Адаптивное распределение вычислительного бюджета в многоролевом анализе программных задач

Халилов Джабраиль Эльнурович
Национальный исследовательский университет «Высшая школа экономики»
Исследовательский проект Xynapse / BVC
github.com/jabrailkhalil/xynapse

2026

Аннотация

Многоролевой анализ программной задачи позволяет до изменения кода независимо проверить локализацию дефекта, стратегию исправления, затрагиваемые зависимости и тестовое покрытие. Bounded Verification Council (BVC) превращает такую проверку из режима с постоянной ценой в управляемый сервис: второй раунд критики запускается только после формализованной проверки расхождения и полноты, а верхняя граница бюджета известна до старта. Эксперимент выполнен на 60 реальных задачах SWE-bench Verified из 11 открытых Python-репозиториев. Адаптивный алгоритм обработал 32 задачи за 6 модельных вызовов и направил 28 задач на расширенный 10-вызовный маршрут. Относительно совета с обязательной критикой число вызовов уменьшилось с 600 до 472 (на 21.3%), объём токенов — с 18 365 172 до 14 282 809 (на 22.2%), расчётная стоимость — с 2 402,61 до 1 838,46 руб. (на 23.5%). BVC решил 5/60 задач, полный совет — 6/60; пять успешных задач совпали, а одна была решена только полным советом (разность $-1{,}7$ п. п., $p = 1{,}0$). Таким образом, BVC продемонстрировал практически значимую способность избирательно расходовать inference-time бюджет и дал прозрачное объяснение каждого решения gate; равенство, неухудшаемость и превосходство функционального качества не установлены.

Ключевые слова: большие языковые модели, программная инженерия, многоагентные системы, адаптивная критика, вычислительный бюджет, SWE-bench Verified, BVC.

1. Введение

Современный AI-ассистент способен не только сформировать patch, но и разложить задачу на несколько независимых инженерных позиций. Архитектор локализует причину и границы изменения; разработчик предлагает конкретную реализацию; рецензент проверяет инварианты и совместимость; тестировщик формулирует наблюдаемое поведение. Многоролевые диалоги и iterative refinement уже используются как способы увеличить полноту рассуждения [3, 7, 12, 14]. Практическая проблема состоит в том, что каждый дополнительный раунд оплачивается вызовами модели, входным контекстом и выходными токенами.

Архитектура Xynapse IDE и первоначальная модель Bounded Verification Council были сформулированы в выпускной квалификационной работе автора [19]. Настоящее исследование продолжает дипломную работу исполняемой проверкой одного конкретного свойства: может ли заранее включаемый человеком многоролевой анализ тратить полный бюджет избирательно, а не на каждой задаче. Актуальная программная реализация Xynapse развивается в открытом репозитории проекта [18]; ссылка на репозиторий фиксирует кодовую реализацию, тогда как диплом остаётся источником архитектуры и постановки метода.

В статье рассматривается не неограниченная дискуссия агентов, а управляемый процесс с двумя заранее известными маршрутами. После независимого первого раунда вычисляются две диагностические величины. Если позиции достаточно полны и согласованы, система сразу синтезирует план; иначе выполняется ровно один раунд взаимной критики. Поэтому верхняя граница цены известна до запуска, а фактическая цена определяется наблюдаемым состоянием анализа.

Вклад работы

1. задана исполняемая функция маршрутизации между 6 и 10 вызовами;
2. проведено парное сравнение BVC с полным советом на 60 одних и тех же задачах и одинаковом кодовом контексте;
3. экономия разложена по вызовам, токенам, рублям, задачам и успешным решениям;
4. качество patch сохранено как обязательная контрольная метрика: все результаты прошли официальный тестовый контур SWE-bench Verified.

Практическая сила BVC

Алгоритм объединяет четыре свойства, редко доступные одновременно:

- **предсказуемость:** до запуска известен диапазон 6–10 вызовов;
- **адаптивность:** полный бюджет оплачивается только при измеримом расхождении или неполноте первого раунда;
- **объяснимость:** решение gate восстанавливается по D_{vote}, D_{cov} и замороженным порогам;
- **измеримость:** экономия получена в end-to-end запуске и рассчитана по фактическим вызовам, токенам и тарифным категориям.

Тем самым BVC делает многоролевой анализ не дорогой опцией «всё или ничего», а контролируемым уровнем инженерной проверки.

2. Связанные подходы и постановка задачи

AutoGen формализует многоагентный разговор как композицию ролей и сообщений [14]; multiagent debate исследует ценность независимых позиций и последующего обмена аргументами [3]. Reflexion и Self-Refine показывают общий принцип улучшения ответа во время генерации через обратную связь и повторную переработку [7, 12]. Adaptive-Consistency вводит остановку по наблюдаемому согласию выборок, а FrugalGPT рассматривает адаптивный каскад моделей как способ управлять ценой [1, 2]. Во всех этих схемах дополнительное рассуждение является ресурсом. Инженерная задача состоит не только в том, чтобы иметь возможность потратить больше вычислений, но и в том, чтобы определить достаточный момент остановки.

BVC отличается тремя ограничениями. Во-первых, независимые ответы ролей структурированы по общим осям, поэтому сравниваются решения, а не свободная формулировка. Во-вторых, второй раунд ограничен единицей. В-третьих, gate заранее заморожен и не изменяется после просмотра результатов. Исследовательский вопрос формулируется так:

> *Насколько адаптивный BVC сокращает фактический расход относительно полного совета с обязательной критикой и какой функциональный результат наблюдается при этом сокращении?*

3. Алгоритм BVC

3.1. Роли и четыре оси решения

Первый раунд состоит из четырёх независимых вызовов. Каждая роль возвращает типизированные решения по множеству

$$\mathcal{A} = \{\mathit{root\ cause}, \mathit{fix\ strategy}, \mathit{dependencies}, \mathit{test\ coverage}\}.$$

Для локализации, поверхности изменения и покрытия используется расстояние Жаккара между множествами [6]; для стратегии — расстояние между значениями замороженного категориального словаря. Свободный пояснительный текст в голосование не входит. Для двух пустых множеств расстояние принято равным

нулю; для пустого и непустого — единице. Это соглашение совпадает с реализацией, зафиксированной до основной серии.

Пусть V_a — множество валидных решений ролей по оси a, а $d_a(i,j) \in [0,1]$ — расстояние между двумя решениями. При $|V_a| \geq 2$ среднее попарное расхождение равно

$$D_a = \frac{2}{|V_a|(|V_a|-1)} \sum_{\substack{i<j \\ i,j \in V_a}} d_a(i,j).$$

Общая величина расхождения вычисляется только по осям, на которых доступны минимум два валидных голоса:

$$D_{\text{vote}} = \frac{1}{|\mathcal{A}^*|} \sum_{a \in \mathcal{A}^*} D_a, \qquad \mathcal{A}^* = \{a : |V_a| \geq 2\}.$$

Второй показатель отражает не различие ответов, а потерю структуры. Из $4 \times 4 = 16$ ячеек «роль–ось» подсчитываются отсутствующие или невалидные:

$$D_{\text{cov}} = \frac{N_{\text{missing}} + N_{\text{invalid}}}{16}.$$

Такое разделение важно: единодушно пропущенная зависимость может иметь низкое D_{vote}, но высокое D_{cov}.

3.2. Замороженный gate и число вызовов

Пороговые значения до запуска основной серии были зафиксированы как

$$\tau_{\text{vote}} = 0{,}30, \qquad \tau_{\text{cov}} = 0{,}25.$$

Это инженерные значения этапа проектирования: они не подбирались по outcomes 60 задач и не заявляются оптимальными. Их назначение — сделать правило исполняемым до просмотра результата; post-hoc чувствительность маршрутов к порогам анализируется отдельно. Поскольку D_{vote} не определено при $\mathcal{A}^* = \varnothing$, правило записывается в охраняемой форме:

$$I_{\text{crit}} = \begin{cases} 1, & \mathcal{A}^* = \varnothing, \\ \mathbb{1}\left[D_{\text{vote}} > \tau_{\text{vote}} \ \vee \ D_{\text{cov}} > \tau_{\text{cov}}\right], & \mathcal{A}^* \neq \varnothing. \end{cases}$$

Общее число успешных модельных вызовов на задачу:

$$C_{\text{BVC}} = 4 + 4I_{\text{crit}} + 1 + 1 = 6 + 4I_{\text{crit}}.$$

Первые четыре слагаемых — независимые роли; при необходимости ещё четыре вызова выполняют критику; затем один вызов синтезирует план и один формирует patch. Полный совет всегда использует $C_{\text{fixed}} = 10$.

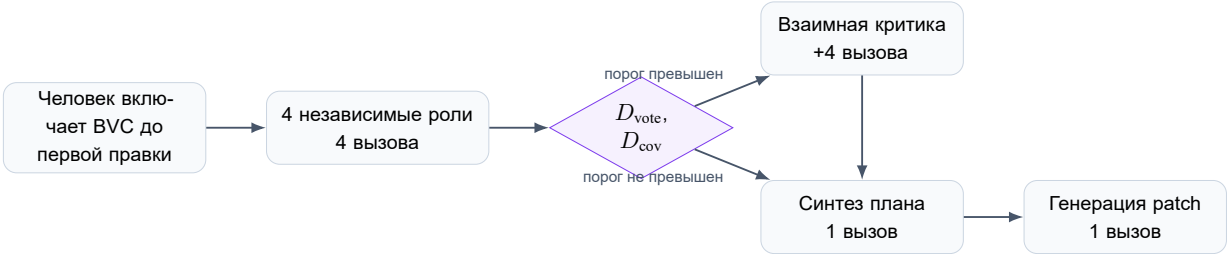


Рис. 1: Два ограниченных маршрута BVC: 6 либо 10 модельных вызовов.

3.3. Аналитическая экономия по числу вызовов

Пусть $p = \Pr(I_{\text{crit}} = 1)$ — наблюдаемая доля задач с критикой. Среднее число вызовов тогда равно

$$\mathbb{E}[C_{\text{BVC}}] = 6 + 4p,$$

а относительная экономия против полного совета:

$$S_C = 1 - \frac{6 + 4p}{10} = 0{,}4(1 - p).$$

В эксперименте $p = 28/60 = 0{,}4667$, поэтому

$$\mathbb{E}[C_{\text{BVC}}] = 7{,}8667, \qquad S_C = 0{,}2133.$$

Это точно соответствует фактическим $472 = 32 \cdot 6 + 28 \cdot 10$ вызовам и экономии $128 = 600 - 472$ вызовов.

3.4. Расчёт стоимости

Для каждого запуска сохранялись некэшированные входные токены T_u, кэшированные входные токены T_c и выходные токены T_o. По замороженному тарифу стоимость вычислялась как

$$R = 0{,}30\frac{T_u}{1000} + 0{,}075\frac{T_c}{1000} + 0{,}50\frac{T_o}{1000} \quad \text{руб.}$$

Это расчётный тариф, замороженный в протоколе до основной генерации, а не заявление о фактическом счёте или будущей цене провайдера. Он нужен только для одинакового сопоставления методов внутри одной экспериментальной матрицы; пересчёт по другому тарифу не требует повторной генерации.

4. Экспериментальный дизайн

4.1. Задачи и общий контекст

Использована outcome-blind выборка из 60 задач SWE-bench Verified [5, 9]: 30 задач страты medium, 27 — hard и 3 — very hard. Из замороженной ревизии на 500 задач были взяты все три instance трудоёмкостью более четырёх часов, 27 из 42 задач страты 1–4 часа и 30 из 261 задачи страты 15 минут–1 час; задачи проще 15 минут не включались, поскольку BVC предполагается включать осознанно для нетривиальной работы. Квоты репозиториев распределялись пропорционально квадратному корню из размера доступного пула, а внутри квоты порядок определял SHA-256 от фиксированного seed и `instance_id`. Issue, эталонный patch и результаты моделей в отборе не участвовали.

Задачи относятся к 11 репозиториям, включая Django, SymPy, Sphinx, Astropy, pytest, xarray и scikit-learn. Для всех методов по каждой задаче передавались одно и то же issue и одни и те же исходные файлы. Использован замороженный результат BM25 retrieval [11]: от двух до восьми файлов, от 71 369 до 111 192 символов на задачу, всего 245 файлов и 5 691 570 символов. Из официального retrieval-артефакта извлекались только блоки исходного кода; демонстрационный patch из legacy-поля `text` в prompt не передавался. Все 245 файлов были независимо сверены с нужными base commit побайтно.

4.2. Сравниваемые методы

Для ответа на вопрос об адаптивном бюджете ресурсно наиболее сопоставимая пара — BVC и полный совет. У них совпадают роли, схема решений, синтезатор, финальная инструкция patch и верхняя граница в 10 вызовов. Единственное операционное различие состоит в том, что полный совет всегда проводит критику, а BVC применяет замороженный gate. Дополнительными ориентирами служат прямой patch одним вызовом и один предварительный план плюс patch двумя вызовами.

Таблица 1: Матрица методов основной серии.

Метод	Предварительный анализ	Вызовы на задачу	Запусков
Direct	отсутствует	1	60
Один план	один структурированный план	2	60
BVC	4 роли; критика по gate; синтез	6 или 10	60
Полный совет	4 роли; обязательная критика; синтез	10	60

Все методы использовали одну версию модели, temperature $= 0$, одинаковые лимиты финального patch и официальный исполняемый evaluator. Единицей анализа являлась задача; результаты сопоставлялись попарно. Доля решённых задач дополнена 95% Wilson-интервалом [13]; парные различия — точным двусторонним критерием Мак-Немара [8] и стратифицированным paired bootstrap на 10 000 выборок [4].

4.3. Что именно было запущено

Конфигурация генерации была общей для всей основной матрицы: Yandex Cloud AI Studio, запрошенный идентификатор `deepseek-v4-flash`, возвращённый URI `gpt://deepseek-v4-flash/latest`, temperature $= 0$, prompt-версия `main60-3.0.0`. Provider seed отсутствовал, поэтому нулевая температура не трактовалась как гарантия детерминизма. Отдельный параметр thinking/non-thinking в запросе не передавался: использовался режим по умолчанию данного URI, а возвращённое поле `reasoning_content` сохранялось [16]. Лимит ответа роли и плана/синтеза составлял 4 096 токенов, лимит финального unified diff — 16 384 токена. Порядок четырёх методов вращался по детерминированному расписанию `bvc-main60-schedule-v1`, чтобы один метод не выполнялся систематически первым или последним.

Для каждого `instance_id` процедура состояла из шести проверяемых этапов:

1. загрузить только issue, repository, base commit и общий context-manifest; difficulty, gold patch, test patch, `FAIL_TO_PASS` и `PASS_TO_PASS` были закрыты от модели;
2. выполнить замороженную последовательность вызовов выбранного метода;
3. сохранить prompt, ответ, usage, finish reason, лимит и SHA-256 каждого вызова;
4. извлечь один repository-root unified diff без ручного ремонта;
5. применить его к чистому checkout нужного base commit;
6. запустить официальный контейнерный evaluator: успехом считался только проход всех обязательных fail-to-pass и pass-to-pass тестов.

Шесть отдельных development-задач использовались для отладки формата и harness и навсегда исключены из 60-задачной статистики. Пустой, неразбираемый или неприменимый patch оставался неуспешным outcome, а не удалялся из данных.

5. Результаты

5.1. Маршрутизация gate

Короткий маршрут был выбран для 32/60 задач (53,3%), расширенный — для 28/60 (46,7%). Следовательно, механизм действительно адаптировал бюджет: он не выродился ни в постоянные 6, ни в постоянные 10 вызовов.

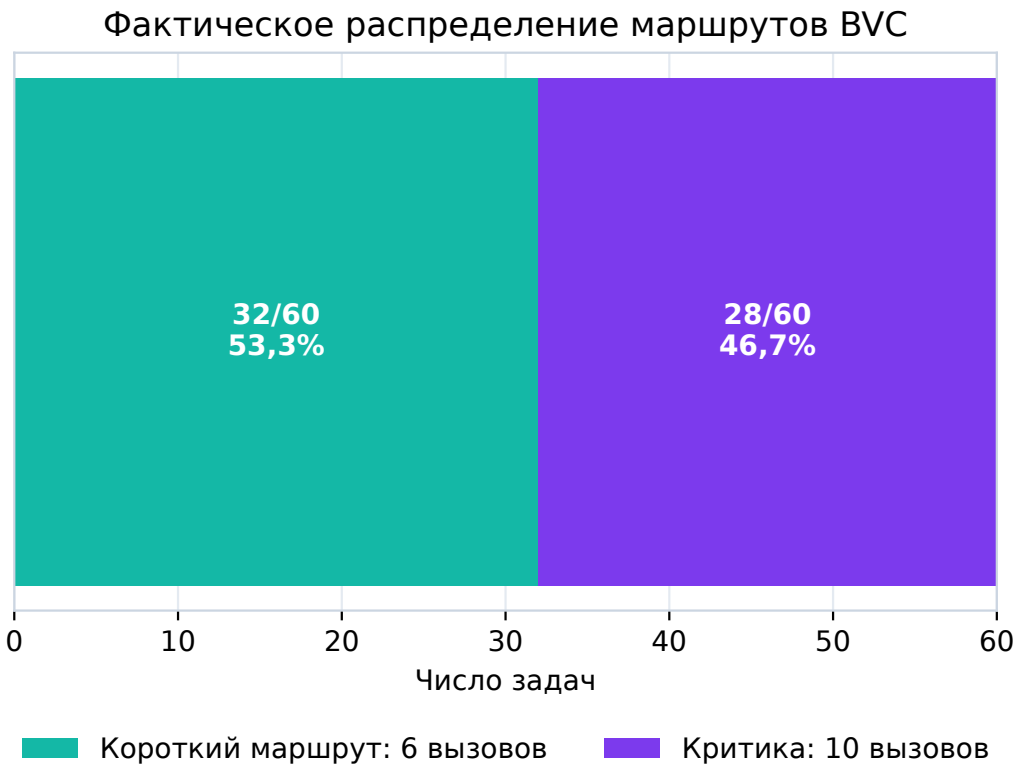


Рис. 2: Фактическая доля двух маршрутов BVC.

Частота критики составила 43,3% в medium, 48,1% в hard и 66,7% в very hard. Последняя оценка основана только на трёх задачах и потому показывается как описание выборки, а не как устойчивый тренд.

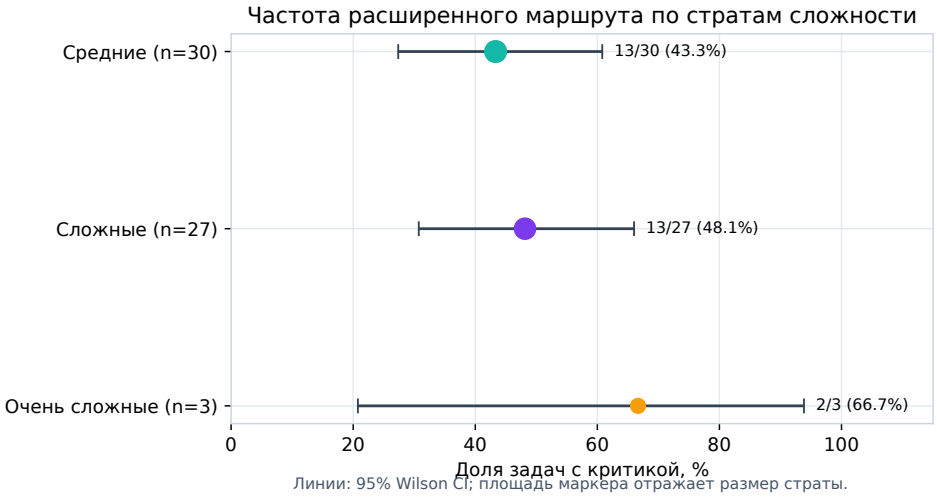
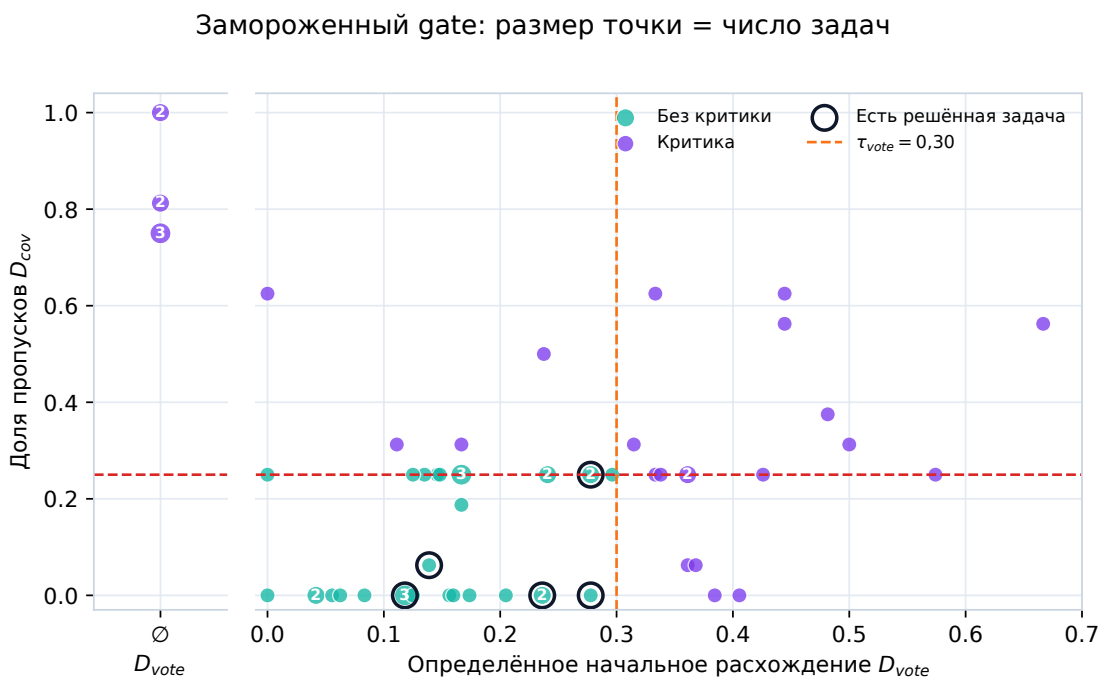


Рис. 3: Частота расширенного маршрута в трёх заранее заданных стратах сложности.

На плоскости $(D_{\text{vote}}, D_{\text{cov}})$ видна работа двух числовых порогов. Часть расширенных маршрутов определяется также третьим условием — отсутствием оси с двумя валидными голосами. В первом раунде все четыре оси имели минимум два валидных голоса у 35 задач, три оси — у 18, ни одной — у 7; других значений не наблюдалось. Для 7 последних задач D_{vote} не определялось. Среди остальных $n = 53$ задач корреляция Пирсона между D_{vote} и D_{cov} равна $r = 0.412$: показатели связаны, но не дублируют друг друга.



∅: нет оси с двумя валидными голосами; пунктир: строгие пороги $>$.

Рис. 4: Начальные диагностические значения, решение gate и функциональный исход.

Post-hoc анализ порогов изменяет только контрфактическое решение маршрутизации на уже сохранённых диагностических значениях. Он не пересчитывает планы, patch или outcomes и потому не является поиском лучшего порога по качеству.



Меняются только маршруты и аналитическая цена; функциональные outcomes не пересчитываются.

Рис. 5: Чувствительность доли критики и аналитической экономии вызовов к порогам; оранжевым отмечена замороженная конфигурация.

5.2. Экономия вызовов, токенов и стоимости

Таблица 2: Ресурсный результат BVC относительно полного совета.

Метрика	Полный совет	BVC	Экономия
Модельные вызовы	600	472	128 (21.3%)
Токены	18 365 172	14 282 809	4 082 363 (22.2%)
Расчётная стоимость, руб.	2 402,61	1 838,46	564,15 (23.5%)
Стоимость на задачу, руб.	40,04	30,64	9,40

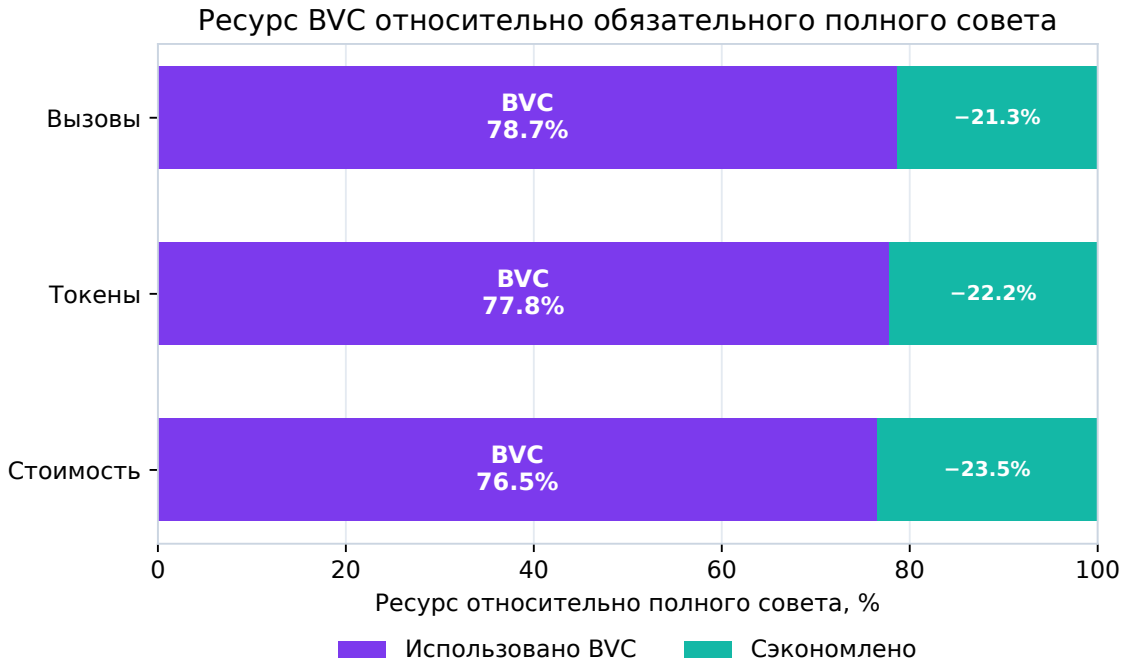


Рис. 6: Нормированный расход: фиолетовая часть использована BVC, бирюзовая часть сэкономлена относительно фиксированных 100% полного совета.

В среднем на одной задаче BVC сохранил 68 039 токенов, 2,13 вызова и 9,40 руб. При этом экономия токенов (22,2%) близка к экономии вызовов (21,3%). Это показывает, что основной механизм результата — именно пропуск четырёх critique-вызовов, а не случайное сокращение длины отдельных ответов.

Фактический объём по задачам неоднороден. Медиана BVC равна 190 866 токенов, межквартильный диапазон — от 180 212 до 308 979; у полного совета медиана равна 307 980, диапазон — от 298 149 до 318 023. Двухмодальная структура BVC закономерно отражает два маршрута.

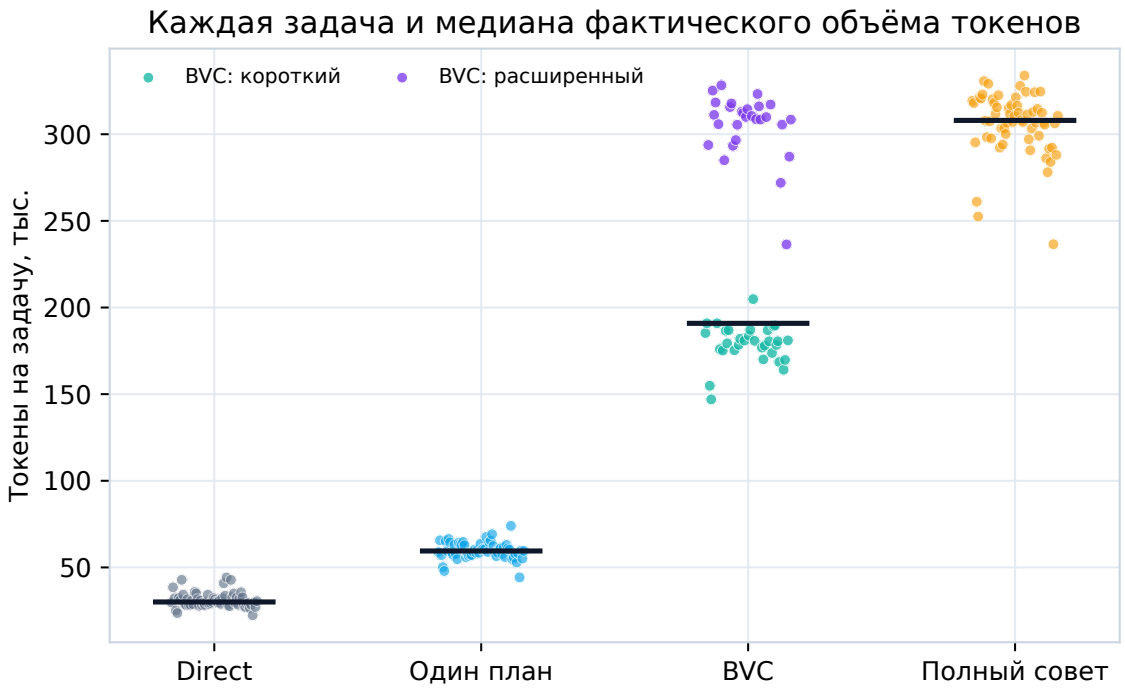


Рис. 7: Каждая задача и медиана токенов; для BVC цвет разделяет два маршрута.

Таблица 3: Подгруппы BVC по фактически выбранному маршруту.

Маршрут	Задач	Вызовов	Средние токены	Средняя цена, руб.
6 вызовов, без критики	32	192	179 160	22,78
10 вызовов, с критикой	28	280	305 345	39,62

Короткий маршрут в среднем стоил 22,78 руб. на задачу, расширенный — 39,62 руб. Разность 16,84 руб. описывает разность подгрупп, но не является чистой каузальной ценой второго раунда: состав задач и длины ответов также различаются. Все пять успешных BVC-задач оказались в короткой подгруппе; расширенная подгруппа содержала больше hard и very hard задач. Эти числа описывают маршрутизацию наблюдаемой выборки и не позволяют каузально заключить, что отказ от критики повышает качество.

5.3. Парная экономия по задачам

Агрегированная сумма может скрывать задачи, на которых адаптивный маршрут оказался длиннее из-за вариации ответа. Поэтому для каждого instance вычислена парная разность

$$\Delta T_i = T_{i,\text{fixed}} - T_{i,\text{BVC}}.$$

Положительное значение означает экономию BVC. Сумма всех ΔT_i равна 4 082 363 токенов. На 11/60 задачах $\Delta T_i < 0$, то есть BVC израсходовал больше токенов; максимальный перерасход составил 13 971 токенов. Все такие случаи относятся к расширенному маршруту, где оба режима выполняли по 10 вызовов, а разность создавалась длиной независимо сгенерированных ответов.

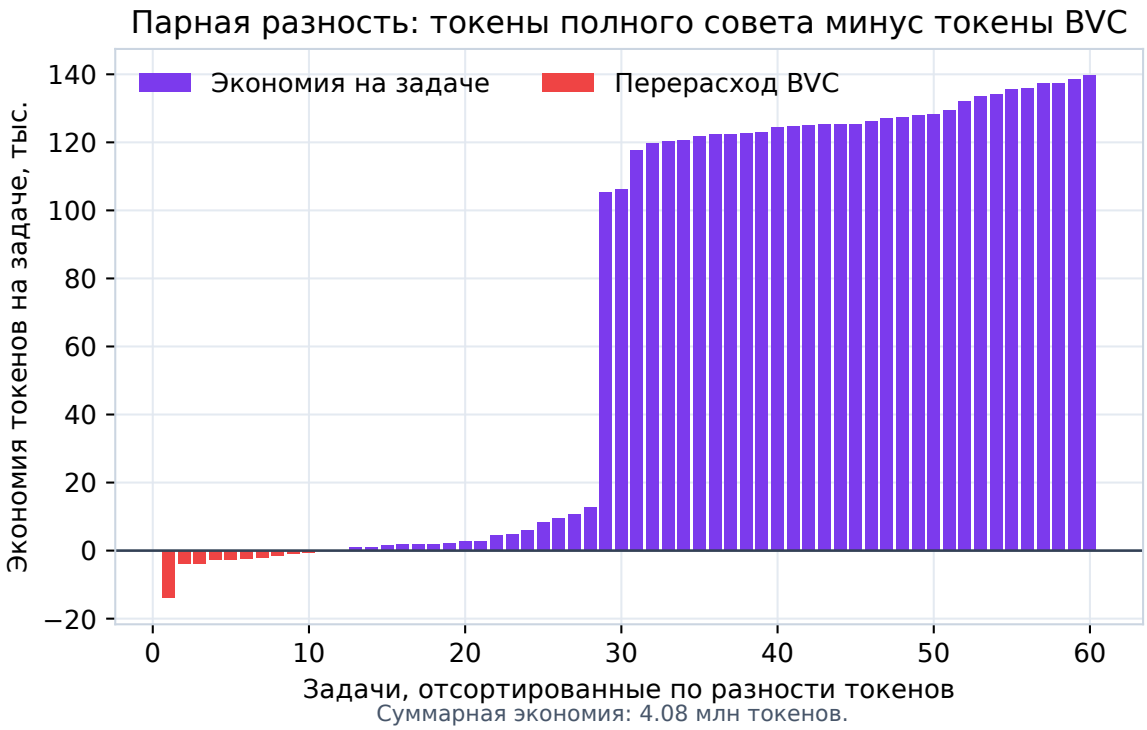


Рис. 8: Распределение парной экономии токенов по 60 задачам.

Для проверки того, что агрегированная экономия не создаётся несколькими выбросами, дополнительно рассчитаны парные разности $\Delta X_i = X_{i,\text{fixed}} - X_{i,\text{BVC}}$. Это расширенный описательный анализ, выполненный после основной замороженной проверки: среднее повторно оценивалось в 10 000 bootstrap-выборках внутри тех же трёх страт. Средняя экономия на задачу составила 2,13 вызова, 68 039 токенов и 9,40 руб. Соответствующие 95% интервалы равны $[1{,}60; 2{,}67]$ вызова, $[51\,980; 83\,480]$ токенов и $[6{,}75; 12{,}05]$ руб. Интервалы приведены как описание разброса на этой замороженной смеси страт. Положительность экономии вызовов следует из конструкции gate при доле критики меньше единицы и не является отдельной проверяемой гипотезой; для токенов и рублей отдельные парные разности могут быть отрицательными.



Рис. 9: Средняя парная экономия и 95% стратифицированные bootstrap-интервалы; положительный знак означает меньший расход BVC.

Медианы парной экономии равны 111 910 токенов и 7,92 руб. Эти величины не обязаны иметь один и

тот же коэффициент пересчёта: рублёвая цена является линейной комбинацией трёх категорий токенов с разными весами, а их доли меняются между задачами. Поэтому медиана суммы токенов не определяет медиану стоимости. Прямой пересчёт всех 60 пар даёт именно 7,92 руб.; на 10/60 задачах наблюдался рублёвый перерасход, максимальный — 12,84 руб.



Рис. 10: Связь экономии всех токенов с тарифно-взвешенной экономией стоимости.

5.4. Из чего складывается денежная экономия

Стоимость разложена на кэшированный вход, некэшированный вход и выход. Поскольку каждая роль повторно получает значительную часть общего кодового контекста, кэширование заметно снижает цену входа, но не устраняет стоимость выходных решений и critique-ответов.

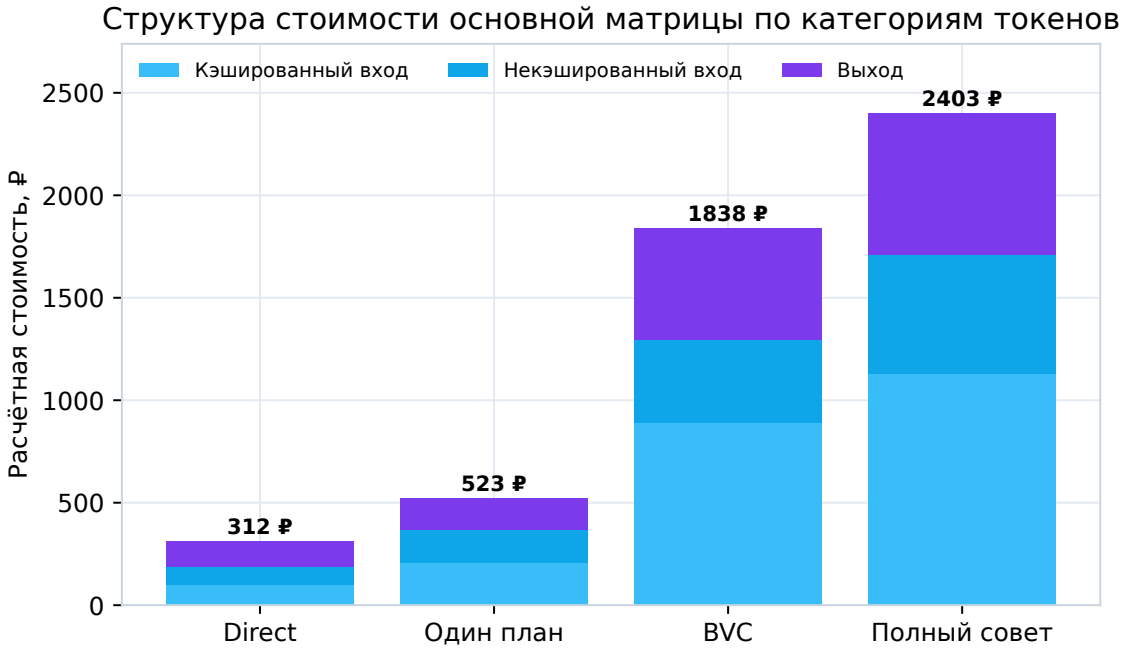


Рис. 11: Компоненты расчётной стоимости основной серии.

5.5. Функциональный результат как контроль экономии

Официальный evaluator признал решёнными 5/60 задач BVC и 6/60 задач полного совета. Парная таблица имеет вид:

Таблица 4: Попарное пересечение функциональных исходов BVC и полного совета.

	Полный совет: решено	Полный совет: не решено	Всего
BVC: решено	5	0	5
BVC: не решено	1	54	55
Всего	6	54	60

Разложение по фактическому маршруту уточняет источник единственной дискордантной пары:

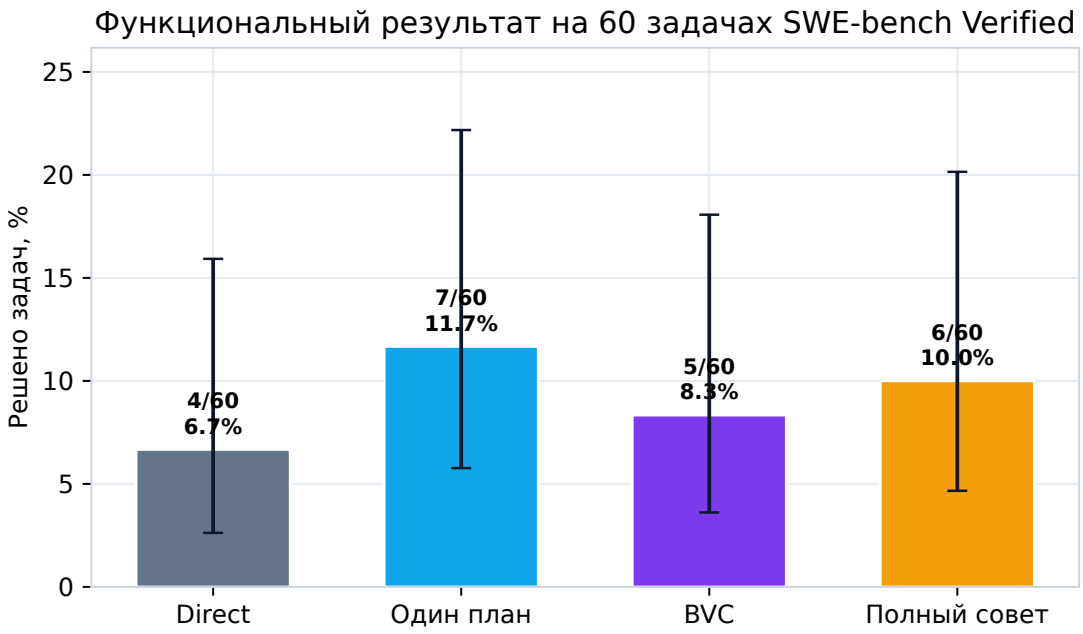
Таблица 5: Функциональные исходы BVC и полного совета по маршруту BVC.

Маршрут BVC	Задач	BVC решено	Полный совет решено
Короткий, 6 вызовов	32	5	5
Расширенный, 10 вызовов	28	0	1

Единственная задача, решённая только полным советом (`django__django-13128`), принадлежит расширенной подгруппе. В ней оба режима выполнили критику и имели одинаковую структуру из 10 вызовов. Поэтому наблюдаемую разность нельзя приписать решению gate о пропуске критики; при недоступном provider seed два структурно одинаковых запуска всё равно могли сгенерировать разные планы и patch.

Разность долей равна $-1{,}7$ п. п., 95% стратифицированный bootstrap-интервал — $[-5{,}0;\,0{,}0]$ п. п., точное $p_{\text{McNemar}} = 1{,}0$. Следовательно, данные не доказывают равенство методов, но показывают, что пять из шести успехов полного совета сохранены при сокращении ресурсов более чем на одну пятую. Нулевая верхняя граница bootstrap-интервала следует из пустой клетки «только BVC»: в каждой resampling-реплике разность BVC–полный совет неположительна. Это свойство наблюдаемой матрицы, а не самостоятельное свидетельство о знаке эффекта в генеральной совокупности.

Для каждой отдельной доли Wilson-интервалы широки: BVC 8,3% $[3{,}6;\,18{,}1]$, полный совет 10,0% $[4{,}7;\,20{,}1]$. График всех методов нужен для контекста, а вывод об экономии строится на структурно наиболее близкой паре BVC–полный совет.



Вертикальные интервалы: 95% Wilson CI для каждой доли; сравнение методов является парным.

Рис. 12: Доли решённых задач и 95% Wilson-интервалы.

В общем протоколе заранее заявленным первичным сравнением было BVC против одного плана с допустимой потерей -5 п. п.. Его интервал $[-10{,}0; +3{,}3]$ пересекает этот порог: нижняя граница лежит ниже -5 п. п., поэтому неухудшаемость BVC не установлена.

5.6. Цена одного наблюдаемого успеха и точка безубыточности

Описательная стоимость одного решённого экземпляра задачи:

$$R_{\text{success}} = \frac{R_{\text{total}}}{N_{\text{resolved}}}.$$

Для BVC она составила 367,69 руб., для полного совета — 400,44 руб., то есть на 8.2% меньше. Показатель не является независимой статистической оценкой: знаменатель мал, а решённые задачи различаются. Чувствительность к изменению знаменателя на одну задачу показывает хрупкость этой нормировки:

Успехов BVC	4	5 (наблюдалось)	6
Руб./успех	459,62	367,69	306,41

Наблюдаемая цена полного совета равна 400,44 руб./успех. Следовательно, операционная нормировка BVC лучше только при пяти и более успехах; потеря одного успеха меняет её направление. Это иллюстрация чувствительности, а не доказательство экономического превосходства.

6. Практическая ценность и применение

6.1. Что именно доказано измерениями

Наиболее сильный вывод не требует экстраполяции качества. Gate реально выбрал оба маршрута и тем самым:

- исключил 128 из 600 потенциальных вызовов;
- сократил объём на 4 082 363 токенов;
- уменьшил расчётную стоимость на 564,15 руб.;

- сохранил пять общих с полным советом функциональных успехов.

Это является прямой эмпирической демонстрацией адаптивной экономии. В среднем на одну задачу алгоритм исключил 2,13 вызова, 68 039 токенов и 9,40 руб. расчётной стоимости. Главная польза здесь операционная: команда получает глубокую проверку там, где gate обнаруживает основания для неё, не оплачивая одинаковый раунд критики для каждой задачи.

6.2. Как использовать результат в Xynapse

Дипломная архитектура Xynapse предполагает явное включение BVC до первой правки [19]. Эксперимент уточняет продуктовый контракт. Интерфейс может заранее показывать диапазон 6–10 вызовов, после первого раунда — фактическое решение gate и его числовые основания, после завершения — токены и расчётную стоимость. Пользователь получает не неограниченный «рой», а проверяемый этап планирования с фиксированной верхней границей.

Для мониторинга полезны четыре метрики:

1. доля задач, направленных на критику;
2. средняя и 95-й процентиль токенов на маршрут;
3. функциональный outcome после официальных тестов;
4. цена решённой задачи относительно полного совета.

При смене модели, тарифа или контекстного окна те же формулы позволяют пересчитать экономику без изменения определения алгоритма.

7. Границы интерпретации

Эксперимент использует 60 задач и одну конфигурацию модели; низкое число успехов создаёт широкие интервалы качества. Наблюдаемая цена зависит от замороженного тарифа. Расширенный маршрут получал задачи после диагностического сигнала, поэтому подгруппы короткого и полного маршрутов не рандомизированы. Не проверялись равнобюджетные альтернативы вроде best-of-10 direct, self-consistency или цикла «patch–ошибка–повтор». Поэтому работа изолирует экономию адаптивного gate относительно собственной неадаптивной версии, но не доказывает, что многоролевая декомпозиция является лучшим способом потратить 6–10 вызовов. Контекст был ограничен замороженным BM25-пакетом, а генератор не имел инструментального цикла с обратной связью тестов; абсолютные доли успеха относятся именно к этому однопроходному контуру. Agentless и SWE-agent показывают, что локализация, отбор patch и агентно-компьютерный интерфейс являются самостоятельными факторами результата [15, 17]. Кроме того, последующий аудит SWE-bench Verified показал остаточные проблемы тестов и возможную контаминацию; поэтому набор больше не рекомендуется как мера текущей frontier-способности моделей [10]. В данной работе он используется уже не как рейтинг модели, а как один замороженный исполняемый тест для парного сравнения четырёх методов на одинаковых задачах. Это сохраняет смысл утверждения «в этой матрице и при этом evaluator», но запрещает переносить проценты на всю разработку. По этим причинам статья утверждает адаптивную экономию, но не статистическое неухудшение и не преимущество точности.

Три подготовленные статьи используют одну и ту же основную матрицу из 240 запусков, но отвечают на разные вопросы: настоящая работа анализирует бюджет, сопутствующая методическая работа — аудит контура, а третья — заранее включаемый пользовательский сценарий и разговорную переформулировку. Это не три независимых эксперимента; совпадающие агрегаты между статьями должны совпадать буквально.

8. Заключение

BVC превратил многоролевую проверку из режима с постоянными 10 вызовами в ограниченный, объяснимый процесс с фактическим средним 7,87 вызова. На 60 реальных задачах это дало экономию 21,3% вызовов, 22,2% токенов и 23,5% расчётной стоимости относительно полного совета. Функциональный контроль зафиксировал 5 решений BVC против 6 у полного совета, причём пять успехов совпали. Практический результат состоит в том, что предложенный в дипломе Xynapse/BVC механизм [19] получил

количественно проверенную форму: полный inference-time бюджет можно включать по наблюдаемому расхождению, сохраняя заранее известную верхнюю границу, прозрачный учёт стоимости и воспроизводимое основание каждого решения. Именно это делает BVC практически сильным алгоритмом управления проверкой: он не обещает «магическое» улучшение, а даёт измеримый контроль над глубиной анализа.

Декларации

Внешнее финансирование и конфликт интересов отсутствуют. LLM-системы использовались в эксперименте как исследуемые генераторы, при подготовке разговорных переформулировок и при редакционной проверке рукописи. Все числа и графики повторно строятся программно из сохранённых JSON/JSONL; ответственность за дизайн, интерпретацию и окончательный текст несёт автор. ВКР [19] цитируется как источник архитектуры Xynapse/BVC, а не как источник экспериментальных чисел.

Литература

[1] P. Aggarwal, A. Madaan, Y. Yang и Mausam. «Let's Sample Step by Step: Adaptive-Consistency for Efficient Reasoning and Coding with LLMs». В: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023, с. 12375—12396. DOI: 10.18653/v1/2023.emnlp-main.761. URL: https://aclanthology.org/2023.emnlp-main.761/.

[2] L. Chen, M. Zaharia и J. Zou. «FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance». В: *arXiv preprint arXiv:2305.05176* (2023). URL: https://arxiv.org/abs/2305.05176.

[3] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum и I. Mordatch. «Improving Factuality and Reasoning in Language Models through Multiagent Debate». В: *arXiv preprint arXiv:2305.14325* (2023). URL: https://arxiv.org/abs/2305.14325.

[4] B. Efron и R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993. DOI: 10.1201/9780429246593.

[5] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press и K. Narasimhan. «SWE-bench: Can Language Models Resolve Real-World GitHub Issues?» В: *International Conference on Learning Representations*. 2024. URL: https://arxiv.org/abs/2310.06770.

[6] S. Kosub. «A Note on the Triangle Inequality for the Jaccard Distance». В: *Pattern Recognition Letters* 120 (2019), с. 36—38. URL: https://arxiv.org/abs/1612.02696.

[7] A. Madaan и др. «Self-Refine: Iterative Refinement with Self-Feedback». В: *Advances in Neural Information Processing Systems*. Т. 36. 2023. URL: https://arxiv.org/abs/2303.17651.

[8] Q. McNemar. «Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages». В: *Psychometrika* 12.2 (1947), с. 153—157. DOI: 10.1007/BF02295996.

[9] OpenAI. *Introducing SWE-bench Verified*. 13 авг. 2024. URL: https://openai.com/index/introducing-swe-bench-verified/ (дата обр. 28.07.2026).

[10] OpenAI. *Why SWE-bench Verified No Longer Measures Frontier Coding Capabilities*. 23 февр. 2026. URL: https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/ (дата обр. 28.07.2026).

[11] S. Robertson и H. Zaragoza. «The Probabilistic Relevance Framework: BM25 and Beyond». В: *Foundations and Trends in Information Retrieval* 3.4 (2009), с. 333—389. DOI: 10.1561/1500000019.

[12] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan и S. Yao. «Reflexion: Language Agents with Verbal Reinforcement Learning». В: *arXiv preprint arXiv:2303.11366* (2023). URL: https://arxiv.org/abs/2303.11366.

[13] E. B. Wilson. «Probable Inference, the Law of Succession, and Statistical Inference». В: *Journal of the American Statistical Association* 22.158 (1927), с. 209—212. DOI: 10.1080/01621459.1927.10502953.

[14] Q. Wu и др. «AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation». В: *arXiv preprint arXiv:2308.08155* (2023). URL: https://arxiv.org/abs/2308.08155.

[15] C. S. Xia, Y. Deng, S. Dunn и L. Zhang. «Agentless: Demystifying LLM-based Software Engineering Agents». В: *arXiv preprint arXiv:2407.01489* (2024). URL: https://arxiv.org/abs/2407.01489.

[16] Yandex Cloud. *Миллион токенов контекста: запускаем DeepSeek V4 Flash в Yandex AI Studio*. 28 мая 2026. URL: https://yandex.cloud/ru/blog/yandex-ai-studio-deepseek-v4-flash (дата обр. 03.08.2026).

[17] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan и O. Press. «SWE-agent: Agent–Computer Interfaces Enable Automated Software Engineering». В: *arXiv preprint arXiv:2405.15793* (2024). URL: https://arxiv.org/abs/2405.15793.

[18] Д. Э. Халилов. *Xynapse: интеллектуальная среда разработки с интегрированным AI-ассистентом*. GitHub. 2026. URL: https://github.com/jabrailkhalil/xynapse (дата обр. 03.08.2026).

[19] Д. Э. Халилов. «Интеллектуальная среда разработки с интегрированным AI-ассистентом для российского рынка». Выпускная квалификационная работа. Москва: Национальный исследовательский университет «Высшая школа экономики», 2026.