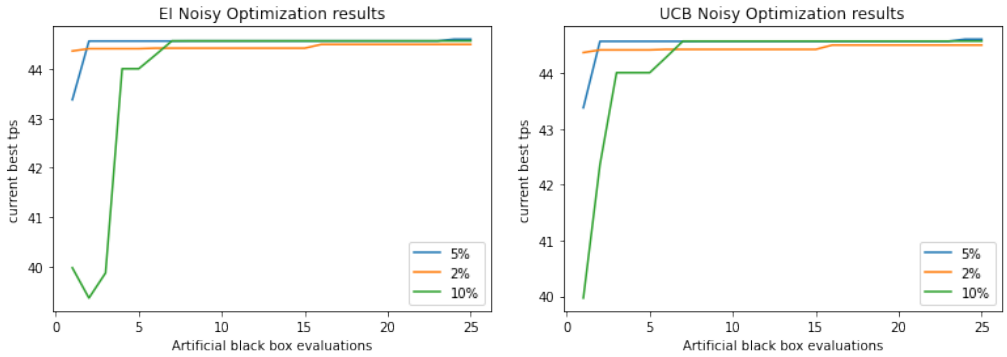


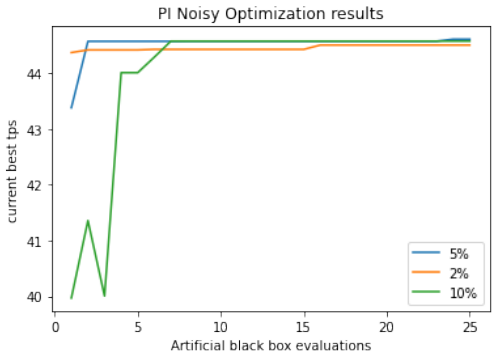

Рис. 6.1: Максимальный tps на каждом этапе итерации для методов (EIa), UCB(b), TS (c)и PI(d)




(a)




(b)




(c)

Рис. 6.2: Устойчивость к шуму для методов (EIa), UCB(b), и PI(c)