

Ajuste Automático de Parâmetros Bayesiano com Persistência de Contexto para Operadores Semânticos

Arthur Vinícius, Luiza Regi, Rafael Alexandre, Willian Barreiros Jr., George Teodoro

¹Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brasil

{arthurvinicius, luizaregi, rafael, willianjunior, george}@dcc.ufmg.br

Abstract. *The confidence interval-guided Multi-Armed Bandit (MAB) algorithm is one of the approaches employed to optimize execution plans in systems based on semantic operators over LLMs. However, such approaches typically reinitialize their operator estimates at each optimization process, without reusing knowledge accumulated from previous executions. This work reformulates the sampling process as a Bayesian procedure driven by Thompson Sampling, proposing the CXTBAY algorithm. With this new algorithm the posterior distributions are preserved across executions, and informative priors can be interpolated for previously unsampled operators using a Gaussian Process over operator embeddings. Compared to MAB, CXTBAY managed to find the same execution final plan 85% of the times, outperforming MAB's quality in the remaining cases. Further, this was achieved with at most $2.3\times$ less time and resources required in the optimization process.*

Resumo. *O Multi-Armed Bandit (MAB), guiado por intervalos de confiança, tem sido empregado na otimização de planos de execução em sistemas baseados em operadores semânticos sobre LLMs. Entretanto, essas abordagens normalmente reinicializam suas estimativas a cada nova execução, sem reutilizar o conhecimento acumulado em execuções anteriores. Este trabalho reformula a amostragem como um procedimento Bayesiano conduzido por Thompson Sampling, propondo o algoritmo CXTBAY, que persiste distribuições a posteriori entre execuções e interpola priors informativas, para operadores nunca amostrados, a partir de um Processo Gaussiano sobre embeddings de operadores. Comparado ao MAB, o CXTBAY conseguiu encontrar o mesmo plano final 85% das vezes, encontrando planos de qualidade superior nos demais casos. Isto foi feito com até $2,3\times$ menos tempo e recursos gastos no processo de otimização.*

1. Introdução

Os Grandes Modelos de Linguagem (*Large Language Models* – LLMs) impulsionaram uma nova geração de aplicações capazes de processar dados não estruturados por meio de operações semânticas, executando tarefas como extração de informações, classificação, sumarização e recuperação de conhecimento usando LLMs [Deleate et al. 2023, Thakur et al. 2021]. Esse avanço motivou o desenvolvimento de sistemas de operadores semânticos, que abstraem a complexidade da interação com diferentes modelos e permitem que desenvolvedores especifiquem apenas a lógica da aplicação.

Sistemas como o Palimpzest [Liu et al. 2025] e LOTUS [Patel et al. 2025] modelam aplicações como sequências de operadores lógicos, tais como *Map*, *Filter*, *Join* e

Aggregate, cada um dos quais pode ser concretizado por diferentes operadores físicos, que variam desde chamadas diretas a distintos modelos de linguagem até estratégias mais elaboradas, como *Mixture-of-Agents*, geração com contexto reduzido e refinamento iterativo. Nesse contexto, o Abacus [Russo et al. 2026] propõe um otimizador baseado em custos para a seleção dos operadores físicos de uma aplicação. Durante a otimização, o sistema amostra implementações físicas e estima, para cada uma, qualidade, custo financeiro e latência. Com base nessas estimativas, identifica os operadores potencialmente pertencentes à fronteira de Pareto e emprega uma extensão do algoritmo Cascades, denominada *Pareto-Cascades* [Agrawal 1995, Wang et al. 2008], para selecionar o plano físico que melhor satisfaz o objetivo e as restrições definidos pelo usuário.

Apesar da eficiência desse processo, o mecanismo de exploração empregado pelo Abacus apresenta duas limitações relevantes: um alto determinismo na amostragem de operadores, comprometendo a exploração do espaço de busca, e a falta de armazenamento do conhecimento adquirido durante a otimização para a reutilizado em execuções posteriores. O sistema modela a seleção de operadores físicos como um problema de *Multi-Armed Bandit* (MAB) [Agrawal 1995, Wang et al. 2008], recorrendo às médias amostrais e limites de confiança para orientar a exploração do espaço de busca. Entretanto, ao final de cada execução, as estimativas obtidas não são incorporadas a uma memória persistente, fazendo com que novas otimizações reiniciem o processo exploratório.

Visando mitigar essas limitações, o presente trabalho propõe uma estratégia de otimização Bayesiana [Shahriari et al. 2016] com aprendizado persistente para sistemas de operadores semânticos, em duas etapas. Primeiro, substituímos a representação baseada em médias e limites de confiança por distribuições de probabilidade atualizadas via Thompson Sampling (BAY) [Russo et al. 2018], modelando explicitamente a incerteza do desempenho dos operadores. Em seguida, introduzimos um mecanismo de armazenamento persistente dessas distribuições, organizado pelo contexto de cada operador (CXT-BAY), permitindo que novas execuções sejam inicializadas com o conhecimento acumulado em execuções anteriores. Experimentalmente, o CXTBAY encontrou o mesmo plano final que o MAB em 85% das execuções e planos de qualidade superior nas demais, com custo de otimização até $2,3\times$ menor e tempo de busca $2,5\times$ menor, caracterizando maior eficiência.

O restante do texto está organizado da seguinte forma. A Seção 2 situa este trabalho em relação aos principais sistemas de operadores semânticos encontrados na literatura, enquanto a Seção 3 detalha o Abacus, otimizador que serve de base à proposta, assim como as limitações de seu mecanismo de amostragem. A partir dessas limitações, a Seção 4 apresenta as estratégias BAY e CXTBAY, cuja avaliação experimental é discutida na Seção 5. Por fim, a Seção 6 aponta as limitações deste estudo e os trabalhos futuros que delas decorrem, além de resumir as principais conclusões da pesquisa.

2. Trabalhos Relacionados

Este trabalho insere-se no contexto de aplicações construídas a partir de operadores semânticos. Esses operadores correspondem a transformações de dados apoiadas por modelos de linguagem de grande escala, cujas tarefas podem ser especificadas em linguagem natural. Diferentemente dos operadores relacionais tradicionais, que atuam predominantemente sobre atributos e estruturas previamente definidos, os operadores semânticos

permitem processar dados não estruturados ou semiestruturados, como documentos, imagens e textos livres. Entre os operadores mais comuns estão *Map*, *Filter*, *Join*, agregações e operações top-*k* [Shankar et al. 2025].

Sistemas como Palimpzest [Liu et al. 2025] e LOTUS [Patel et al. 2025] oferecem abstrações declarativas para a construção de pipelines compostos por esses operadores. Nesse modelo, o usuário descreve um plano lógico que representa as etapas necessárias ao processamento dos dados, sem definir antecipadamente todos os detalhes de execução. Essa abordagem permite expressar aplicações complexas, como a extração de informações sobre reações adversas a medicamentos em artigos biomédicos, representada pelo benchmark BioDEX [Deleate et al. 2023], e a identificação de cláusulas relevantes em contratos jurídicos, avaliada pelo conjunto CUAD [Hendrycks et al. 2021].

A partir do plano lógico, o sistema deve selecionar um plano físico, descrevendo com que parâmetros ou configurações o plano lógico será executado. Um mesmo operador semântico pode ser executado por diferentes LLMs, prompts, servidores, estratégias de redução de contexto ou configurações de inferência. Essas alternativas apresentam compromissos distintos entre qualidade dos resultados, custo monetário e latência. Consequentemente, a escolha das implementações físicas constitui um problema de otimização, especialmente quando o espaço de configurações cresce de forma combinatória com o número de operadores do pipeline.

O LOTUS [Patel et al. 2025], além de formalizar operadores semânticos, propõe implementações aproximadas para alguns desses operadores, explorando modelos *proxy*, *embeddings* e técnicas de amostragem para reduzir o custo de execução e fornecer garantias de qualidade em relação a algoritmos de referência. Sua abordagem, entretanto, concentra-se principalmente em otimizações específicas para cada operador, não tratando de forma geral a seleção conjunta de implementações heterogêneas para todos os operadores de um pipeline.

O Aryn [Anderson et al. 2025] aborda o processamento de dados não estruturados sob uma perspectiva sistêmica mais ampla. O sistema combina componentes de *parsing* de documentos, um mecanismo distribuído de execução e um planejador capaz de traduzir consultas em linguagem natural para planos de processamento. Sua principal contribuição está na geração de planos semânticos explicáveis sobre grandes coleções documentais, abrangendo tanto a preparação dos dados quanto a execução dos operadores.

O DocETL [Shankar et al. 2025], por sua vez, emprega agentes baseados em LLMs para reescrever pipelines complexos de processamento documental. O sistema pode decompor operadores, particionar documentos, introduzir novas etapas de processamento e produzir critérios de avaliação específicos para a tarefa. Em vez de apenas selecionar uma implementação física dentro de um conjunto predefinido, o DocETL explora reescritas mais abertas do pipeline com o objetivo de melhorar a qualidade dos resultados.

Como uma extensão direta do Palimpzest, o Abacus [Russo et al. 2026], discutido em detalhes na Seção 3, formula a seleção de implementações físicas como um problema de otimização multiobjetivo baseada em custos. O sistema considera conjuntamente qualidade, custo monetário e latência, permitindo ainda que uma dessas dimensões seja otimizada sob restrições impostas às demais. Para estimar o desempenho dos operadores físicos, o Abacus utiliza uma estratégia de amostragem baseada em *Multi-Armed Ban-*

dits, que pode incorporar exemplos de validação, crenças prévias sobre o desempenho dos operadores e julgamentos produzidos por LLMs. A busca procura identificar operadores pertencentes à fronteira de Pareto, em vez de selecionar apenas a melhor alternativa segundo um único objetivo. Além disso, o Abacus introduz o algoritmo Pareto-Cascades, que adapta os princípios de otimização do Cascades para cenários com múltiplos objetivos e restrições. Em vez de manter somente um subplano considerado ótimo para cada expressão lógica, o algoritmo preserva conjuntos de subplanos não dominados, permitindo que diferentes compromissos entre qualidade, custo e latência sejam considerados na composição do plano físico final.

O Stretto [Sanmartino et al. 2026] também investiga a geração de planos físicos para sistemas de processamento apoiados por LLMs. Sua abordagem procura identificar, a partir de uma amostra do conjunto de entrada, um plano de menor custo cuja qualidade permaneça comparável à de uma configuração de referência mais cara. Além da seleção de planos, o sistema incorpora técnicas de compressão do *KV cache*, ampliando o espaço de alternativas de execução e considerando otimizações relacionadas à infraestrutura de inferência.

Em síntese, os trabalhos existentes podem ser agrupados em três perspectivas complementares. Palimpzest e LOTUS concentram-se na definição de abstrações declarativas e de operadores semânticos; Aryn e DocETL enfatizam a geração e a reescrita de pipelines; enquanto Abacus aborda a seleção de planos físicos e a otimização de sua execução. Este trabalho parte especificamente da linha introduzida pelo Abacus e propõe uma estratégia Bayesiana com persistência contextual do conhecimento adquirido durante a otimização, buscando reutilizar e atualizar estimativas obtidas em execuções anteriores, em vez de reconstruir o processo de avaliação a cada nova execução. Resta mencionar também que Abacus se mostrou significativamente mais eficiente que Palimpzes, LOTUS e DocETL, sendo então o principal sistema estado da arte [Russo et al. 2026].

3. Abacus: Um Sistema Otimizador de Operadores Semânticos

O Abacus [Russo et al. 2026] representa um avanço ao introduzir um otimizador extensível e baseado em custos para sistemas de operadores semânticos. Como é ilustrado na Figura 1, a partir de um plano lógico, o sistema enumera, para cada operador, um amplo conjunto de implementações físicas candidatas, formado por diferentes combinações de modelos, estratégias de execução e hiperparâmetros, podendo alcançar milhares de alternativas. Como a avaliação exaustiva desse espaço é impraticável, é realizada uma amostragem de operadores físicos que possam compor um plano físico Pareto-ótimo. Este processo é modelado pelo Abacus como um problema de *Multi-Armed Bandit* (MAB) [Agrawal 1995, Wang et al. 2008], no qual cada operador físico corresponde a um braço. Sob um orçamento limitado de amostragem, expresso em número de invocações a LLMs ou em custo financeiro, o otimizador seleciona adaptativamente quais candidatos avaliar, concentrando os recursos disponíveis nos operadores com maior potencial de integrar soluções Pareto-ótimas.

Esse procedimento de amostragem de operadores físicos opera iterativamente sobre uma fronteira de operadores físicos candidatos associada a cada operador lógico. A cada iteração, os operadores ativos são executados sobre um lote de entradas amostradas do conjunto de validação ou, na ausência deste, do próprio conjunto de entrada. Para cada

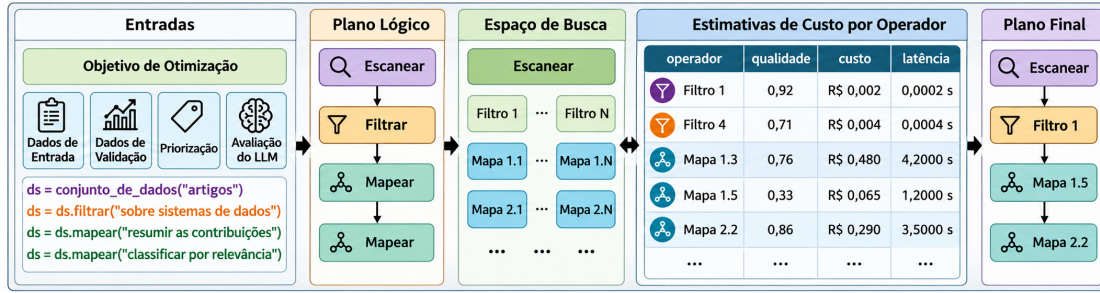


Figura 1. Visão geral do processo de otimização do Abacus. A partir de um plano lógico, o sistema enumera operadores físicos candidatos, estima qualidade, custo e latência por amostragem e seleciona o plano físico que melhor atende ao objetivo de otimização.

métrica relevante ao objetivo de otimização, como qualidade, custo financeiro e latência, o sistema define respectivos limites superior (*Upper Confidence Bound* ou UCB) e inferior de confiança (*Lower Confidence Bound* ou LCB) baseados na média dos operadores executados até o momento e um termo de exploração. Em seguida, é verificado para cada candidato se seus limites superiores de confiança apresentam sobreposição com os limites inferiores de ao menos um operador pertencente à fronteira de Pareto. Essa sobreposição indica que, devido à incerteza das estimativas, o candidato ainda pode ser Pareto-ótimo e, portanto, deve permanecer em avaliação. Quando não há sobreposição, o operador é removido da fronteira e substituído por um novo candidato retirado do conjunto de operadores físicos ainda não amostrados. O termo de incerteza incorporado aos limites de confiança favorece a exploração, pois operadores com poucas observações apresentam intervalos mais amplos e permanecem por mais tempo como possíveis integrantes da fronteira de Pareto. O processo é repetido até que o orçamento de amostragem, expresso em número de invocações ou em custo monetário, seja atingido.

Apesar de eficaz, essa formulação apresenta duas limitações que motivam este trabalho. A primeira diz respeito ao equilíbrio entre exploração e exploração. O mecanismo de UCB do Abacus explora o espaço de operadores de forma conservadora: mesmo sob restrições de custo mais relaxadas, o otimizador não converte a folga disponível em exploração mais ampla, tendendo considerar os operadores mais baratos ainda que operadores mais custosos pudessem resultar em qualidade superior. A segunda é ausência de persistência de conhecimento entre execuções. O MAB tradicional assume que a amostragem recomeça do zero a cada consulta, de modo que todas as estimativas construídas durante a otimização são descartadas ao término do *pipeline*. Consequentemente, cada nova execução reinicia a exploração sem memória do que foi aprendido.

4. Otimização Bayesiana com Amostragem de Thompson e Contexto

Esta seção apresenta as abordagens de otimização propostas para superar limitações do mecanismo de *Multi-Armed Bandit* (MAB) empregado pelo Abacus. Nossa solução é motivada pela substituição dos intervalos de confiança do MAB por um sistema com amostragem de Thompson. A superioridade do método de Thompson comparado a UCB do MAB já foi mostrada na literatura, mas não aplicada a este domínio [Zhong et al. 2021].

A ideia principal deste método é o uso de distribuições posteriores, atualizadas com as amostras obtidas no processo de otimização, que representem a crença corrente

sobre o desempenho de cada operador físico em qualidade, custo e tempo de execução — isso para cada operador lógico. Dessas distribuições deriva-se, por simulação, a probabilidade de cada operador pertencer à frente de Pareto.

O algoritmo base, descrito em Algoritmo 1, atualiza iterativamente um conjunto de *posteriors* (Π) dados os operadores físicos amostrados e executados enquanto há orçamento de amostragem. Esse orçamento pode ser definido como o número de invocações LLM que podem ser geradas no curso da otimização. A cada iteração é feita uma amostragem dos operadores físicos mais promissores, ou seja, com maior chance de comporem planos da frente de Pareto (linha 2). Esses operadores são executados (linha 3) e tem suas métricas de custo e qualidade adicionadas ao registro corrente (linha 4). Com novas informações de execuções reais, é feita uma estimativa das variáveis de objetivo para planos físicos simulados (linha 5), sendo esses dados usados para produzir uma nova frente de Pareto de planos físicos. Essa frente de Pareto pode ser usada diretamente para a geração do plano físico final pelo algoritmo *Pareto-Cascades* [Russo et al. 2026] (linha 9), ou pode ser usado para atualizar os *posteriors* dos operadores físicos (linha 6), reduzindo a incerteza de estimativas para novas iterações de otimização. Os detalhes técnicos de dada operação são discutidos nos parágrafos a seguir.

Algoritmo 1 Estratégia Bayesiana de Otimização

Entrada: *posteriors* Π , métricas médias de planos físicos μ

Parâmetros: Monte-Carlo samples S

```

1: enquanto há orçamento de otimização faça
2:   operadores_fisicos = thompsonSampling( $\Pi$ )
3:   métricas = execução(operadores_fisicos)
4:   atualizaMédias( $\mu$ , métricas)
5:   sim = monteCarlo( $\mu$ ,  $S$ )
6:   frente_pareto = pareto(sim)
7:   atualizaPosterior( $\Pi$ , frente_pareto)
8: fim enquanto
9: return ParetoCascades(frente_pareto)

```

Para buscar frentes de Pareto melhores, assim como aumentar a confiança nas estimativas de custo, operadores físicos são amostrados para serem testados. Aqui é feita uma amostragem de Thompson [William et al. 1933] a partir das *posteriors* que descrevem o desempenho esperado de cada operador físico em cada objetivo (linha 2). Vale ressaltar que cada objetivo tem seu próprio *posterior*, sendo Π_q , Π_c e Π_t os *posteriors* de custo, tempo de execução e qualidade, respectivamente, para o conjunto de operadores físicos. A amostragem multiobjetivo prescinde de qualquer escalarização dos objetivos. Em vez de combinar Π_q , Π_c e Π_t em uma distribuição única, sorteia-se de cada uma independentemente, de modo que cada operador físico recebe um vetor $(\hat{o}_{qi}, \hat{o}_{ci}, \hat{o}_{ti})$ — uma realização plausível de seu desempenho, coerente com a crença corrente.

Após a execução dos operadores amostrados e atualização de métricas (linhas 3-4), e feito uma simulação de planos físicos a fim de gerar uma nova frente de Pareto mais acurada. Neste processo, é feita uma amostragem aleatória de operadores físicos, que são combinados para gerar S planos (linha 5). A estimação das variáveis de objetivo para os planos simulados é feita da mesma forma que pelo Abacus [Russo et al. 2026], descrita

pelas equações:

$$\hat{p}_q = \prod_{i=1}^M \hat{o}_{qi} \quad \hat{p}_c = \sum_{i=1}^M \hat{o}_{ci} \quad \hat{p}_t = \max_{\text{path} \in p} \sum_{i \in \text{path}} \hat{o}_{ti} \quad (1)$$

Nas equações acima temos que o *posterior* de qualidade $\hat{p}_q$ para um plano físico se baseia na métrica de qualidade $\hat{o}_{qi}$ de cada i -ésimo operador físico do plano. O mesmo vale para as demais variáveis de objetivo de custo ($\hat{p}_c$) e tempo de execução $\hat{p}_t$.

Após a geração dos planos simulados, esses são comparados por dominância de Pareto. Nota-se que é mantida uma frente de Pareto sobre todas as variáveis objetivo, e não por um escore agregado. A opção por não escalarizar é deliberada: qualquer combinação da forma $w_q \hat{o}_{qi} + w_c \hat{o}_{ci} + w_t \hat{o}_{ti}$ exigiria o treinamento de um conjunto de pesos w . Como o objetivo do Abacus é recuperar a frente inteira — e não um ponto dela —, os objetivos permanecem separados durante toda a amostragem, e sua conciliação é postergada para a seleção do plano final, a cargo do algoritmo *Pareto-Cascades* [Russo et al. 2026].

Para o cálculo do *posterior* são usadas as distribuições Beta ($B(\alpha, \beta)$) para a qualidade (pois está limitada no intervalo $[0, 1]$), e Gaussiana ($\mathcal{N}(\mu, \sigma^2)$) para custo e tempo de execução (pois não estão limitados a $[0, 1]$) [Zhong et al. 2021]. Cada operador físico é identificado por um conjunto de parâmetros de suas distribuições, sendo estes parâmetros atualizados após cada iteração de execuções (linhas 3-4).

Adicionando Contexto à Otimização Bayesiana. Um desafio do processo de otimização de planos está no uso de *priors*. Este conhecimento é limitado ao se trabalhar com conjuntos diferentes de possíveis operadores físicos. Ou seja, mesmo que haja um histórico de execução para um conjunto de operadores (e.g., modelos A e B de LLM), esse conhecimento não pode ser reaproveitado para novos conjuntos de operadores (e.g., incluindo agora os modelos C e D). Para este fim, propomos a criação de uma base de distribuições por operador físico que pode ser usada como *prior* para qualquer operação de otimização. Caso os operadores utilizados em um processo de otimização já houvessem sido executados, o *prior* exato é retornado. Caso seja um operador novo, é retornado um *prior* baseado no operador de maior similaridade já presente.

Para implementar essa base de *priors*, cada operador físico é descrito por meio de um *embedding* contendo (i) uma codificação *one-hot* do tipo do operador, com uma posição por tipo e o valor unitário na posição correspondente, (ii) uma codificação *multi-hot* dos modelos utilizados, com valor unitário nas posições dos modelos presentes, e (iii) um conjunto de características numéricas normalizadas, como a temperatura média, o número de propositores e os parâmetros de recuperação por contexto. Para cada chave, representada pelo *embedding*, são persistidos os parâmetros das distribuições Beta e Gaussiana.

Ao se deparar com um operador nunca testado, é usado um processo gaussiano para estimar os *priors* de cada variável objetivo a partir dos operadores mais similares. A similaridade entre *embeddings* foi calculada por meio do *Radial Basis Function kernel*, definido como $k(e_1, e_2) = \exp(-\gamma \|e_1 - e_2\|^2)$. Para a distribuição Beta, os valores (μ, σ) do processo gaussiano são convertidos pelo método dos momentos [Berk 1996], enquanto para a distribuição gaussiana isso não é necessário. Por meio das soluções descritas, é

possível a criação e manutenção de uma base de *priors* que é usada como base de busca para operadores já testados, e base de consulta para estimação de *priors* para operadores nunca testados, com base nos conhecimentos adquiridos de outras execuções.

5. Avaliação Experimental

Os experimentos foram conduzidos em um nó HGX com 8 GPUs NVIDIA H200, dois processadores AMD EPYC 9534 (128 cores) e 2,2 TB de RAM, executando Rocky Linux 9.7 e driver NVIDIA 550.163. As tarefas foram executadas em Kubernetes v1.28.15, com um *pod* por GPU. Quatro modelos foram servidos localmente via vLLM [Kwon et al. 2023]: Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct, Qwen3-235B-A22B e gpt-oss-120b. O espaço de operadores físicos foi gerado pelas regras nativas do Abacus, incluindo seleção de modelo, *Mixture-of-Agents*, geração com contexto reduzido e *Critique-and-Refine* [Russo et al. 2026].

A avaliação utilizou o benchmark CUAD [Hendrycks et al. 2021], composto por contratos jurídicos nos quais devem ser identificados os trechos associados a 41 categorias de cláusulas. O plano lógico contém um operador semântico de mapeamento, e sua qualidade é avaliada pelo *F1-score* das predições. Os parâmetros do Abacus foram mantidos nos valores padrão, $k = 6$ e $j = 4$, correspondendo, respectivamente, ao número de operadores avaliados simultaneamente em cada iteração e ao número de contratos inicialmente fornecidos a cada novo operador. Nesse ambiente, foram comparadas as estratégias MAB, proposta originalmente no Abacus, e CXTBAY, correspondente à abordagem Bayesiana com contexto. O CXTBAY pode ser usado com diferentes quantidades de conhecimento persistido. Todo o código usado na preparação dos dados e execução dos experimentos a seguir se encontram em repositório público ¹.

5.1. Parâmetros Experimentais

Cada parâmetro variado atua sobre uma etapa específica da otimização. Em todos os experimentos, foi utilizada a política `MaxQuality@FixedCost`, cujo objetivo é maximizar a qualidade do plano, medida pelo *F1-score*, respeitando um limite fixo para seu custo de execução. Foram considerados cinco limites de custo, com $\mathcal{C} = \{0,1, 0,3, 0,5, 0,7, 0,9\}$ \$. Como os modelos são servidos localmente, esses custos são simulados a partir dos preços por *token* definidos nas *model cards* do código-fonte, baseados em valores de serviços comerciais de inferência. Portanto, representam uma medida relativa de consumo entre operadores, e não um desembolso efetivo.

Também foi variado o orçamento máximo de amostragem, que limita o número de invocações aos LLMs durante a avaliação dos operadores e, conseqüentemente, controla o custo e o tempo da fase de otimização. Orçamentos maiores permitem acumular mais evidências sobre os candidatos, ao custo de um maior número de chamadas e de *tokens* processados. Esse parâmetro representa, assim, o esforço empregado na *descoberta* dos operadores.

5.2. Análise do Efeito do Aprendizado Persistente

Para validar a premissa de que a preservação do histórico de avaliações acelera o processo de otimização, investigamos o impacto do acúmulo de memória no comportamento

¹<https://github.com/ArthurVdoc/palimpzest-CXTBAY>

do algoritmo CXTBAY. O experimento consistiu em simular a persistência de conhecimento entre submissões de workloads, avaliando o desempenho do sistema inicializado com diferentes tamanhos de contexto histórico. O processo de treinamento do modelo Bayesiano contemplou a realização de até 15 execuções prévias (denominadas Execs.). Cada Exec. representa um ciclo completo de otimização restrito a um orçamento máximo de 50 amostras. Após essa fase de consolidação dos *priors* e atualização das distribuições *a posteriori*, procedeu-se à medição de uma nova otimização de plano de execução.

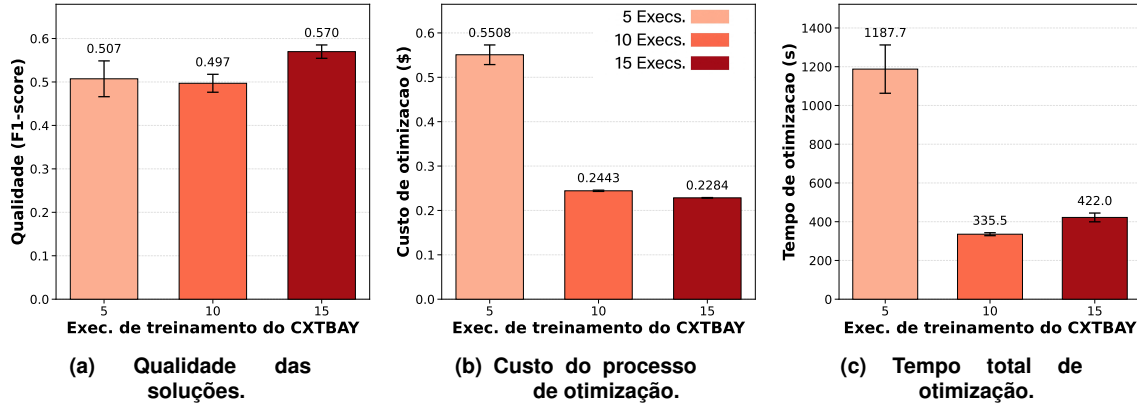


Figura 2. Resultados do desempenho do CXTBAY com diferentes contextos iniciais. Contextos representam 5, 10, e 15, processos de otimização realizados antes da avaliação (Execs.). Valores médios reportados com intervalo de confiança de 95%.

Os resultados, consolidados na Figura 2, demonstram a correlação direta entre a maturidade do contexto e a eficiência computacional do sistema. A transição de um contexto incipiente (5 Execs.) para uma base de conhecimento maior (10 Execs.) resulta em uma mitigação drástica do esforço exploratório cego. Especificamente, observou-se uma redução de até $2,30\times$ no custo financeiro da otimização e um decréscimo expressivo no tempo total, que se revelou pelo menos $2,81\times$ menor em comparação à configuração com 5 execuções. Esse fenômeno atesta que o algoritmo internaliza rapidamente as características latentes do espaço de busca, evitando alocar o orçamento de amostragem em implementações físicas cujo desempenho empírico passado já as desqualificou para a fronteira de Pareto.

Sob a perspectiva da eficácia da tarefa (medida pelo F1-score de extração), a expansão do contexto de 10 para 15 execuções proporcionou ganhos estritamente marginais. Esse platô de convergência indica que a principal contribuição do aprendizado persistente não reside em descobrir arranjos utópicos invisíveis a buscas exaustivas, mas sim em alcançar a máxima eficiência de Pareto com uma fração irrisória do custo temporal e financeiro. Considerando que a marca de 10 execuções prévias apresentou um ponto interessante no que se refere ao compromisso entre qualidade e tempo, esta configuração foi utilizada no restante dos experimentos.

5.3. Comparação do CXTBAY com MAB

Nesta seção, estabelecemos uma comparação empírica direta entre a estratégia de ajuste dinâmico proposta (CXTBAY) e o método padrão de exploração baseado em Multi-

Armed Bandits (MAB) com poda por limites de confiança, adotado originalmente pelo otimizador do sistema Abacus.

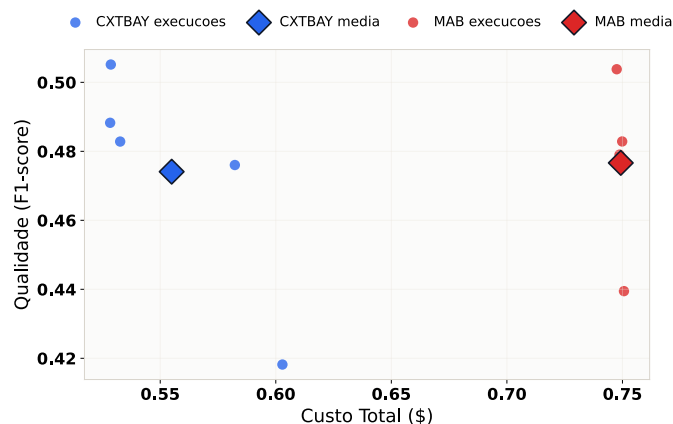


Figura 3. Comparação do CXTBAY e do MAB para orçamento de amostragem de 40 requisições. Além das execuções individuais, é mostrado o ponto médio de todas as execuções.

Em termos de eficácia na identificação de pipelines robustos, ambas as abordagens demonstraram capacidade de convergir para configurações de alta qualidade, como mostra a Figura 3. O modelo CXTBAY obteve um F1-score mediano de 0,484, apresentando um ganho marginal, porém sistemático, em relação à mediana de 0,479 alcançada pelo baseline MAB. Este resultado evidencia que a substituição de uma poda heurística baseada em intervalos de confiança (UCB), por uma estimativa de Monte Carlo guiada por amostragem de Thompson, não degrada a acurácia do sistema. Ao contrário, a modelagem probabilística fornece uma exploração mais direcionada, garantindo a descoberta de arranjos ótimos no espaço de busca sem comprometer o rigor da extração de dados.

Os resultados da Figura 3 também evidenciam que o custo total do CXTBAY é $2,3\times$ menor para os resultados de melhor qualidade. Também vemos que o CXTBAY continua a explorar soluções em um espaço de busca de custo maior que o MAB. Isso se deve a um melhor conhecimento do espaço de busca dado o seu contexto acumulado. Além da eficiência de custo, o CXTBAY proporcionou uma aceleração de $2,5\times$ no tempo de busca necessário para definir o plano de execução. Em ambientes de alto desempenho, onde a latência introduzida pelo otimizador deve ser estritamente controlada para não anular os ganhos da aceleração do pipeline, essa redução consolida a viabilidade do método.

Ao analisar os planos gerados por cada algoritmo, foi visto que o comportamento do CXTBAY não se limitou apenas a reduzir o tempo/custo necessário para encontrar a região conservadora do MAB. Em 85% das execuções, o plano final replica a configuração de *Critique-and-Refine* que o MAB seleciona. Nas demais, o CXTBAY, aproveitando a folga de limite de custo de execução, gera uma configuração de *Mixture-of-Agents* que entregou uma qualidade de 0.5648, que representa o melhor resultado do conjunto de experimentos. A diversidade de seleção sintetiza o percurso: o MAB emprega uma única configuração física de operador semântico para o plano lógico em suas quarenta execuções, já o CXTBAY anterior ao treinamento dispersa-se por seis, a fase de treino percorre dez sem repetição, e a estratégia após as 10 execuções de treinamento contrai-se

para duas. A memória não apenas registra o que foi explorado, mas concentra a busca subsequente naquilo que se mostrou proveitoso, preservando a capacidade de se afastar da região estável quando isso é favorável para a otimização.

5.4. Análise do Orçamento de Amostragem

A viabilidade de otimizadores baseados em custo para sistemas de operadores semânticos depende intrinsecamente de quão bem o algoritmo gerencia os recursos computacionais disponíveis. Assim, foi analisado o efeito do orçamento de amostragem que atua como um teto para o número de avaliações empíricas permitidas (invocações aos LLMs) nos resultados globais do processo de otimização. A Figura 4 ilustra o comportamento comparativo entre o *baseline* MAB e o CXTBAY treinado sob diferentes limites de amostragem.

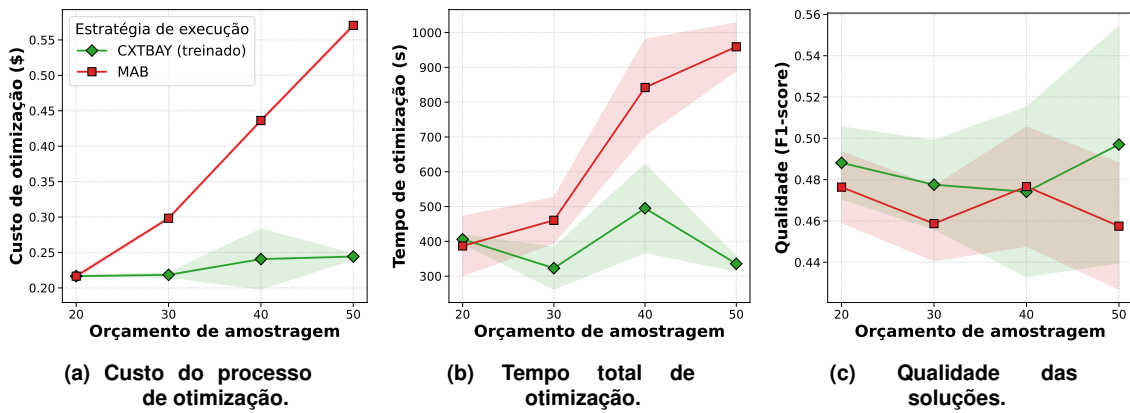


Figura 4. Avaliação do CXTBAY e MAB variando o orçamento de amostragem. Faixas indicam o intervalo de confiança de 95% para as execuções com os 5 custos fixos.

Observa-se que o MAB possui uma sensibilidade do custo de otimização próxima do linear em relação ao orçamento disponível. Ao ser exposto a orçamentos menos restritivos (como 40 ou 50 amostras), o método faz um uso mais indiscriminado dos recursos disponíveis, como visto na Figura 4a. Como o MAB redefine suas estimativas de desempenho a cada nova execução e constrói seus intervalos de confiança (*confidence bounds*) do zero, um orçamento maior prolonga uma fase de exploração cega. Isso resulta em custos financeiros e tempos totais de otimização significativamente inflados. Em contrapartida, o CXTBAY exibe um comportamento sublinear, devido à inicialização do seu contexto com um mapa probabilístico do espaço de busca. Consequentemente, ele consegue aplicar melhor os recursos disponíveis, interrompendo a amostragem de operadores manifestamente sub-ótimos logo nas primeiras iterações e contendo o crescimento do custo e do tempo de otimização.

A discrepância no uso de recursos torna-se ainda mais crítica ao analisarmos a métrica de sucesso da tarefa. O gasto excessivo do MAB nem sempre se reverte em ganhos de qualidade no plano físico gerado. A Figura 4c demonstra que, em orçamentos elevados, o F1-score alcançado pelo MAB sofre grande variância ou estagna, indicando que o algoritmo gasta recursos avaliando variações de operadores que pouco contribuem para o aprimoramento da fronteira de Pareto. Alternativamente, o CXTBAY, guiado pelo *Thompson Sampling*, não apenas atinge resultados de qualidade equivalentes ou superiores ao MAB, mas também utiliza a margem extra de orçamento de forma mais eficaz.

6. Conclusões e direções futuras

Este trabalho apresentou duas contribuições sobre o mecanismo de amostragem de operadores do Abacus: (i) uma reformulação Bayesiana que substitui a poda por intervalos de confiança por uma estimativa da probabilidade de pertencimento à fronteira de Pareto, (ii) a persistência das distribuições *a posteriori* entre execuções e, para operadores nunca amostrados, interpolação de *priors* informativas por meio de um Processo Gaussiano sobre *embeddings* de operadores. A avaliação sobre o *benchmark* CUAD mostrou que a formulação Bayesiana amplia substancialmente a exploração do espaço, mas que essa exploração, sem conhecimento que a oriente, converte-se em dispersão de resultado cerca de $3,2\times$ maior. O aprendizado persistente reconcilia as duas dimensões: após o treino, a dispersão do *F1-score* comprime-se por fator próximo de três, situando-se abaixo da do próprio MAB, com qualidade equivalente e custo de otimização $2,3\times$ menor sob o maior orçamento de amostragem.

Embora os resultados experimentais tenham sido promissores, vemos algumas avenidas de expansão do mesmo. Primeiramente, as soluções propostas foram avaliadas apenas com o *benchmark* CUAD. Embora ele seja um *benchmark* significativo, sendo usado também no trabalho original do Abacus [Russo et al. 2026], será interessante observar os ganhos em outros *benchmarks*, como o BioDEX. A extensão do conjunto de *benchmarks* também permitirá avaliar os possíveis ganhos quando observados outros tipos de operadores, como *Filter*, *Join* e agregações. Estima-se que os perfis de custo e qualidade de outros operadores podem diferir substancialmente dos avaliados neste trabalho. Adicionalmente, também será interessante analisar as implicações do uso de diferentes plataformas de hardware, não somente do ponto de vista de aumentar ou reduzir a capacidade de processamento, mas também analisando o comportamento da busca quando as relações de qualidade e custo mudam. Dessa forma, a avaliação a outros *benchmarks*, assim como a extensão das plataformas de hardware analisadas constitui a continuidade natural deste trabalho.

Além dessa extensão, planeja-se expandir a avaliação do CXTBAY com o aprimoramento da modelagem do espaço de busca através da investigação do uso de representações semânticas latentes para as tarefas como alternativa das codificações *one-hot* e *multi-hot* para criar os *embeddings* dos operadores físicos e modelos. Além disso, a incorporação de métricas dinâmicas de infraestrutura diretamente no modelo Bayesiano, tais como as otimizações de compressão do KV cache, poderia refinar ainda mais o balanço entre custo, latência e qualidade, consolidando o ajuste automático dinâmico para ambientes de alto desempenho contínuo.

Agradecimentos. Este trabalho foi parcialmente financiado por CNPq, CAPES, FAPEMIG e Instituto Kunumi. Os autores agradecem à instituição pelo apoio financeiro e pelo compromisso com o avanço da pesquisa científica.

Referências

Agrawal, R. (1995). The Continuum-Armed Bandit Problem. *SIAM J. Control Optim.*, 33(6):1926–1951.

- Anderson, E., Fritz, J., Lee, A., Li, B., Lindblad, M., Lindeman, H., Meyer, A., Parmar, P., Ranade, T., Shah, M. A., Sowell, B., Tecuci, D., Thapliyal, V., and Welsh, M. (2025). The design of an LLM-powered unstructured analytics system. In *Proceedings of the Conference on Innovative Data Systems Research*, CIDR.
- Berk, R. H. (1996). Continuous univariate distributions, volume 2.
- Deleate, K., Van der Straeten, S., Luyckx, K., and Demeester, T. (2023). BioDEX: Large-scale biomedical adverse drug event extraction for real-world pharmacovigilance. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13429–13446, Singapore. Association for Computational Linguistics.
- Hendrycks, D., Burns, C., Chen, A., and Ball, S. (2021). CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 28479–28492.
- Kwon, W. et al. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGCOMM Conference*.
- Liu, C. et al. (2025). Palimpsest: Declarative data processing for ai pipelines. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.
- Patel, P. et al. (2025). Lotus: Enabling semantic queries with large language models over structured and unstructured data. *Proceedings of the VLDB Endowment (PVLDB)*.
- Russo, D., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. (2018). A tutorial on thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96.
- Russo, M. et al. (2026). Abacus: A cost-based optimizer for semantic operator systems. *Proceedings of the VLDB Endowment (PVLDB)*.
- Sanmartino, G., Urban, M., Papotti, P., and Binnig, C. (2026). The stretto execution engine for llm-augmented data systems. *arXiv preprint arXiv:2602.04430*.
- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.
- Shankar, S., Chambers, T., Shah, T., Parameswaran, A. G., and Wu, E. (2025). DocETL: Agentic query rewriting and evaluation for complex document processing. *Proceedings of the VLDB Endowment*, 18(9):3035–3048.
- Thakur, N., Reimers, N., Ruckl’e, A., Srivastava, A., and Gurevych, I. (2021). BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *ArXiv*, abs/2104.08663.
- Wang, Y., Audibert, J.-y., and Munos, R. (2008). Algorithms for Infinitely Many-Armed Bandits. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.
- William, R. T. et al. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.
- Zhong, Z., Chueng, W. C., and Tan, V. Y. (2021). Thompson sampling algorithms for cascading bandits. *Journal of Machine Learning Research*, 22(218):1–66.