Expert-Guided Bayesian Optimization for Sustainable Protein Design

Anna Thomas[1,2]* **Georgios Zaverdinos**[3] **Petros Mandalis**[3] **Andreas Orfanoudakis**[3]
Akihiro Takino[3] **Sohum Patnaik**[2] **David J. Kraus**[3] **Panos Kostopoulos**[3]
Caroline Cotto[4] **Nikos Tsiaparas**[3] **Dan Jurafsky**[1] **Aadit Patel**[5]

[1]Stanford Computer Science [2]Food Intelligence Lab [3]Proxy Foods AI
[4]NECTAR [5]Bayes & Gambit

Abstract

Agentic laboratories give LLM agents control of experiments, alongside scientific tools such as optimizers and code execution. How should such systems choose the next experiment? We study whether an LLM agent should propose experiments directly, delegate them to Bayesian optimization (BO), or select the design space itself and delegate refinement. We formalize the third option as Expert-Guided Bayesian Optimization (EGBO), in which an expert (e.g., an LLM or a human) selects a low-dimensional subspace for BO and may adaptively expand it over time. We decompose EGBO's suboptimality into a selection gap and an optimization gap, and characterize the coverage–dimension tradeoff governing when expert guidance helps. To support *in silico* prototyping before costly laboratory work, we introduce FORMULATEBENCH, a suite of 24 sustainable protein formulation tasks. In batched experiments that mirror our laboratory deployment, EGBO outperforms conventional BO. The strongest LLMs match EGBO without noise but underperform it under noise, even when told the noise level. When deployed to optimize two plant-based dairy products, EGBO improves utility, as assessed by a trained human panel, by 29% and 26% in 10 batched iterations each. In a comparison with a professional food scientist given the same time budget, EGBO reached a best observed utility of 0.992, vs. 0.850 for the food scientist.

1 Introduction

Agentic laboratories, in which LLM agents control experiments alongside scientific tools such as optimizers and code execution, are increasingly used to accelerate scientific discovery [Abolhasani and Kumacheva, 2023, Roohani et al., 2025, Fu et al., 2026]. A central question for such systems is how to choose the next experiment. One option is for the LLM to propose experiments directly. LLMs bring broad prior knowledge, but direct comparisons find them competitive mainly at small budgets [Zhang et al., 2023, Liu et al., 2024] and insufficiently sensitive to experimental feedback [Gupta et al., 2025]. Another option is to delegate to Bayesian optimization (BO) [Snoek et al., 2012, Frazier, 2018], which uses feedback efficiently but lacks domain knowledge, and whose sample efficiency deteriorates sharply as the number of variables grows [Srinivas et al., 2012, Bull, 2011]. Existing hybrids let the LLM propose candidates, restart a search that has plateaued, or reshape BO's search space [Chang et al., 2025, Cissé et al., 2025, Han et al., 2025, Brunzema et al., 2026], but which division of labor works best, and under what conditions, has not been systematically characterized.

We study this question in the challenging testbed of sustainable protein formulation. Animal agriculture accounts for roughly 16.5% of global greenhouse gas emissions and is a leading driver of

*thomasat@cs.stanford.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

pollution, biodiversity loss, and pandemic risk [Gerber et al., 2013, Twine, 2021, Espinosa et al., 2020]. Sustainable proteins, such as plant-based, fermentation-enabled, and cultivated products, can substantially reduce these impacts [Poore and Nemecek, 2018, Good Food Institute, 2023, Henchion et al., 2017]. Adoption, however, depends on matching the sensory and nutritional properties of animal products [Onwezen et al., 2021, Ulhas et al., 2023]. Designing such a formulation is expensive trial and error, because each candidate must be physically prepared and evaluated by a human sensory panel [Beckley et al., 2008, Tomsov, 2023]. As in much of experimental science, candidates are prepared and evaluated in batches, and the measurements are noisy. The design space is also large. Candidate ingredients and processing steps can number in the thousands, although a good formulation typically uses only a small subset of them.

We formalize a division of labor between the LLM and BO as *Expert-Guided Bayesian Optimization* (EGBO). An expert selects which variables to optimize over, and BO determines their values within the resulting low-dimensional subspace. The expert can be an LLM agent or a human scientist, so EGBO also formalizes common practice in ML-assisted discovery, where an expert narrows the design space before search begins [Lao et al., 2024, Roohani et al., 2025]. This practice is double-edged. When the expert is right, sample efficiency can improve dramatically, but when the expert is wrong, the optimizer can underperform random search. Because EGBO concentrates the expert's contribution into one interpretable choice, expert reliability reduces to two parameters, recall and false-positive rate, which lets us bound performance as a function of expert quality.

Our contributions include:

1. **A formal division of labor between expert and optimizer.** We formalize EGBO, in which an expert selects a subspace and BO optimizes within it, optionally expanding the subspace over time. We decompose EGBO's error into a selection gap, the expert's responsibility, and an optimization gap, BO's responsibility, and we measure both throughout the paper. On a controlled synthetic benchmark (Hartmann6 embedded in 200 dimensions), expert guidance helps only when the expert's recall is high and false positives are rare, and a weak expert can do worse than random search.

2. **FORMULATEBENCH and when to delegate to an optimizer.** To support evaluating formulation methods before costly laboratory work, we introduce FORMULATEBENCH, 24 plant-based formulation tasks (14 meat, 10 dairy) grounded in real nutritional targets. We implement the three strategies (direct iterative LLM optimization, conventional BO, and EGBO) as fixed pipelines. For every LLM tested, most of the error in its first recipe comes from the concentrations rather than the choice of ingredients, which is precisely the part that BO refines in EGBO. In our target setting of batched, noisy experiments, EGBO outperforms conventional BO and outperforms direct LLM optimization with all three LLMs tested, significantly for two. Without noise, the strongest LLMs match EGBO. Given code execution, none of three LLM agents ran a BO loop, even when told that BoTorch was installed, and most runs fit no model of their data at all.

3. **Real-world deployment and human comparison.** We deploy LLM-guided EGBO with a food company to optimize two plant-based dairy products, improving sensory utility, as assessed by a trained human panel, by 29% and 26% in 10 iterations. In a controlled comparison, EGBO outperformed a professional food scientist by 17%.

2 Related Work

LLMs, BO, and agentic laboratories. LLMs have been used as optimizers in their own right [Yang et al., 2024, Zhang et al., 2023] and as components of BO, for example to warm-start the search or act as the surrogate [Liu et al., 2024]. In experimental design, LLM agents can be insensitive to experimental feedback and outperformed by classical methods [Gupta et al., 2025]. Hybrid systems divide the work in different ways, from an LLM that intervenes when BO stagnates [Cissé et al., 2025] or partitions the search space [Han et al., 2025] to an agent that steers a BO backend [Brunzema et al., 2026]. Closest to EGBO, GPTuner uses an LLM to select database configuration knobs before optimization [Lao et al., 2024]. We instead compare direct LLM optimization, BO, and expert-guided BO on the same tasks, characterize when each is preferable, and deploy the result in a laboratory.

High-dimensional BO. BO has been scaled to high dimensions with random embeddings (REMBO [Wang et al., 2016]), trust regions (TuRBO [Eriksson et al., 2019]), and sparsity-inducing priors or penalties (SAASBO [Eriksson and Jankowiak, 2021], SEBO [Liu et al., 2023]). These methods must infer the relevant subspace from evaluations, consuming budget that EGBO saves by obtaining the initial subspace from the expert. We discuss additional related work in Appendix B.

3 Expert-Guided Bayesian Optimization

Though directly motivated by sustainable protein formulation, EGBO applies to any setting in which an expensive black-box function $f : \mathcal{X} \to \mathbb{R}$ must be maximized over a high-dimensional domain $\mathcal{X} = \prod_{i=1}^{N}[0, u_i^{\max}] \subseteq \mathbb{R}^N$, and where high-quality solutions are expected to depend on only a small subset of the N variables. In the sustainable protein setting, f could be the panel-assessed sensory score of a plant-based formulation parameterized by ingredients and processing steps.

EGBO maintains an active set of variables $S_r \subseteq [N]$ at each round r, and optimizes over the restricted domain $\mathcal{X}_{S_r} = \{x \in \mathcal{X} : x_i = 0 \text{ for all } i \notin S_r\}$. The algorithm has Q rounds. In the first round, the expert proposes an initial active set S_1. In later rounds, the expert may inspect the optimization history and add more variables. The active set grows monotonically: $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_Q$. $Q = 1$ and $Q > 1$ correspond to one-shot and adaptive EGBO, respectively. Let T be the total evaluation budget, including an initial-design budget of n_{init}. We allocate the remaining $T - n_{\text{init}}$ evaluations across the Q BO rounds as $T_1, \ldots, T_Q$, with $n_{\text{init}} + \sum_{r=1}^{Q} T_r = T$. The initial design may include an expert-proposed point, as in FORMULATEBENCH, or may be generated using a standard space-filling design such as Sobol sampling. Earlier evaluations remain valid when the active set expands because $\mathcal{X}_{S_r} \subseteq \mathcal{X}_{S_{r+1}}$. Algorithm 1 describes the procedure.

Algorithm 1 Expert-Guided Bayesian Optimization (EGBO)

Require: Domain $\mathcal{X}$, expert $\mathcal{E}$, total budget T, expert rounds Q, initial design budget n_{init}, round budgets $T_1, \ldots, T_Q$ satisfying $n_{\text{init}} + \sum_{r=1}^{Q} T_r = T$

```
Initialize dataset 𝒟 ← ∅
Query expert ℰ for initial active set S_1 ⊆ [N]
Construct and evaluate n_init initial points in 𝒳_{S_1}, and add them to 𝒟
Fit the Gaussian process surrogate using 𝒟
for r = 1, ..., Q do
    for t = 1, ..., T_r do
        Choose x ∈ 𝒳_{S_r} using the BO acquisition function
        Evaluate y = f(x) + ε
        Add (x, y) to 𝒟 and update the Gaussian process surrogate
    end for
    if r < Q then
        Query expert ℰ with the current history 𝒟 for additional variables A_{r+1} ⊆ [N] \ S_r
        Set S_{r+1} ← S_r ∪ A_{r+1}
    end if
end for
return x̂_T ∈ arg max_{(x,y)∈𝒟} y
```

3.1 Theoretical Analysis

To provide a simple analysis of the performance of Algorithm 1, we make three assumptions. A central feature of sustainable protein formulation problems is that high-quality sparse solutions are rarely unique. A target sensory profile may be achievable via multiple disjoint ingredient sets.

Assumption 1 (Family of sparse approximate optima)**.** *There exist $k \in \mathbb{N}$ and $\eta \geq 0$ such that the family*

$$\begin{aligned}\mathcal{R}_\eta = \big\{ R \subseteq [N] : |R| \leq k, \\ \exists\, \tilde{x}^{(R)} \in \mathcal{X} \textit{ with } \operatorname{supp}(\tilde{x}^{(R)}) \subseteq R \\ \textit{and } f(\tilde{x}^{(R)}) \geq f(x^*) - \eta \big\}\end{aligned} \tag{1}$$

is non-empty. We refer to $\tilde{x}^{(R)}$ as a witness *for support R, a specific k-sparse near-optimum supported on R. We call η the* sparsity gap*, k the* target sparsity level*, and $M = |\mathcal{R}_\eta|$ the* multiplicity*. The* support union $\mathcal{U}_\eta = \bigcup_{R \in \mathcal{R}_\eta} R$ *collects all variables that appear in some sparse near-optimum, with $\kappa = |\mathcal{U}_\eta|$.*

Assumption 1 is strictly weaker than assuming a unique sparse global optimum, since it requires only that *some* k-sparse point achieves near-optimal value.

We characterize expert quality relative to $\mathcal{U}_\eta$:

Assumption 2 (Expert quality)**.** *At each round q, the expert's additions to S_{q+1} are independent across coordinates conditional on S_q, with fresh draws each round. Every variable $i \in \mathcal{U}_\eta$ not yet in S_q is added with probability $p \in (0, 1]$ (*recall*); every $j \notin \mathcal{U}_\eta$ not yet in S_q is added with probability $q_{\mathrm{fp}} \in [0, 1)$ (*false-positive rate*). In the one-shot setting ($Q = 1$), each $i \in \mathcal{U}_\eta$ is included in S with probability p and each $j \notin \mathcal{U}_\eta$ with probability q_{fp}.*

We additionally require a guarantee from the BO subroutine used within each restricted domain.

Assumption 3 (BO subroutine guarantee)**.** *For any active set $S \subseteq [N]$, when the BO subroutine is run on $\mathcal{X}_S$ with T_S evaluations whose acquisition functions are each maximized over $\mathcal{X}_S$, it returns $\hat{x}_{T_S}$ satisfying $f_S^\star - f(\hat{x}_{T_S}) \le \varepsilon_{\mathrm{BO}}(T_S, |S|)$ with probability at least $1 - \delta_{\mathrm{BO}}$, where $\varepsilon_{\mathrm{BO}}(T, d)$ is non-increasing in T and non-decreasing in d.*

We use this as an abstract finite-sample guarantee for the BO subroutine. Different BO algorithms satisfy different versions of such a guarantee under different smoothness, kernel, and noise assumptions. We state our results in terms of the generic $\varepsilon_{\mathrm{BO}}$. Throughout, we assume the expert's variable selection randomness and the BO subroutine's randomness are independent.

EGBO helps only if two events occur. First, the expert must choose an active set that contains a good sparse solution. Second, BO must successfully optimize inside that active set. We capture this with the decomposition

$$f(x^*) - f(\hat{x}_T) = \underbrace{f(x^*) - f_S^\star}_{\text{selection gap}} + \underbrace{f_S^\star - f(\hat{x}_T)}_{\text{optimization gap}}, \tag{2}$$

where $f_S^\star = \max_{x \in \mathcal{X}_S} f(x)$. The selection gap measures the cost of restricting the search to the expert-selected subspace. The optimization gap measures BO's suboptimality within that subspace. If S contains the support of a sparse near-optimal witness $R \in \mathcal{R}_\eta$, then the restricted domain $\mathcal{X}_S$ contains a point whose value is within η of the global optimum. Therefore, $f(x^*) - f_S^\star \le \eta$. The remaining error is then controlled by the BO subroutine, which operates in dimension $|S|$ rather than the ambient dimension N.

Theorem 1 (One-shot EGBO)**.** *Suppose Assumptions 1, 2, and 3 hold. Consider one-shot EGBO: the expert is queried once to obtain an active set S, and BO is then run on $\mathcal{X}_S$ for T evaluations. Fix any witness support $R \in \mathcal{R}_\eta$. With probability at least $p^{|R|}(1 - \delta_{\mathrm{BO}}) \ge p^k(1 - \delta_{\mathrm{BO}})$, the returned point $\hat{x}_T$ satisfies*

$$f(x^*) - f(\hat{x}_T) \le \eta + \varepsilon_{\mathrm{BO}}(T, |S|). \tag{3}$$

Moreover, under Assumption 2, $\mathbb{E}[|S|] = \kappa p + (N - \kappa) q_{\mathrm{fp}}$, where $\kappa = |\mathcal{U}_\eta|$.

The theorem says that one-shot EGBO succeeds when the expert-selected active set contains a sparse near-optimal support. In that case, the total error is at most the sparsity gap η plus the error from running BO in the lower-dimensional space $\mathcal{X}_S$. When there are many different sparse near-optimal supports, the probability of covering at least one of them can be much higher. If $\mathcal{R}_\eta$ contains M' pairwise disjoint supports of size at most k, then

$$\Pr(\exists R \in \mathcal{R}_\eta : R \subseteq S) \ge 1 - (1 - p^k)^{M'}.$$

Proposition 1 (Coverage under adaptive expansion)**.** *Suppose EGBO performs Q expert-query rounds with monotonic active-set growth, $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_Q$. Under Assumption 2, define $\pi_Q = 1 - (1 - p)^Q$. For any fixed witness support $R \in \mathcal{R}_\eta$, $\Pr(R \subseteq S_Q) = \pi_Q^{|R|}$. In particular, $\Pr(R \subseteq S_Q) \ge \max\{0,\, 1 - k(1 - p)^Q\}$. The expected final active-set size is*

$$\mathbb{E}[|S_Q|] = \kappa\big(1 - (1 - p)^Q\big) + (N - \kappa)\big(1 - (1 - q_{\mathrm{fp}})^Q\big). \tag{4}$$

Proposition 1 shows the main tradeoff in adaptive EGBO. More expert-query rounds increase the probability of covering a sparse near-optimal support, because each relevant variable has more chances to be added. However, more rounds also increase the expected number of false-positive variables. Thus adaptivity can improve the selection gap, but may make the BO problem harder by increasing $|S_Q|$.

Remark 1 (When one-shot EGBO improves over vanilla BO)**.** *The one-shot bound shows that EGBO improves over vanilla BO when the cost of using the expert-selected subspace is smaller than the benefit of optimizing in lower dimension. In particular, if S contains a witness support, then EGBO is better than vanilla BO whenever*

$$\eta + \varepsilon_{\mathrm{BO}}(T, |S|) < \varepsilon_{\mathrm{BO}}(T, N). \tag{5}$$

The left-hand side is the EGBO bound, a selection gap η plus a BO optimization gap in dimension $|S|$. The right-hand side is the corresponding BO optimization gap in the full ambient dimension N.

All proofs are in Appendix C.

4 Experimental Evaluation

We first evaluate EGBO under controlled expert quality on Hartmann6 embedded in 200 dimensions (Section 4.1), then with Claude Opus 4.7 as the expert on FORMULATEBENCH's 24 plant-based formulation tasks. We compare against vanilla BO, random search, REMBO, SEBO, TuRBO, SAASBO, BO on k randomly selected variables, and fixed-support and unrestricted iterative LLM optimization. All methods receive $T = 20$ objective evaluations (30 in the batched experiments). Stochastic optimizers use 10 seeds (20 for the Hartmann6 Q-sweep), while each temperature-zero iterative LLM baseline contributes one cached trajectory per category. Appendix D gives full experimental details. Our code is available at `https://github.com/Food-Intelligence-Lab/egbo`.

4.1 Synthetic Experiments on Hartmann6

Methods. To Hartmann6, we append 194 dummy coordinates that do not affect the objective. The random subset + BO baseline is given the oracle sparsity level $k = 6$. We sweep two axes: (i) the number of expert query rounds Q, and (ii) the expert quality parameters.

Results. As shown in Figure 1, EGBO with a strong expert ($p = 0.9$, $q_{fp} = 0.01$) performs second best, after the oracle. EGBO with a weak expert ($p = 0.2$, $q_{fp} = 0.1$), in contrast, is outperformed by random search at $Q = 1$, because with low coverage it spends its entire budget in a subspace without a good solution. The (p, q_{fp}) heatmap (Appendix E.3) maps this selection gap and shows that EGBO closes most of the gap to the oracle only when recall is high and false positives are rare. Adaptive querying helps the weak expert, reducing regret from 3.20 at $Q = 1$ to 2.68 at $Q = 16$, as Proposition 1 predicts. For the strong expert, whose one-shot selection already covers a near-optimal support, the effect of adaptivity is small and depends on when the expert is queried (Appendix E.4).

4.2 FORMULATEBENCH

Methods. To compare EGBO to baselines on sustainable protein formulation tasks, we construct FORMULATEBENCH, 24 tasks (14 meat, 10 dairy) in which the goal is to select plant-based ingredients and concentrations to match an animal product's nutritional profile. Specifically, the objective is z-score RMSE over eight nutrients, computed from USDA FoodData Central [USDA, 2019]. All methods use the same filtered pools of 106 meat or 90 dairy ingredients drawn from NECTAR [NECTAR, 2025]; full construction details are in Appendix F.1. The problem's mathematical structure—non-negative least squares (NNLS)—guarantees an optimum with at most $m + 1 = 9$ ingredients under the simplex constraint (Proposition 2).

We additionally report two analytic floors per category: $\mathrm{Oracle}_{\text{full}}$, the constrained NNLS optimum over all d ingredients, and $\mathrm{Oracle}_{S_{\mathrm{LLM}}}$, the same restricted to the LLM's chosen subset. These give an empirical decomposition of the suboptimality bound, with $\mathrm{Oracle}_{\text{full}} \to \mathrm{Oracle}_{S_{\mathrm{LLM}}}$ the selection gap and $\mathrm{Oracle}_{S_{\mathrm{LLM}}} \to f(\hat{x}_T)$ the residual optimization gap.

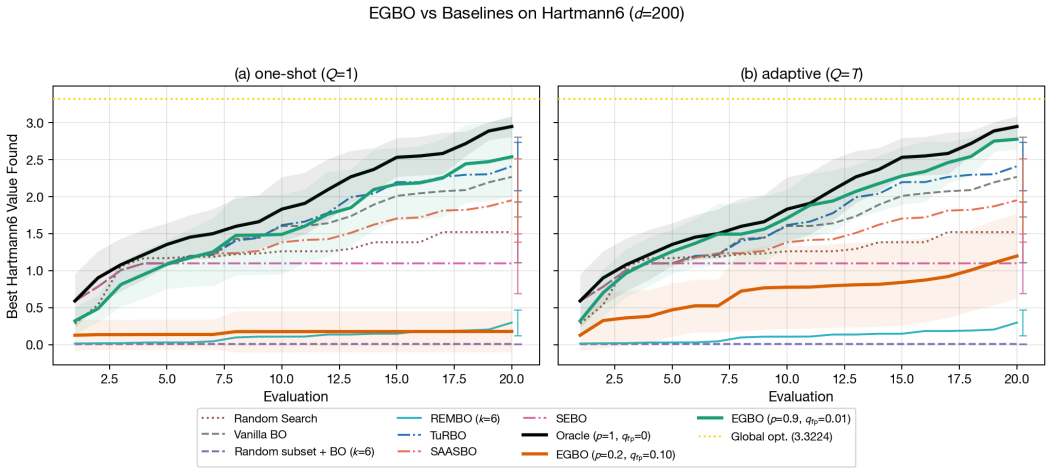

Figure 1: EGBO vs. baselines on Hartmann6 embedded in $N = 200$ dimensions, with 95% CIs. CIs for all methods are shown in Figure 5 of Appendix E. **(a)** One-shot expert query ($Q = 1$): the simulated expert proposes an active variable set once before BO begins, and BO runs in that fixed subspace for the remaining budget. **(b)** Fully adaptive ($Q = T$): the expert is queried before each of the $T = 20$ evaluations, including the five initial Sobol points, with monotonically growing active set. The strong-expert variant ($p = 0.9$, $q_{fp} = 0.01$) closely tracks the oracle in both regimes; the weak-expert variant ($p = 0.2$, $q_{fp} = 0.10$) underperforms random search under one-shot querying but improves under adaptive querying.

Context provided to the LLM expert. In the LLM-guided and LLM-only settings, Claude Opus 4.7 is told only that it is optimizing the overall similarity of a plant-based formulation to the target animal product and is shown the available ingredient names. After each evaluation, the iterative LLM methods receive only a scalar target-dissimilarity score (lower is better). This abstracts a setting in which the relationship between formulation and perceived similarity, e.g. as judged by a sensory panel, is unknown.

EGBO and the two iterative LLM baselines share the LLM expert's first proposed recipe, which counts as evaluation 1. EGBO fixes the ingredients in this recipe and uses BO to optimize their mixture fractions over the simplex. Its initial design contains the LLM expert's recipe and four seeded local perturbations, after which BO begins at evaluation 6. The fixed-support iterative LLM is restricted to the same ingredients but directly proposes new concentrations from scalar feedback. The unrestricted iterative LLM may change both ingredients and concentrations. We call the better of the two in each category *direct LLM optimization*. EGBO uses 10 BO seeds; each LLM baseline uses temperature-zero decoding and one cached trajectory per category.

BO configuration. We evaluate two one-shot EGBO variants that share the expert-selected ingredients, first recipe, initialization, and budget. *EGBO (default BO)* uses BoTorch 0.18.1's native `SingleTaskGP` defaults (RBF kernel, learned homoskedastic noise) with qLogNEI. *EGBO (LLM-configured BO)* uses one configuration chosen once by Claude Opus 4.7 from a prespecified menu, given only a high-level task description and the budget (Matérn-$5/2$ ARD kernel, default lengthscale prior, learned noise, qLogNEI), fixed across all 24 categories. Our theory does not cover expert configuration of the BO subroutine, so we treat it as an empirical ablation.

Results in the target setting. Our target setting is that of our deployment (Section 5) and of much experimental science, in which candidates are evaluated in batches and measurements are noisy. We therefore evaluate 10 batches of 3 recipes with Gaussian observation noise of standard deviation 0.15 times the category's normalizer, and report the noiseless case for comparison (Figure 2, Appendix F.7). Each method is scored by the true distance of its recommended recipe, the one with the best observed score. With noise, both iterative LLM arms stop improving after the first batch, and telling the LLM the noise level does not help (Appendix F.6). EGBO keeps improving and ends below direct LLM optimization (0.48 vs. 0.68; 19/24 categories, $p < 0.001$, Wilcoxon signed-rank). It also

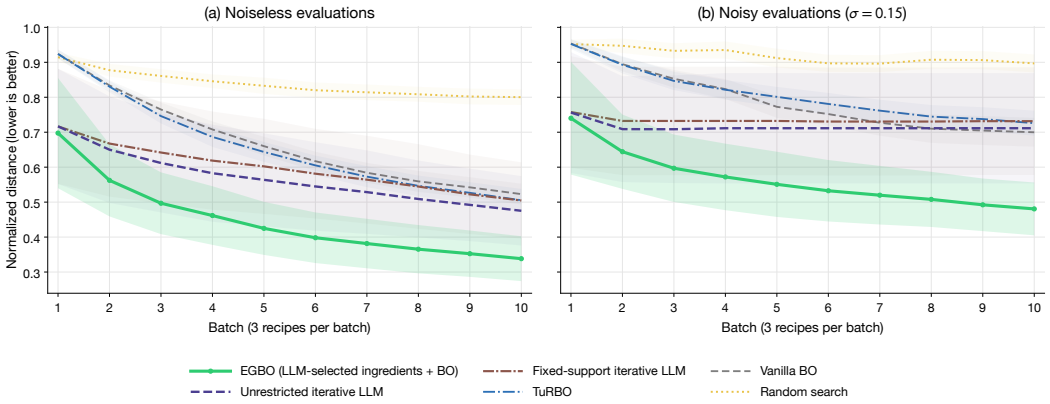

Figure 2: FORMULATEBENCH in our target setting of 10 batches of 3 recipes with Claude Opus 4.7 as the LLM, (a) without and (b) with Gaussian observation noise. Curves show the mean over 24 categories of the true normalized distance of each method's recommended recipe (lower is better); the shaded region shows a 95% confidence interval across categories for EGBO (lighter for other methods). EGBO and the conventional baselines average 10 seeds per category; each iterative LLM arm is one trajectory per category. Results with other LLMs are in Table 6; one-recipe-at-a-time results with all baselines are in Appendix F.2, and at other noise levels in Appendix F.6.

outperforms vanilla BO and TuRBO, on 21/24 and 22/24 categories, respectively. Without noise, EGBO also reaches the lowest distance after 30 evaluations, 0.34, compared with 0.44 for direct LLM optimization (19/24 categories, $p < 0.001$) and 0.51–0.52 for TuRBO and vanilla BO. With GPT-6.1 Sol or Gemini 4 Argon as the LLM (Table 6), direct LLM optimization matches EGBO without noise (differences not significant). Under noise, it falls behind for GPT-6.1 Sol (18/24, $p < 0.001$) but not significantly for Gemini 4 Argon (15/24, $p = 0.12$).

One recipe at a time. With one evaluation per step and no noise (Appendix F.2), LLM-configured EGBO has the lowest mean distance at 10 and 20 evaluations among all methods, including SAASBO, SEBO, and REMBO. It outperforms the unrestricted iterative LLM on 20/24 categories at evaluation 20. By 20 evaluations, the two NNLS oracles show that most of the remaining error comes from the choice of ingredients rather than their concentrations. Yet letting the expert add ingredients midway (adaptive EGBO, $Q = 2$) does not improve on one-shot EGBO on average (Appendix F.4).

When does each strategy win? We compare direct LLM optimization, full-space BO (the better of vanilla BO and BO warm-started from the LLM's recipe), and EGBO per category (Appendix F.5). Direct LLM optimization is best on 14/24 categories at evaluation 5 and ties EGBO at evaluation 10 (11 each). EGBO is best on 18/24 at evaluation 20, and full-space BO is best on at most 2/24 after evaluation 5. EGBO's advantage over direct LLM optimization at evaluation 20 grows with the LLM's *concentration error*, the gap between its first recipe and the best mixture of the same ingredients (Spearman $\rho = 0.54$, $p = 0.007$). Its advantage over full-space BO shrinks as the LLM's *ingredient choice* worsens ($\rho = -0.51$, $p = 0.011$). Together, these results suggest a division of labor: the LLM is most reliable at choosing what to vary, and BO at determining how much.

Do LLM agents delegate on their own? The strategies above are fixed pipelines. To test whether an LLM agent would choose to delegate, we gave three LLM agents two tools in the noisy setting, one that evaluates a recipe and returns its score and one that executes Python code (Appendix F.9). None ran a BO loop, even when told that BoTorch was installed, and 100 of 168 runs fit no model of their data at all. No agent significantly outperformed EGBO, and Gemini 4 Argon, which fit Gaussian processes most often, performed worst (0.66, vs. 0.55 for EGBO).

5 Real-World Deployment: Sustainable Dairy Formulation

We deploy EGBO in two real-world sustainable protein applications in partnership with a food company, each in our target setting of batched experiments with noisy panel evaluations. In each, the

goal is to move a plant-based product (yogurt or cottage cheese) toward a target sensory profile under a limited budget. Dairy products are particularly high-impact targets for plant-based reformulation due to the impact of the dairy industry on deforestation, climate change, land use, and animal welfare [Pendrill et al., 2022, Gerber et al., 2013, IPBES, 2019, von Keyserlingk et al., 2013]. For yogurt, we conduct a controlled comparison against a professional human food scientist. The food company has developed a commercially available, LLM-based agentic framework, which we denote Food Scientist Agent (FSA) and use as the expert in EGBO. To ensure reproducibility, we release FSA's output, the optimization variables, and all code. We set $Q = 1$, as both FSA's strong initial recipes (utility 0.767, 0.776) and performance of LLM-guided EGBO on FORMULATEBENCH suggest one-shot selection is reasonable for this setting. We leave real-world evaluation of adaptive expansion to future work.

5.1 Methods.

The full candidate ingredient pool is the food company's database ($N \approx 88{,}000$). Prior to optimization, FSA performed one-shot variable selection. Given the product category, the target dairy sensory profile, and the plant-based constraint, it selected a small set of active ingredients from the full pool. For yogurt, FSA selected $d = 7$ active ingredients. The sensory objective was defined over four attributes assessed on a 0 to 15 scale: Consistency (target $t_1 = 12$), Creaminess (target $t_2 = 5$), Tanginess (target $t_3 = 5$), and Uniformity (target $t_4 = 13$). The initial recipe had sensory scores $s^{(0)} = (5, 3, 10, 13)$, creating a nontrivial multi-objective problem in which one attribute already matches the target exactly while three require coordinated improvement. For cottage cheese, FSA selected $d = 9$ active ingredients. The sensory objective was defined over six attributes assessed on a 0 to 15 scale: Homogeneity (target $t_1 = 6$), Dispersion Phase (target $t_2 = 6$), Slipperiness (target $t_3 = 6$), Sourness (target $t_4 = 6.5$), Dairy (target $t_5 = 7$), and Off-Notes (target $t_6 = 0$). The initial recipe had sensory scores $s^{(0)} = (4, 8, 9, 10, 3, 7)$, presenting a more complex optimization landscape, since five of six attributes require substantial correction, with Off-Notes requiring near-complete elimination and Dairy requiring more than a doubling. In both cases, the total laboratory budget was $T = 10$ iterations of 3 recipes each (30 total recipes).

Scalar utility. In both campaigns, we aggregate the multi-objective sensory targets into a scalar utility score. For a candidate formulation x, let $s_j(x)$ denote the observed panel score for attribute j, t_j its target, m the number of sensory attributes, and $S_{\max}$ the maximum value of the sensory scale. With uniform weights $w_j = 1/m$, the scalar utility, which lies in $[0, 1]$ with 1 for an exact match, is

$$f(x) = \frac{1}{m} \sum_{j=1}^{m} \left(1 - \frac{|s_j(x) - t_j|}{S_{\max}} \right). \tag{6}$$

At each iteration, EGBO proposed a batch of recipes within the active subspace, which were prepared under standardized protocols and scored by the sensory panel.

5.2 Results

EGBO developed a plant-based yogurt formulation with a baseline utility of $f(x^{(0)}) = 0.767$ (recipe produced from FSA) and improved it to a best observed utility of $f(\hat{x}_T) = 0.992$ over 10 laboratory iterations (30 total recipes), a 29% improvement. The mean absolute error across the four sensory attributes fell from 3.5 to 0.13 points on a 0–15 scale. The first, space-filling batch within the FSA-selected subspace already raised utility to 0.917; BO, beginning at iteration 3, raised it further to 0.992. For cottage cheese, EGBO improved a plant-based formulation from a baseline utility of $f(x^{(0)}) = 0.776$ to a best observed utility of $f(\hat{x}_T) = 0.975$ over 10 laboratory iterations (30 total recipes), a 26% improvement.[2] Mean absolute attribute error fell from 3.6 to 0.67 points. Space-filling batches within the FSA-selected subspace raised utility to 0.850 by iteration 2, and BO raised it to 0.975. In both campaigns, the expert-selected subspace accounts for much of the gain and BO for the refinement, consistent with the decomposition in Equation (2). Tables 1 and 10 summarize the per-attribute results for yogurt and cottage cheese respectively.

[2]In the cottage cheese campaign, the six attribute weights were entered as $w_j = 0.17$ (summing to 1.02) rather than $1/6$, so all reported cottage cheese utilities are on that scale, with a maximum of 1.02. Rescaling to $w_j = 1/6$ gives a baseline of 0.761 and a best of 0.956, the same 26% improvement. Because the scaling is uniform, it does not affect the recipes proposed by BO.

Attribute	Target	Baseline	Best	Error	
				Before	After
Consistency	12	5	12	7	0
Creaminess	5	3	5	2	0
Tanginess	5	10	5	5	0
Uniformity	13	13	12.5	0	0.5
Utility $f(x)$	1.000	0.767	0.992	0.233	0.008

Table 1: Overall utility of the plant-based yogurt, as assessed by the median scores of a trained panel of five participants, improved 29% relative to baseline.



Figure 3: Best plant-based yogurt formulation (iteration 7, recipe 3).

5.3 EGBO vs. Human Food Scientist

We compare EGBO with a professional human food scientist (HFS) on the yogurt task. HFS had prior experience in recipe formulation but no prior experience specifically with plant-based dairy, to avoid the situation of HFS simply reproducing a suitable recipe from memory. This mirrors the setting of a company entering a new product category rather than reproducing a known recipe. HFS had unrestricted internet and literature access and reported consulting scientific and technical sources before beginning. HFS did not have access to EGBO's suggestions or results. We report two comparisons under the same approximate laboratory time (40 hours). In the workflow comparison, HFS develops a yogurt from scratch and reaches utility 0.850, while EGBO (FSA selection plus BO) reaches 0.992. The same-start comparison isolates refinement. From FSA's recipe (0.767), HFS reaches 0.867 and BO reaches 0.992. It is the laboratory analogue of our FORMULATEBENCH comparison between direct and BO-based refinement, with a human expert in place of the LLM.

Experiment 1: full workflow comparison. HFS was asked to develop, from scratch, a plant-based yogurt recipe targeting the same four sensory objectives used in the EGBO campaign (Consistency 12, Creaminess 5, Tanginess 5, Uniformity 13). Despite having full freedom to perform batching, HFS chose a conventional iterative workflow, preparing a candidate formulation, letting it cool, tasting it, deciding on the next modification, and repeating. A total of 29 trials were executed over approximately 40 hours. Of these, 19 were classified as failed trials (the operator abandoned the attempt and restarted from scratch), and 10 produced formulations deemed sufficiently promising to evaluate for utility. Figure 31 shows the utility progression across the 10 evaluated trials. The best utility achieved was 0.850 (trial 28), with a general upward trend as HFS progressively refined the plant base and texture.

Experiment 2: same-start refinement comparison. To isolate the refinement step, HFS was given FSA's recipe ($f(x^{(0)}) = 0.767$) and asked to optimize it toward the same four sensory targets. HFS was allocated 10 iterations, matching the EGBO campaign. Similarly, HFS chose to produce one recipe per iteration, and EGBO proposed batches of three recipes per iteration. Each HFS iteration required approximately 4 hours of laboratory time, for a total of approximately 40 hours.

Figure 32 shows the utility for both EGBO and HFS starting from the same baseline. EGBO reached a utility of 0.950 by iteration 4 and peaked at 0.992 by iteration 7. HFS improved more slowly, reaching 0.825 at iteration 3 and 0.867 at iteration 10. The gap was largest around iterations 6–7; by iteration 10, EGBO's best-so-far (0.992) exceeded HFS's best-so-far (0.867) by 14.4%.

Time and utility comparison. Table 2 summarizes the comparison. The results highlight two distinct advantages of the EGBO workflow. First, EGBO allows *parallelism*. It proposes a batch of 3 recipes per iteration, all of which can be prepared and evaluated in a single laboratory session, whereas HFS's sequential reasoning requires one-at-a-time execution. Second, EGBO is more *consistent*. HFS's trajectory shows regressions (iterations 4–5 and 8 in Experiment 2), reflecting the difficulty of mentally tracking coupled ingredient interactions across multiple sensory dimensions, whereas EGBO's GP surrogate maintains a global model that reduces such backsliding.

Comparison	Method	Start	Best	Relative Gain	Lab time
Experiment 1: Overall workflow	HFS from scratch	–	0.850	–	$\sim$40 h
Experiment 1: Overall workflow	EGBO: FSA + BO	0.767	0.992	+17% vs. HFS	$\sim$40 h
Experiment 2: Same-start refinement	HFS refinement	0.767	0.867	–	$\sim$40 h
Experiment 2: Same-start refinement	BO refinement	0.767	0.992	+14% vs. HFS	$\sim$40 h

Table 2: Human comparison on plant-based yogurt. We first compare the overall workflows under the same approximate laboratory time budget; the same-start comparison isolates the refinement step. EGBO's refinement includes two initial space-filling batches (Section 5).

6 Discussion

The current pace of development in sustainable proteins is incompatible with the urgency needed for climate change mitigation [Zurek et al., 2022]. Self-driving and agentic laboratories offer a path to the throughput this challenge requires [Abolhasani and Kumacheva, 2023, Fu et al., 2026], but their architecture—how frontier models, optimization algorithms, and experimental access divide the work—remains open. We study one interface in this architecture, when the model should delegate the choice of experiments to an optimizer, and find that EGBO outperforms a professional food scientist on plant-based yogurt formulation.

When should an agent delegate to an optimizer? Our results suggest a decision rule. Direct LLM optimization is preferable when the budget is very small, when evaluations are nearly noiseless and the LLM is a strong optimizer, or when the LLM's first proposal is already close to optimal. EGBO is preferable when the LLM can identify which variables matter but not their values, and when the budget is moderate or evaluations are noisy. When the LLM's choice of variables is poor, the agent should revise the design space (e.g., unrestricted LLM optimization or adaptive EGBO) or fall back to BO over the full space. Since an agent rarely knows its regime in advance, a natural next step is to diagnose it from early evaluations (e.g., replicating a recipe to estimate noise) before delegating. Because LLM agents with code execution did not delegate on their own (Section 4.2), the interface between agent and optimizer must currently be designed into the system. Training agents to recognize when to delegate, e.g., by post-training on noisy closed-loop tasks, is a promising direction.

Limitations. We study one interface of an agentic laboratory, the choice of experiments, rather than its full architecture. *Benchmark:* the main comparison uses Claude Opus 4.7, with replications across two other LLMs (Appendices F.7–F.8). These LLMs have no tools such as literature search or code execution (Appendix F.9 tests agents with code execution in a minimal tool-calling loop), and FORMULATEBENCH's nutritional objective and synthetic Gaussian noise are proxies for sensory evaluation. Learned taste and texture surrogates [Lee et al., 2023], robotic execution, and multi-fidelity BO [Mikkola et al., 2023, Sabanza-Gil et al., 2025] could further ease the evaluation bottleneck. *Deployment:* each campaign is a single run, compared with one food scientist but not with direct LLM optimization. The best utilities of both EGBO and the food scientist are maxima over single panel evaluations, which, as in FORMULATEBENCH (Appendix F.6), are biased upward, more so for EGBO's 30 recipes. Natural next steps are replicated re-scoring of top recipes, an iterative-LLM arm, and a subspace-sampling control. *Theory:* Assumption 2's independent coordinate inclusion is used only for the closed-form coverage bound, whereas LLM errors may be correlated across ingredients. The expert-quality parameters (p, q_{fp}) are defined relative to the unknown $\mathcal{U}_\eta$, and the analysis does not cover LLM-configured BO or shrinking active sets.

Societal impact. EGBO could be used to optimize formulations for cost or palatability at the expense of nutrition, so we focus on applications with environmental and public-health benefits. It could also reduce demand for food scientists. In our framework, however, humans remain responsible for objective specification and quality oversight, while EGBO automates iterative refinement, not scientific judgment.

Acknowledgments and Disclosure of Funding

Thank you to Theodore John Leonsis, the Bezos Earth Fund, Food System Innovations, and Stanford HAI for supporting this work.

References

Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.

Deborah Becker, Christophe Schmitt, Lionel Bovetto, Cornelia Rauh, Christopher McHardy, and Christoph Hartmann. Optimization of complex food formulations using robotics and active learning. *Innovative Food Science and Emerging Technologies*, 83:103232, 2023.

Jacqueline H Beckley, M Michele Foley, Elizabeth J Topp, Jack C Huang, and Witoon Prinyawiwatkul. *Accelerating new food product design and development*. John Wiley & Sons, 2008.

Paul Brunzema, Louis Tiao, Nhat Le, Kevin De Angeli, Yao Xuan, and Djordje Gligorijevic. Agentic Bayesian optimization through surrogate-augmented autoresearch. *arXiv preprint arXiv:2608.00316*, 2026.

Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

Liwei Cao, Danilo Russo, Kobi Felton, Daniel Salley, Abhishek Sharma, Graham Keenan, Werner Mauer, Huanhuan Gao, Leroy Cronin, and Alexei A. Lapkin. Optimization of formulations using robotic experiments driven by machine learning DoE. *Cell Reports Physical Science*, 2(1):100295, 2021.

Chih-Yu Chang, Milad Azvar, Chinedum Okwudire, and Raed Al Kontar. LLINBO: Trustworthy LLM-in-the-loop Bayesian optimization. *arXiv preprint arXiv:2505.14756*, 2025.

Abdoulatif Cissé, Xenophon Evangelopoulos, Vladimir V. Gusev, and Andrew I. Cooper. Language-based Bayesian optimization research assistant (BORA). *arXiv preprint arXiv:2501.16224*, 2025.

David Eriksson and Martin Jankowiak. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pages 493–503, 2021.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2019.

Romain Espinosa, Damian Tago, and Nicolas Treich. Infectious diseases and meat production. *Environmental and Resource Economics*, 76:1019–1044, 2020. doi: 10.1007/s10640-020-00484-3.

Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Kelsey Fu, Luna Lyu, Shuzhen Li, Suozhi Huang, Suheng Xu, Joy Zheng, Xuefeng Liu, Shilong Liu, Greg Barbone, Ying Liu, Linxi Jim Fan, Yuke Zhu, Matthew Davis, Kaiyi Jiang, Sherry Yang, Xu Kuang, Yuan Luo, Francesca Dominici, Christina Curtis, Xiaojie Qiu, Alberto Abadie, Caroline Uhler, Mary Tang, Joseph C. Wu, Ju Li, Jared Toettcher, José L. Avalos, Rebecca Rojansky, Huimin Zhao, John P. A. Ioannidis, Barry Rand, Emma Lundberg, Andrew S. Rosen, Zhuang Liu, Shana Kelley, Chenyan Xiong, Barbara E. Engelhardt, Thomas Montine, Christina V. Theodoris, Katherine S. Pollard, Fuxin Li, Xueyue Zhang, Tianyi Peng, Dmitri Basov, Xiang Qian, Eric Xing, Ali Yazdani, Jure Leskovec, Nigam Shah, Zhenan Bao, Pietro Perona, Sanfeng Wu, Clifford Brangwynne, Le Cong, and Mengdi Wang. Agentic laboratories of the future: Towards world models for scientific discovery. *Preprints*, 2026. doi: 10.20944/preprints202608.0213.v1. URL `https://www.preprints.org/manuscript/202608.0213`.

Pierre J. Gerber, Henning Steinfeld, Benjamin Henderson, Anne Mottet, Carolyn Opio, Jeroen Dijkman, Alessandra Falcucci, and Giuseppe Tempio. Tackling climate change through livestock: A global assessment of emissions and mitigation opportunities. Technical report, Food and Agriculture Organization of the United Nations (FAO), Rome, 2013.

Good Food Institute. Environmental impacts of alternative proteins. https://gfi.org/resource/environmental-impacts-of-alternative-proteins/, 2023. [Accessed 18-12-2025].

Rushil Gupta, Jason Hartford, and Bang Liu. LLMs for Bayesian optimization in scientific domains: Are we there yet? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Dong Han, Zhehong Ai, Pengxiang Cai, Shanya Lu, Jianpeng Chen, Zihao Ye, Shuzhou Sun, Ben Gao, Lingli Ge, Weida Wang, Xiangxin Zhou, Xihui Liu, Mao Su, Wanli Ouyang, Lei Bai, Dongzhan Zhou, Tao Xu, Yuqiang Li, and Shufei Zhang. ChemBOMAS: Accelerated BO in chemistry with LLM-enhanced multi-agent system. *arXiv preprint arXiv:2509.08736*, 2025.

Maeve Henchion, Maria Hayes, Anne Maria Mullen, Mark Fenelon, and Brijesh Tiwari. Future protein supply and demand: strategies and factors influencing a sustainable equilibrium. *Foods*, 6 (7):53, 2017.

IPBES. Global assessment report on biodiversity and ecosystem services. Technical report, Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services, Bonn, Germany, 2019.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at LLMs for material discovery: Are they actually good for Bayesian optimization over molecules? In *International Conference on Machine Learning*, pages 25603–25622, 2024.

Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization. *Proceedings of the VLDB Endowment*, 17(8):1939–1952, 2024.

Brian K Lee, Emily J Mayhew, Benjamin Sanchez-Lengeling, Jennifer N Wei, Wesley W Qian, Kelsie A Little, Matthew Andres, Britney B Nguyen, Theresa Moloy, Jacob Yasonik, et al. A principal odor map unifies diverse tasks in olfactory perception. *Science*, 381(6661):999–1006, 2023.

Sulin Liu, Qing Feng, David Eriksson, Benjamin Letham, and Eytan Bakshy. Sparse Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3754–3774. PMLR, 2023.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations*, 2024.

Petrus Mikkola, Julien Martinelli, Louis Filstroff, and Samuel Kaski. Multi-fidelity Bayesian optimization with unreliable information sources. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 7425–7454. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/mikkola23a.html.

NECTAR. Taste of the industry 2025. https://www.nectar.org/sensory-research/2025-taste-of-the-industry, 2025.

Marleen C Onwezen, Emily P Bouwman, Machiel J Reinders, and Hans Dagevos. A systematic review on consumer acceptance of alternative proteins: Pulses, algae, insects, plant-based meat alternatives, and cultured meat. *Appetite*, 159:105058, 2021.

Florence Pendrill, Toby A. Gardner, Patrick Meyfroidt, U. Martin Persson, Justin Adams, Tasso Azevedo, Mairon G. Bastos Lima, Matthias Baumann, Philip G. Curtis, Veronique De Sy, Rachael Garrett, Javier Godar, Elizabeth Dow Goldman, Matthew C. Hansen, Robert Heilmayr, Martin Herold, Tobias Kuemmerle, Michael J. Lathuillière, Vivian Ribeiro, Alexandra Tyukavina, Mikaela J. Weisse, and Chris West. Disentangling the numbers behind agriculture-driven tropical deforestation. *Science*, 377(6611):eabm9267, 2022.

Joseph Poore and Thomas Nemecek. Reducing food's environmental impacts through producers and consumers. *Science*, 360(6392):987–992, 2018.

Yusuf H Roohani, Andrew H. Lee, Qian Huang, Jian Vora, Zachary Steinhart, Kexin Huang, Alexander Marson, Percy Liang, and Jure Leskovec. BioDiscoveryAgent: An AI agent for designing genetic perturbation experiments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HAwZGLcye3.

Víctor Sabanza-Gil, Riccardo Barbano, Daniel Pacheco Gutiérrez, Jeremy S Luterbacher, José Miguel Hernández-Lobato, Philippe Schwaller, and Loïc Roch. Best practices for multi-fidelity Bayesian optimization in materials and molecular research. *Nature Computational Science*, 5(7):572–581, 2025.

Yihang Shen and Carl Kingsford. Computationally efficient high-dimensional Bayesian optimization via variable selection. In *International Conference on Automated Machine Learning*, 2023.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte Carlo tree search based variable selection for high dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2022.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. In *IEEE Transactions on Information Theory*, volume 58, pages 3250–3265, 2012.

Anna Thomas, Adam Yee, Andrew Mayne, Maya B. Mathur, Dan Jurafsky, and Kristina Gligorić. What can large language models do for sustainable food? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=f6SFHNfuMu.

Alexey Tomsov. The Role Of Sensory Panels At Redefine Meat. https://www.redefinemeat.com/blog/sensory-panels/, 2023. [Accessed 04-01-2025].

Richard Twine. Emissions from animal agriculture—16.5% is the new minimum figure. *Sustainability*, 13(11):6276, 2021.

Rutwick Surya Ulhas, Rajeev Ravindran, Alok Malaviya, Anushree Priyadarshini, Brijesh K Tiwari, and Gaurav Rajauria. A review of alternative proteins for vegan diets: Sources, physico-chemical properties, nutritional equivalency, and consumer acceptance. *Food research international*, 173: 113479, 2023.

USDA. Fooddata central, 2019. URL https://fdc.nal.usda.gov/. Accessed: 2025-10-04.

Marina A. G. von Keyserlingk, Nicolas P. Martin, Ermias Kebreab, Katharine F. Knowlton, Richard J. Grant, Mark Stephenson, Charles J. Sniffen, Joseph P. Harner, Allen D. Wright, and Steven I. Smith. Invited review: Sustainability of the US dairy industry. *Journal of Dairy Science*, 96(9): 5405–5425, 2013.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Michael R. Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528*, 2023.

Monika Zurek, Aniek Hebinck, and Odirilwe Selomane. Climate change and the urgency to transform food systems. *Science*, 376(6600):1416–1421, 2022.

A Symbols and Abbreviations

Table 3 contains a list of symbols and abbreviations introduced throughout this paper.

Symbol	Used For
Acronyms	
EGBO	Expert-Guided Bayesian Optimization
BO	Bayesian optimization
GP	Gaussian process
NNLS	nonnegative least squares
LLM	large language model
Problem parameters	
N	ambient input dimension
$\mathcal{X}$	input domain, $\mathcal{X} \subseteq \mathbb{R}^N$
f	black-box objective, $f : \mathcal{X} \to \mathbb{R}$
x^*	global maximizer of f on $\mathcal{X}$
$\|x\|_0$	number of nonzero (active) coordinates of x
$u_i^{\max}$	upper bound of the i-th coordinate
Sparsity structure	
k	target sparsity level (max size of any sparse near-optimal support)
η	sparsity gap: $f(\tilde{x}) \geq f(x^*) - \eta$ for some k-sparse $\tilde{x}$
$\tilde{x}^{(R)}$	witness for support R: a k-sparse near-optimum supported in R
R	a sparse near-optimal support, $R \subseteq [N]$, $\|R\| \leq k$
$\mathcal{R}_\eta$	family of all k-sparse η-near-optimal supports
M	multiplicity, $M = \|\mathcal{R}_\eta\|$
M'	number of pairwise disjoint supports in $\mathcal{R}_\eta$ used in bounds
$\mathcal{U}_\eta$	support union, $\bigcup_{R \in \mathcal{R}_\eta} R$
κ	size of the support union, $\kappa = \|\mathcal{U}_\eta\|$
Expert quality	
E	domain expert (human, LLM, or other knowledge source)
p	per-variable expert recall on $\mathcal{U}_\eta$
q_{fp}	per-variable expert false-positive rate outside $\mathcal{U}_\eta$
Algorithm and active set	
T	total BO budget (function evaluations)
Q	number of expert query rounds
τ	BO iterations per round, $\tau = \lfloor T/Q \rfloor$
S, S_q	active variable set (at round q)
d, d_Q	active set size, $d = \|S\|$ (resp. $d_Q = \|S_Q\|$)
$\mathcal{X}_S$	restricted domain $\{x \in \mathcal{X} : x_i = 0 \text{ for } i \notin S\}$
Convergence quantities	
$\hat{x}_T$	best point observed by EGBO after T evaluations
f_S^*	best achievable value in $\mathcal{X}_S$, $\max_{x \in \mathcal{X}_S} f(x)$
Δ_S	selection gap, $f(x^*) - f_S^*$
r_T	optimization gap, $f_S^* - f(\hat{x}_T)$
$\varepsilon_{\mathrm{BO}}(T, d)$	BO subroutine convergence rate as a function of budget T and dimension d
δ_{BO}	failure probability of the BO subroutine
NNLS objective (Section 4.2)	
V	ingredient-to-nutrient matrix, $V \in \mathbb{R}_{\geq 0}^{m \times N}$
v^*	target nutrient profile, $v^* \in \mathbb{R}^m$
m	number of nutrient targets

Table 3: Symbols and abbreviations used in this paper.

B Extended Related Work

LLMs in optimization and experimental design. A growing body of work uses LLMs as components of optimization or experimental design loops. LLAMBO [Liu et al., 2024] uses LLMs to inform

surrogate modeling and candidate generation for BO, while OPRO [Yang et al., 2024] treats the LLM itself as the optimizer over natural-language solution descriptions. Closer to our setting, GPTuner [Lao et al., 2024] uses LLMs to select database configuration knobs and propose search ranges before optimization. GPTuner can be viewed as a special case of EGBO with $Q = 1$ and a discrete database-knob domain. EGBO formalizes this class, extends it to continuous high-dimensional formulation problems with adaptive expansion, characterizes its regime of applicability, and deploys it for sustainable protein formulation.

LLMs versus BO, and hybrids. Head-to-head comparisons of LLM and BO optimizers report results that depend on the setting. LLMs match or beat BO for hyperparameter optimization under small budgets [Zhang et al., 2023], and LLM-based warm-starting helps most when observations are scarce [Liu et al., 2024]. In contrast, in genetic-perturbation and molecular design, LLM agents were insensitive to experimental feedback—replacing outcomes with permuted labels did not change their performance—and were outperformed by linear bandits and GP optimization [Gupta et al., 2025]. LLM features improve BO over molecules only when the model is pretrained or fine-tuned on domain data [Kristiadi et al., 2024]. Hybrid systems divide the work in different ways: an LLM intervenes when BO stagnates [Cissé et al., 2025], LLM proposals are screened by a statistical surrogate [Chang et al., 2025], an LLM partitions a chemical search space for BO [Han et al., 2025], or an LLM agent steers the search while delegating modeling and acquisition to a BO backend [Brunzema et al., 2026]. Data-driven alternatives to expert variable selection select active variables from evaluations [Song et al., 2022, Shen and Kingsford, 2023], at the cost of the budget spent identifying them. Our work differs in four ways. It compares direct LLM optimization, BO, and expert-guided BO on the same tasks. It characterizes when each is preferable as a function of budget, observation noise, batching, and the LLM. It isolates the LLM's contribution to a single interpretable decision, the choice of subspace, with accompanying theory. Finally, it deploys the result in a laboratory with a human sensory panel.

ML-driven formulation optimization. Within food and formulated-product development, Cao et al. [2021] coupled Thompson Sampling Efficient Multi-objective Optimization (TSEMO) with a naive Bayes stability classifier to optimize a five-ingredient liquid formulation, finding nine acceptable recipes in 15 working days. Becker et al. [2023] applied TSEMO to a three-ingredient whey protein formulation using a fully automated milli-fluidic platform. Neither approach exploits expert priors to reduce the search dimension. Thomas et al. [2025] found that LLMs can exceed the performance of human food scientists in experimental design for plant-based meats, as judged by other expert food scientists.

C Theory and Proofs

Throughout this appendix, we use the independence between the expert's variable-selection randomness and the BO subroutine's randomness assumed in Section 3.

C.1 Suboptimality Decomposition

Proof of Equation (2). By definition,

$$f_S^\star = \max_{x \in \mathcal{X}_S} f(x).$$

Adding and subtracting $f_S^\star$ gives

$$f(x^*) - f(\hat{x}_T) = \big(f(x^*) - f_S^\star\big) + \big(f_S^\star - f(\hat{x}_T)\big).$$

The first term is the selection gap and the second term is the optimization gap.

Now suppose $R \subseteq S$ for some $R \in \mathcal{R}_\eta$. By Assumption 1, there exists a witness $\tilde{x}^{(R)}$ with

$$\mathrm{supp}(\tilde{x}^{(R)}) \subseteq R \qquad \text{and} \qquad f(\tilde{x}^{(R)}) \geq f(x^*) - \eta.$$

Since $R \subseteq S$, we have $\tilde{x}^{(R)} \in \mathcal{X}_S$. Therefore,

$$f_S^\star = \max_{x \in \mathcal{X}_S} f(x) \geq f(\tilde{x}^{(R)}) \geq f(x^*) - \eta.$$

Thus $f(x^*) - f_S^\star \leq \eta$. □

C.2 One-Shot Coverage (Theorem 1)

Proof. Fix a witness support $R \in \mathcal{R}_\eta$. In the one-shot setting, each variable $i \in R$ is included in the active set S independently with probability p. Hence $\Pr(R \subseteq S) = p^{|R|}$. Since $|R| \le k$ and $p \in (0, 1]$, $p^{|R|} \ge p^k$.

On the event $R \subseteq S$, the subspace $\mathcal{X}_S$ contains the witness $\tilde{x}^{(R)}$. By the decomposition above, the selection gap is therefore at most η: $f(x^*) - f_S^\star \le \eta$.

By Assumption 3, when BO is run on $\mathcal{X}_S$ for T evaluations, it returns $\hat{x}_T$ satisfying $f_S^\star - f(\hat{x}_T) \le \varepsilon_{\mathrm{BO}}(T, |S|)$ with probability at least $1 - \delta_{\mathrm{BO}}$.

Combining the selection and optimization gaps gives

$$f(x^*) - f(\hat{x}_T) \le \eta + \varepsilon_{\mathrm{BO}}(T, |S|).$$

The coverage event $R \subseteq S$ and the BO success event are independent by assumption, so the joint event occurs with probability at least $p^{|R|}(1 - \delta_{\mathrm{BO}}) \ge p^k(1 - \delta_{\mathrm{BO}})$.

It remains to compute the expected active-set size. Under Assumption 2,

$$\mathbb{E}[|S|] = \kappa p + (N - \kappa) q_{\mathrm{fp}}.$$

□

C.3 Coverage of Multiple Disjoint Supports

Theorem 1 fixes one sparse near-optimal support R. If there are many such supports, the probability that the expert covers at least one of them can be larger.

Lemma 1 (Coverage of multiple disjoint supports)**.** *Let $\mathcal{R}' \subseteq \mathcal{R}_\eta$ be a collection of pairwise disjoint supports. Suppose each variable in $\mathcal{U}_\eta$ is included independently with probability π. Then*

$$\Pr(\exists R \in \mathcal{R}' : R \subseteq S) = 1 - \prod_{R \in \mathcal{R}'} \left(1 - \pi^{|R|}\right).$$

If additionally $|R| \le k$ for every $R \in \mathcal{R}'$, then

$$\Pr(\exists R \in \mathcal{R}' : R \subseteq S) \ge 1 - (1 - \pi^k)^{|\mathcal{R}'|}.$$

Proof. For a fixed support R, $\Pr(R \subseteq S) = \pi^{|R|}$. Because the supports in $\mathcal{R}'$ are pairwise disjoint, the events $\{R \subseteq S\}$ depend on disjoint sets of independent Bernoulli variables, so they are independent. Hence

$$\Pr(\text{no support in } \mathcal{R}' \text{ is covered}) = \prod_{R \in \mathcal{R}'} \left(1 - \pi^{|R|}\right).$$

If $|R| \le k$ and $\pi \in [0, 1]$, $\pi^{|R|} \ge \pi^k$, so $1 - \pi^{|R|} \le 1 - \pi^k$ and the bound follows. □

In the one-shot setting, $\pi = p$. Therefore, if $\mathcal{R}_\eta$ contains M' pairwise disjoint supports of size at most k, then

$$\Pr(\exists R \in \mathcal{R}_\eta : R \subseteq S) \ge 1 - (1 - p^k)^{M'}.$$

C.4 Adaptive Coverage (Proposition 1)

Proof. Consider any variable $i \in \mathcal{U}_\eta$. In each expert-query round, if i has not already been included, it is added with probability p. The probability that i is not added in any of the Q rounds is $(1 - p)^Q$, so $\pi_Q = 1 - (1 - p)^Q$.

Fix $R \in \mathcal{R}_\eta$. By Assumption 2, inclusion events are independent across coordinates, so $\Pr(R \subseteq S_Q) = \pi_Q^{|R|}$.

A simple union-bound gives $\Pr(R \not\subseteq S_Q) \le |R|(1 - p)^Q \le k(1 - p)^Q$, hence $\Pr(R \subseteq S_Q) \ge \max\{0,\, 1 - k(1 - p)^Q\}$.

Finally, by linearity of expectation,

$$\mathbb{E}[|S_Q|] = \kappa\big(1-(1-p)^Q\big) + (N-\kappa)\big(1-(1-q_{\mathrm{fp}})^Q\big).$$

□

D Shared Experimental Details

This appendix documents resources, GP/acquisition configuration, and LLM prompts shared by the synthetic Hartmann6 study (Appendix E), FORMULATEBENCH (Appendix F), and the real-world dairy deployment (Appendix G). Exact library versions are pinned in each release directory's `requirements.txt`.

D.1 Compute resources

All experiments in this paper run on CPU; no GPU is required at any point. We executed runs on a 14-inch MacBook Pro with an Apple M5 Pro chip (18-core CPU, 20-core GPU, 16-core Neural Engine), 48 GB unified memory, and OS Tahoe 26.4.1 with the library versions pinned in each release directory's `requirements.txt`. Per-experiment compute requirements are summarized in Table 4 and detailed below.

Table 4: Compute resources per experimental setting. Lab time refers to physical-laboratory wall-clock time (recipe preparation, cooling, sensory evaluation), which dominates the deployment budget.

Setting	Hardware	Per-run wall time	Total CPU time	Lab time
Hartmann6 ($N = 200$)	1 CPU core	5–10 min (SAAS); <1 min (others)	∼50 CPU-h	—
FORMULATEBENCH	1 CPU core	2–5 min (SAAS); <1 min (others)	$\sim 10^2$ CPU-h	—
Yogurt deployment	1 CPU core	<1 min per acquisition step	∼10 CPU-min	∼40 h
Cottage cheese deployment	1 CPU core	<1 min per acquisition step	∼10 CPU-min	∼40 h
FORMULATEBENCH (batched, other LLMs, agents)	1 CPU core	<1 min per run (BO); LLM-bound	∼25 CPU-h	—

Hartmann6 ($N = 200$). Each main comparison (one-shot or adaptive) covers 10 methods (including both EGBO presets) × 10 seeds × 20 evaluations = 2,000 function evaluations, and each heatmap cell adds $10 \times 20 = 200$ EGBO evaluations. Per-seed wall time is dominated by NUTS sampling in the SAAS-based methods (SAASBO, SEBO) at roughly 5–10 minutes. All other methods complete in under one minute per seed. Peak resident memory stays below 4 GB. The full Hartmann6 sweep (one-shot + adaptive, all expert regimes) is approximately 50 CPU-hours on the machine above.

FORMULATEBENCH. The main comparison evaluates nine stochastic optimization methods over 10 seeds and two temperature-zero iterative LLM baselines using one cached trajectory per category, for

$$24 \times (9 \times 10 + 2) \times 20 = 44{,}160$$

objective evaluations. The adaptive EGBO ablation adds $24 \times 10 \times 20 = 4{,}800$ evaluations.

Additional FORMULATEBENCH experiments. The batched experiments (Appendix F.7) comprise 3 LLMs × 2 noise levels × 24 categories of iterative LLM arms (one trajectory each) and EGBO, plus random search, vanilla BO, and TuRBO (10 seeds each), at 30 evaluations per run; the comparison with other LLMs, the noise sweeps with other LLMs, and the noise-aware arm follow the one-at-a-time protocol. Together these use about 25 CPU-hours and about 9,000 LLM API calls. The tool-using-agent test (Appendix F.9) used about 4,800 LLM API calls; its sandboxed code execution is negligible. All LLM responses and agent transcripts are released.

Real-world deployment. The computational footprint of each EGBO iteration is negligible. Acquisition-function optimization with `num_restarts` = 20, `raw_samples` = 1024, `sample_shape` = [512], `sequential` = True completes in under one minute on a single CPU core, and per-campaign cumulative acquisition compute is on the order of 10 minutes. The binding constraint is laboratory wall time, since each batch of $q = 3$ recipes requires approximately 4 hours of physical preparation, cooling, and trained-panel sensory evaluation, giving a total of ~ 40 hours of

lab time per 10-iteration campaign. Both the yogurt and cottage cheese campaigns ran on the same compute and panel infrastructure.

Total compute footprint. Summing across all reported experiments, the full reproduction of every numeric result and figure in this paper requires approximately $\mathbf{1.8 \times 10^2}$ CPU-hours of single-machine compute and about 2×10^4 LLM API calls (all responses are released). The experiments also used $\sim \mathbf{160}$ hours of physical laboratory time (about 80 hours for the two EGBO campaigns and 80 hours for the two HFS experiments).

D.2 GP and acquisition configuration

Unless noted otherwise, GP hyperparameters are fit by maximum marginal likelihood using `fit_gpytorch_mll` on `ExactMarginalLogLikelihood`, with `Standardize`($m = 1$) as the outcome transform. Acquisition functions are optimized with `optimize_acqf` using multiple L-BFGS-B restarts, and inputs are normalized to $[0, 1]^d$ before being passed to the GP. The synthetic study and conventional FORMULATEBENCH baselines use their method-specific configurations documented with the released runners. The native-default EGBO control uses BoTorch 0.18.1's default `SingleTaskGP`, whose covariance is an RBF kernel, together with learned homoskedastic noise and qLogNEI. The LLM-configured EGBO arm uses a Matérn-$5/2$ ARD kernel, default lengthscale prior, learned default noise, and qLogNEI. Both use a `SobolQMCNormalSampler` with `sample_shape` $= [256]$, `num_restarts` $= 10$, and `raw_samples` $= 512$. The deployment uses `sample_shape` $= [512]$ because it operates with batches of $q = 3$. The fully Bayesian baselines (SAASBO and SEBO) use `SaasFullyBayesianSingleTaskGP` with NUTS (warmup steps $= 512$, post-warmup samples $= 256$); on NUTS-fit failure we fall back to `SingleTaskGP`.

D.3 LLM expert: prompts and caching

The main FORMULATEBENCH comparison uses Claude Opus 4.7 (`anthropic/claude-opus-4.7` via OpenRouter) at temperature 0 to generate formulations under the limited-information regime used in the main experiment. The LLM is told that it is optimizing the overall similarity of a plant-based formulation to a target animal product and is shown only the available ingredient names. It is not shown the ingredient nutrient profiles, target nutrient vector, or mathematical form of the objective.

For each category, the first LLM response specifies both an ingredient set and initial concentrations. This recipe is shared by EGBO and the two iterative LLM baselines and counts as the first objective evaluation. EGBO fixes its ingredients and uses BO to optimize their concentrations. The fixed-support iterative LLM is restricted to the same ingredients but directly proposes new concentrations, whereas the unrestricted iterative LLM may change both ingredients and concentrations. After each evaluation, the iterative LLM methods receive the previously evaluated formulations and their scalar target-dissimilarity scores, with lower values indicating better performance. They receive no additional information about the hidden nutrition objective.

All formulation-expert calls use the system message "You are an expert plant-based food scientist." The response is parsed as a JSON object containing one complete recipe with at most $k = 9$ ingredients and fractions summing to one. The cap $k = 9$ follows from the NNLS sparsity bound (Proposition 2, Appendix F). The initial user prompt is reproduced in Figure 4.

For reproducibility without API access, each OpenRouter response is cached on disk using a SHA-256 hash of the model, system prompt, user prompt, and decoding settings. We use one cached temperature-zero trajectory per iterative LLM method and category, and include the cached trajectories and per-category result JSONs in the code release.

BO-configuration prompt. The BO-configuration call is also made once at temperature zero and cached. It states that an expert has proposed a recipe of at most nine ingredients, that BO will refine mixture fractions over the simplex, that the total budget is 20 evaluations, and that only scalar target-dissimilarity feedback will be observed. From a fixed menu of kernel, lengthscale-prior, noise, acquisition, and ARD choices, Claude selected a Matérn-$5/2$ kernel, default lengthscale prior, default/learned noise, qLogNEI, and ARD. This configuration is used unchanged for every category.

The Food Scientist Agent (FSA) used in the real-world deployment (Section 5) is a commercially available agentic system. We use its recipes as one-shot inputs and do not invoke it during BO.

User prompt

You are optimizing the overall similarity of a plant-based formulation to animal-based <*category*>. The evaluator is a black box.

Available plant-based ingredients:
<*ingredient names*>

No formulations have been evaluated yet. Propose the strongest initial formulation using your prior knowledge.

Propose exactly one complete formulation. Use at most 9 available ingredients, nonnegative quantities, and no ingredient outside the catalog. Return only JSON in this exact shape:

```
{"recipe":[{"ingredient":"exact catalog name",
  "fraction":0.25}]}
```

Fractions must sum to 1. Do not include commentary.

Figure 4: Prompt used to obtain the shared initial recipe for the LLM-informed methods in FORMULATEBENCH. The system message is "You are an expert plant-based food scientist." EGBO fixes the ingredients in this recipe and optimizes their concentrations. At subsequent evaluations, the iterative LLM baselines instead receive the previously evaluated formulations and their scalar target-dissimilarity scores. The cap $k = 9$ is derived from Proposition 2.

E Synthetic Benchmark: Hartmann6 ($N = 200$)

E.1 Setup

Problem. We embed the standard 6-D Hartmann function into $N = 200$ ambient dimensions. The embedded objective is $f(x) = \text{Hartmann6}(x_{1:6})$, so coordinates $7, \ldots, 200$ are inert and never enter the function. The search domain is $[0, 1]^{200}$. For *subspace methods* (oracle, random subset, REMBO, EGBO), BO optimizes only over the active set S and inactive coordinates are pinned at 0 (matching the support convention $\text{supp}(x) \subseteq S$ used in Assumption 1). Full-dimensional methods (vanilla BO, TuRBO, SAASBO, SEBO) optimize over all 200 coordinates. Because f ignores dimensions 7–200, no pinning is required, but TuRBO initializes its trust-region center at $\mathbf{0.5} \in [0, 1]^{200}$ following BoTorch defaults.

Budget. Total budget $T = 20$, of which $n_{\text{init}} = 5$ are Sobol points (BoTorch `SobolEngine`, scrambled, with the i-th Sobol draw seeded as `seed` $+ i$ so that EGBO and Oracle share init points across active sets) and the remaining 15 are acquisition steps with batch size $q = 1$. We run 10 random seeds per (method, regime). Shaded bands are 95% confidence intervals from the standard error of the mean.

Acquisition optimization. qLogNEI is optimized with `num_restarts` $= 10$, `raw_samples` $= 512$.

Simulated expert. The expert is implemented in `expert_step()` as independent per-coordinate Bernoulli draws, in which each of the 6 relevant coordinates is added to the active set with probability p, and each of the 194 irrelevant coordinates with probability q_{fp}. The convergence-curve experiments report two regimes as representatives of the "EGBO wins" and "EGBO loses" regions: a strong expert $(p, q_{fp}) = (0.9, 0.01)$ and a weak expert $(p, q_{fp}) = (0.2, 0.10)$, plus the oracle $(p, q_{fp}) = (1, 0)$. In adaptive runs the active set grows monotonically across rounds.

Baselines.

- **Random search:** uniform $[0, 1]^{200}$.
- **Vanilla BO:** `SingleTaskGP` + qLogNEI on the full 200-D box.

- **Random subset + BO:** fixed random $k = 6$ subset (seed offset $+10\,000$); BO is run inside it with the rest pinned at 0. We intentionally give this baseline the oracle sparsity k so that the comparison isolates the value of *which* variables are chosen, not *how many*.
- **Oracle:** BO restricted to the true relevant 6-D subspace.
- **REMBO** [Wang et al., 2016]: Gaussian random projection $A \in \mathbb{R}^{200\times 6}$ with $A_{ij} \sim \mathcal{N}(0, 1/\sqrt{6})$. Latent bounds are computed from 4096 Sobol probes back-projected through A rather than the canonical $[-\sqrt{6}, \sqrt{6}]$ box. This gave consistently better REMBO performance in our setting and we report results under the favorable bounds.
- **TuRBO** [Eriksson et al., 2019]: trust-region length $L_0 = 0.8$, $L_{\min} = 2^{-7}$, $L_{\max} = 1.6$, success tolerance 3, failure tolerance $\max(4, d)$ following the original paper's dimension-dependent rule (with batch size $q = 1$).
- **SAASBO** [Eriksson and Jankowiak, 2021]: `SaasFullyBayesianSingleTaskGP` with NUTS (warmup 512, samples 256), with the fallback noted in Appendix D.2.
- **SEBO** [Liu et al., 2023]: SAAS GP with smooth-ℓ_0 penalty, $\lambda_{\max} = 1.0$, smoothing $a = 0.05$, linear homotopy $\lambda_t = \lambda_{\max}\, t/T$, penalty centered at $x_{\text{ref}} = 0$. These values follow the BoTorch reference implementation rather than the original SEBO settings. We adopted the BoTorch values because they were robust across our $N = 200$ regime without per-task tuning.

Run counts and compute. With 10 seeds, $T = 20$ evaluations, and 10 methods (including both EGBO presets), each main Hartmann6 comparison is $10 \times 10 \times 20 = 2{,}000$ function evaluations, and each heatmap cell adds $10 \times 20 = 200$ EGBO evaluations. Full per-seed traces are written to `results/<NAME>/{oneshot,adaptive}/raw.json`. Runs were executed on a single CPU node; per-seed wall time is dominated by NUTS in SAASBO/SEBO at roughly 5–10 minutes.

Entry points. `run_hartmann6.py` (per-method runs); `run_hartmann6.sh` (fan-out across seeds); `merge_hartmann6.py` and `plot_hartmann6.py` (aggregation and figures).

E.2 Best-so-far convergence (95% CIs)

Figure 5 shows Figure 1 with 95% CIs for all methods.

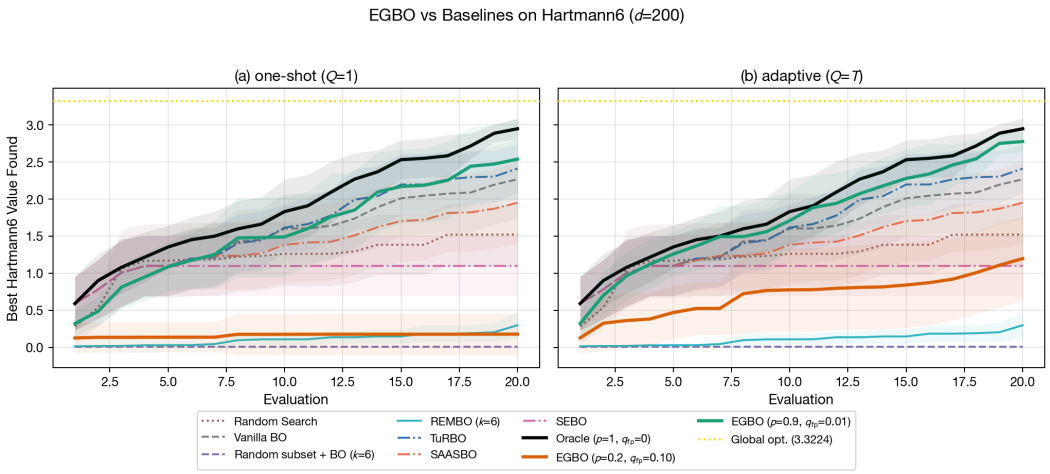

Figure 5: Hartmann6: best-so-far convergence with 95% CIs over 10 seeds.

E.3 Expert-quality heatmap

This experiment maps the (p, q_{fp}) plane empirically to identify the regimes where EGBO improves over vanilla BO. Recall is swept over $p \in \{0.1, 0.2, 0.4, 0.6, 0.8, 0.9\}$ and false-positive rate over $q_{fp} \in \{0, 0.01, 0.05, 0.10, 0.20\}$, all at adaptive $Q = T$ (the schedule of Figure 1(b)) to give the

expert the maximum number of opportunities to recover from misses. For each cell we run 10 seeds at $T = 20$ evaluations and report the gap-closed fraction $(f_{\text{EGBO}} - f_{\text{vanilla}})/(f_{\text{oracle}} - f_{\text{vanilla}})$, where 1.0 means EGBO matches the oracle, 0 means it matches vanilla BO, and negative values indicate EGBO underperforms vanilla BO.

Both recall and false-positive rate matter. With no false positives ($q_{fp} = 0$), EGBO closes most of the oracle gap at every $p \geq 0.4$ (gap-closed fractions 0.56–0.82). Once $q_{fp} > 0$, only the high-recall, low-false-positive cells close the majority of the gap (0.68 and 0.75 at $p \in \{0.8, 0.9\}$, $q_{fp} = 0.01$). At $q_{fp} \geq 0.05$, EGBO closes at most 38% of the gap even at $p = 0.9$, and most cells with $p \leq 0.6$ fall below vanilla BO. At $p \leq 0.2$, EGBO underperforms vanilla BO at every q_{fp}. This is consistent with Theorem 1 and Equation (4). False positives enter the bound through $\varepsilon_{\text{BO}}(T, |S|)$, and under adaptive querying they accumulate across rounds, so even a modest q_{fp} inflates $|S_Q|$ enough to erode the benefit of searching a lower-dimensional subspace. The two presets used in Section 4.1, `egbo_strong` at $(0.9, 0.01)$ (gap-closed fraction 0.75) and `egbo_weak` at $(0.2, 0.10)$ (-1.57), sit in the success and failure regions respectively, picked deliberately to illustrate the two regimes.

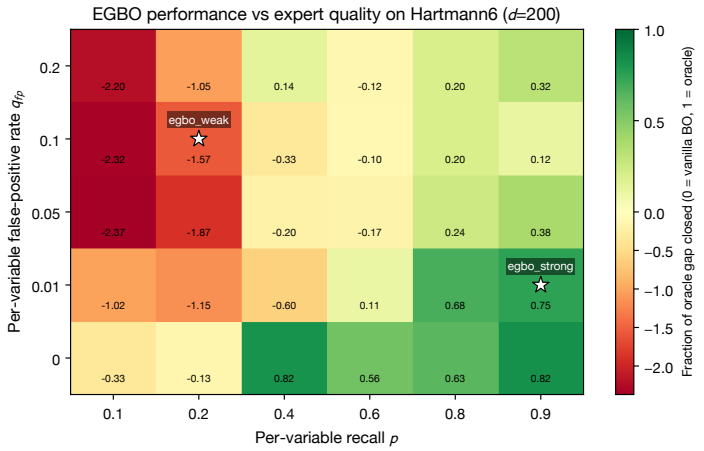

Figure 6: Fraction of the oracle–vanilla gap closed by adaptive EGBO ($Q = T$) on Hartmann6 in $N = 200$ ambient dimensions, as a function of the per-variable recall p (columns) and false-positive rate q_{fp} (rows). Each cell is the mean over 10 random seeds, $T = 20$ evaluations; the colour scale is centred at 0 (vanilla BO), and negative values mean EGBO underperforms vanilla BO. Stars mark the two presets used in the main text: `egbo_strong` at $(p = 0.9, q_{fp} = 0.01)$ and `egbo_weak` at $(p = 0.2, q_{fp} = 0.10)$. EGBO closes the majority of the oracle gap only when recall is high and false positives are rare ($p \geq 0.4$ at $q_{fp} = 0$; $p \geq 0.8$ at $q_{fp} = 0.01$). At $q_{fp} \geq 0.05$ it closes at most 38% of the gap, and at $p \leq 0.2$ it underperforms vanilla BO over the full 200-dimensional space regardless of q_{fp}.

E.4 Intermediate-Q sweep

This experiment tests Proposition 1's prediction that adaptive expert-query rounds help only when the initial prior is incomplete. We sweep $Q \in \{1, 2, 4, 8, 16\}$ for the two representative presets (`egbo_strong`, `egbo_weak`) at 20 seeds per cell, with all other settings matching Appendix E.1. In this sweep the expert is queried once before the initial design and once before each of the $T - n_{\text{init}} = 15$ BO steps, so 16 is the maximum number of distinct query rounds. A run with $Q = 20$ reproduces $Q = 16$ exactly and is omitted. Figure 1(b) and the heatmap in Appendix E.3 instead query the expert before each of the 20 evaluations, including the five initial Sobol points. The two schedules give different results on the same seeds. Under the Figure 1(b) schedule, the strong expert's final regret is 0.55 (vs. 0.79 one-shot), whereas under the sweep's schedule it is 1.19 at $Q = 16$ (vs. 0.98 at $Q = 1$, 20 seeds). Whether adaptivity helps a strong expert therefore depends on when the expert is queried.

The two regimes split along expert quality, as Proposition 1 predicts. With a strong expert, regret is essentially flat across Q (gap-to-optimum 0.98 at $Q = 1$, 1.19 at $Q = 16$), because recall is high enough that one round suffices to cover a near-optimal support, and additional rounds only inflate

$|S_q|$ with false positives. The trend is within the 95% CIs and we do not claim statistical significance at $\alpha = 0.05$. With a weak expert, regret drops monotonically from 3.20 at $Q = 1$ to 2.68 at $Q = 16$, with the largest improvement between $Q = 8$ and $Q = 16$, when the expert is consulted before every BO step. The mechanism is made concrete by the $|S_q|$ trajectories in Appendix E.5. Weak experts only assemble a usable support after many rounds, while strong experts have nothing to add.

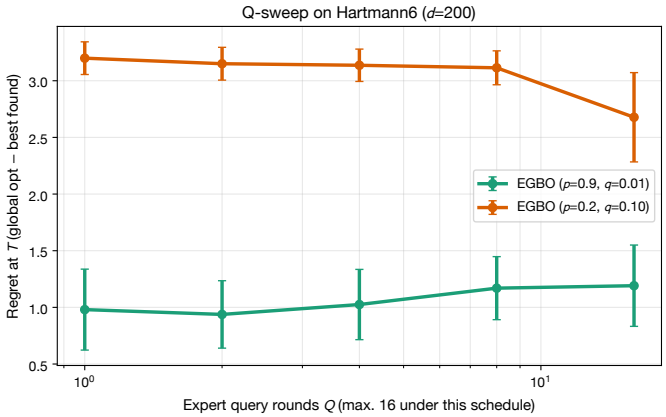

Figure 7: Final regret (gap to the global optimum, lower is better) as a function of the number of expert query rounds Q on Hartmann6 in $N = 200$ ambient dimensions, $T = 20$ evaluations, 20 random seeds per cell. Error bars are 95% confidence intervals. With a strong expert ($p = 0.9$, $q_{fp} = 0.01$, green), regret is essentially flat across Q and trends slightly upward from 0.98 at $Q = 1$ to 1.19 at $Q = 16$. Adaptive querying does not help, and can mildly hurt, a strong expert because per-iteration false positives accumulate and grow $|S_q|$ beyond the true 6-dimensional support, making the BO problem harder under a fixed evaluation budget (the active-set trajectory in Appendix E.5 makes this growth concrete). The trend is within the 95% CIs and we do not claim statistical significance at $\alpha = 0.05$. With a weak expert ($p = 0.2$, $q_{fp} = 0.10$, orange), regret drops from 3.20 at $Q = 1$ to 2.68 at $Q = 16$, with the largest improvement between $Q = 8$ and $Q = 16$, once the expert is consulted before every BO step. $Q = 16$ is the maximum number of distinct query rounds under this schedule ($T - n_{\text{init}} = 15$ BO steps plus the initial query). The two regimes match the prediction of Proposition 1: adaptivity helps only when the prior is incomplete.

E.5 Active-set trajectory

Figure 8 visualizes the active-set dynamics predicted by Equation (4) directly on the Hartmann6 task. For each (Q, expert) configuration we record $|S_q|$ (total active-set size) and $|S_q \cap \mathcal{U}_\eta|$ (number of relevant variables in the active set, where $|\mathcal{U}_\eta| = 6$ since coordinates 7–200 are inert) at every BO iteration, averaged over 20 seeds. We plot three representative Q values: $Q = 1$ (no expansion after initialization), $Q = 4$ (expand every four BO iterations), and $Q = 16$ (expand before every BO step), under both expert presets.

The two panels expose the mechanism behind the Q-sweep regret pattern in Appendix E.4. Under a strong expert (left), $|S_q|$ jumps near the true support size in the first round and the dashed line ($|S_q \cap \mathcal{U}_\eta|$) tracks the solid one closely throughout. The relevant support is essentially covered after one query, so larger Q only adds the slow accumulation of false positives at rate $q_{fp} = 0.01$ per round per inert coordinate. Equation (4) predicts $\mathbb{E}[|S_Q|]$ rises from ≈ 7.3 at $Q = 1$ to ≈ 35 at $Q = 16$ in this regime. The resulting BO dimension cost is what produces the slight upward regret trend in Appendix E.4. Under a weak expert (right), $|S_q|$ grows slowly across rounds for small Q and aggressively for large Q, with the dashed line approaching the $|\mathcal{U}_\eta| = 6$ ceiling only by mid-trajectory at $Q = 16$. This is the regime in which adaptivity pays off, because each additional round provides a fresh chance to recruit a missing relevant variable, at the cost of a large absolute $|S_q|$ dominated by false positives ($\mathbb{E}[|S_{16}|] \approx 164$ by Equation (4)).

Together, this active-set view, the Q-sweep regret in Appendix E.4, and the (p, q_{fp}) heatmap in Appendix E.3 agree. The green band of the heatmap corresponds to fast $|S_q \cap \mathcal{U}_\eta|$ saturation with

controlled $|S_q|$, while the red band corresponds to either incomplete coverage (low p) or excessive $|S_q|$ (high q_{fp}).

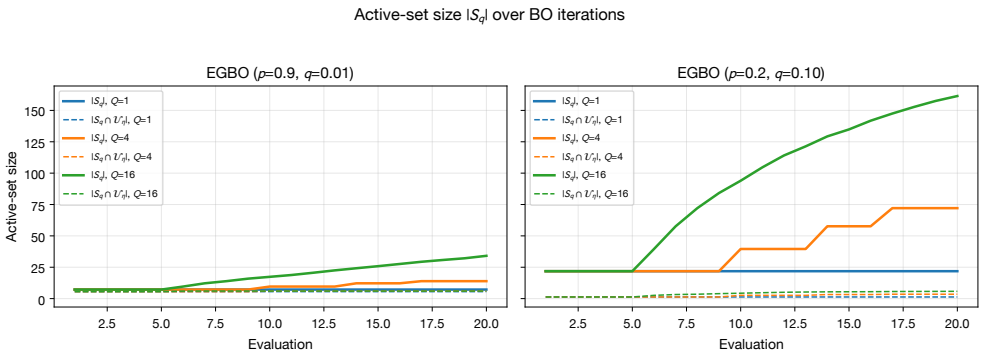

Figure 8: Mean active-set size $|S_q|$ over BO iterations for three representative Q values $\{1, 4, 16\}$ on Hartmann6 in $N = 200$ ambient dimensions, 20 random seeds. *Left:* strong expert ($p = 0.9, q_{fp} = 0.01$). *Right:* weak expert ($p = 0.2, q_{fp} = 0.10$). Solid lines plot $|S_q|$ (total active-set size). Dashed lines plot $|S_q \cap \mathcal{U}_\eta|$ (number of relevant variables in the active set; $|\mathcal{U}_\eta| = 6$ for embedded Hartmann6). Under a strong expert $|S_q|$ saturates within one round near the true support size; under a weak expert, $|S_q|$ grows slowly across many rounds before it covers a near-optimal support, and false positives accumulate when Q is large. This visualization makes Equation (4) of Proposition 1 concrete: low recall p makes round-by-round coverage of $\mathcal{U}_\eta$ unlikely, so weak experts require many rounds before the active set is useful.

F FORMULATEBENCH

F.1 Setup

Problem. The ground set is the union of plant-based ingredients in NECTAR with nutrition data in USDA FoodData Central. Before any method or LLM call, we remove animal-derived, ambiguous, or directly target-revealing entries. For meat, we remove `egg white`, `egg whites`, `fat`, `milk`, and `milk proteins`; for dairy, we remove `mozzarella`. This leaves a common eligible pool of $d = 106$ meat ingredients and $d = 90$ dairy ingredients for every method. The objective is the per-dimension z-score RMSE against an animal-product target on $m = 8$ nutrients (energy, protein, fat, saturated fat, carbohydrates, fiber, sugars, sodium), with a per-category measurement mask $M \subseteq \{1, \ldots, 8\}$ for missing USDA values. Ingredient-set z-score statistics (μ_j, σ_j) are computed once over the filtered candidate ground set.

Simplex parameterization and initialization. Compositions live on Δ^{d-1} and are parameterized as logits $z \in [-12.0, 0.0]^d$ mapped through a softmax. The -12 floor ($e^{-12} \approx 1.5 \times 10^{-6}$) keeps gradients stable while leaving the simplex effectively open. Conventional BO methods use five method-specific initial evaluations that are not centered on the LLM recipe. Vanilla BO, TuRBO, SAASBO, and SEBO use the uniform mixture plus four seeded Dirichlet draws over the full ingredient simplex. Random-subset BO uses the analogous design within its random $k = 9$ subset, and REMBO uses five seeded Sobol points in its latent space. For EGBO, the LLM's first recipe counts as evaluation 1 and the remaining four initial designs are seeded local logit-normal perturbations with standard deviation 0.35 around that recipe. BO acquisition optimization begins at evaluation 6. The fixed-support and unrestricted iterative LLM baselines share the same first recipe and begin feedback-conditioned proposals at evaluation 2.

Normalization across categories. Raw distances are not directly comparable because some animal-product targets are intrinsically farther than others from a typical plant-ingredient mixture. We divide each category's trajectory by the mean distance of the first random-search evaluation over 10 seeds. Thus a normalized distance of 0.5 means that a method has reduced the distance to half that of an average initial random formulation for that category. Aggregate figures average these normalized best-so-far trajectories across categories; per-category figures report raw distance.

Sparsity bound justifying $k = 9$. When the objective is nonnegative least squares with m nutrient targets, a classical result gives a bound on the sparsity of optimal solutions. We use this to set the LLM expert's selection cap to $k = m + 1 = 9$.

Proposition 2 (Sparsity of NNLS nutrition matching)**.** *Let $V \in \mathbb{R}_{\geq 0}^{m \times N}$ be a nonnegative ingredient-to-nutrient matrix and $v^* \in \mathbb{R}^m$ a target nutrient profile. Consider the nonnegative least-squares problem*

$$\min_{x \geq 0} \|Vx - v^*\|_2^2. \tag{7}$$

There exists an optimal solution x^* with $\|x^*\|_0 \leq m$. If the problem additionally imposes a simplex constraint $\sum_{i=1}^N x_i = 1$, there exists an optimal solution with $\|x^*\|_0 \leq m + 1$.

Proof. Among all optimal solutions, pick one for which $|I| = \|x^*\|_0$ is minimal, and let $I = \operatorname{supp}(x^*)$. Suppose for contradiction that $|I| > m$. Then the columns $\{V_i : i \in I\}$ are linearly dependent in $\mathbb{R}^m$, so there exists $d \neq 0$ with $V_I d = 0$. Extending d by zeros outside I, the perturbation $x^* + td$ preserves the objective for all t. Choose t with largest absolute value such that $x^* + td \geq 0$; at least one coordinate becomes zero, contradicting minimality of $|I|$. Hence $|I| \leq m$.

The simplex-constrained version follows the same argument with d restricted to $\ker V_I \cap \{d : \sum_i d_i = 0\}$, the null space of $\begin{pmatrix} V_I \\ \mathbf{1}^\top \end{pmatrix} \in \mathbb{R}^{(m+1)\times|I|}$, which has dimension at least $|I| - (m+1)$. If $|I| > m+1$ this is non-trivial, and the rest of the argument gives $|I| \leq m + 1$. When no box constraint binds, the argument produces a k-sparse optimum for any k at least the stated bound, so $\eta(k) = 0$. □

Budget. $T = 20$, $n_{\text{init}} = 5$ for BO methods, and batch size $q = 1$. EGBO and the stochastic conventional optimizers use 10 seeds per category. Each temperature-zero iterative LLM baseline contributes one cached trajectory per category.

Acquisition optimization. Both reported EGBO variants use qLogNEI with `num_restarts` $= 10$ and `raw_samples` $= 512$. The default variant uses native BoTorch 0.18.1 `SingleTaskGP` defaults; the LLM-configured variant uses the single global configuration documented in Appendix D.2.

Baselines. All methods receive the same filtered candidate pool and optimize over the simplex parameterization above. Full-space methods may use every eligible ingredient; the expert-selected support is an EGBO intervention, not a restriction imposed on those baselines. Random-subset-plus-BO uses a fixed random $k = 9$ subset (seed offset $+10{,}000$), matching EGBO's maximum initial support size. REMBO uses a Gaussian projection $A \in \mathbb{R}^{d \times 9}$ with latent radius $\sqrt{9} = 3$. TuRBO uses failure tolerance $\max(4, d)$. We additionally compare with fixed-support and unrestricted iterative LLM optimization as described in Appendix D.3.

LLM expert. Active-set selection uses Claude Opus 4.7 at temperature 0 as documented in Appendix D.3; the prompt is reproduced in Figure 4.

Compute. FORMULATEBENCH runs were executed on CPU. The SAAS-based methods (SAASBO, SEBO) dominate wall time at ∼2–5 minutes per (seed, category, method). The full sweep took on the order of 10^2 CPU hours.

Entry point. Run `python run_formulatebench.py --domain {meat,dairy} --budget 20 --seeds 10 --n-init 5`. Per-seed traces are written atomically as JSON.

F.2 One recipe at a time

Across all 24 categories, LLM-configured EGBO achieves the lowest mean normalized distance at both 10 and 20 evaluations (lower is better). Figure 9 reports results separately for meat and dairy. At evaluation 10, LLM-configured EGBO outperforms the unrestricted iterative LLM on meat (0.46 vs. 0.58), while the unrestricted LLM has a slightly lower mean distance on dairy (0.44 vs. 0.45). By evaluation 20, LLM-configured EGBO has the lower mean distance in both domains, 0.35 vs. 0.52 on meat and 0.36 vs. 0.41 on dairy.

Across individual categories, LLM-configured EGBO outperforms the fixed-support iterative LLM on 18/24 categories at evaluation 10 and 23/24 at evaluation 20. It outperforms the unrestricted iterative LLM on 16/24 and 20/24 categories, respectively. Per-category results (Appendix F.3) show that direct LLM optimization can be strong on individual tasks but substantially worse on others, whereas EGBO is more consistent.

The two NNLS oracles provide an empirical decomposition of the remaining error. At 10 evaluations, the selection and optimization gaps are of comparable magnitude. By 20 evaluations, the residual optimization gap is smaller than the selection gap, indicating that longer runs increasingly become limited by the expert-selected support. Nevertheless, adaptive expert querying ($Q = 2$), in which the expert may expand the active set after evaluation 10, does not improve performance on average over one-shot EGBO (Appendix F.4).

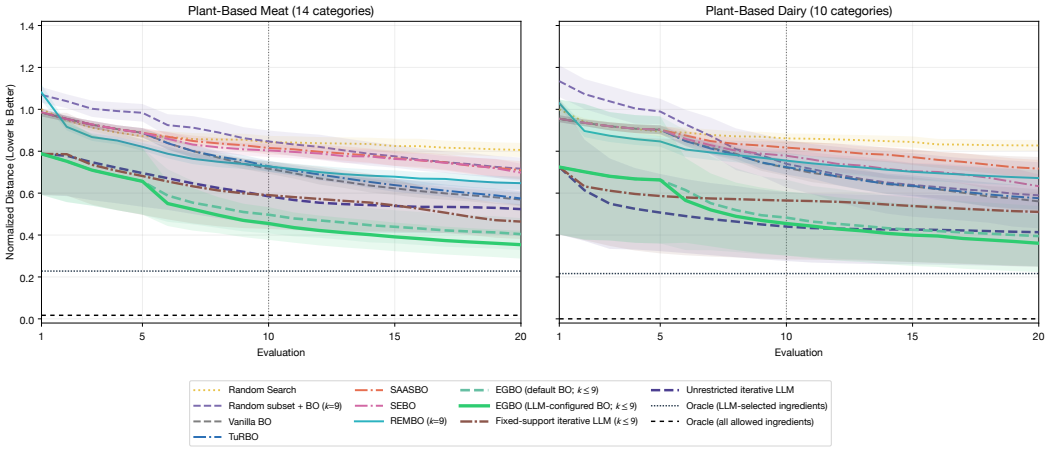

Figure 9: FORMULATEBENCH convergence for 14 meat and 10 dairy categories. Lower is better. Curves show the mean best-so-far normalized distance across categories, and shaded regions show 95% confidence intervals across category-level mean trajectories. The two EGBO curves use the same LLM-selected ingredients and initialization and differ only in whether BO uses native BoTorch defaults or the single global LLM-selected configuration. EGBO and the conventional optimization baselines are averaged over 10 optimization seeds; each temperature-zero iterative LLM baseline contributes one cached trajectory per category. The horizontal lines show constrained NNLS oracles using either the LLM-selected ingredients or all allowed ingredients. Results per category are shown in Appendix F.

F.3 Per-category nutrition results

Figures 10 and 11 show results in each category of meat and dairy.

F.4 Adaptive vs. one-shot EGBO

Setup. We compare one-shot EGBO ($Q = 1$) with adaptive EGBO ($Q = 2$) under the same information regime as the main experiment. Both methods begin with the shared LLM recipe. After evaluation 10, the adaptive expert observes the evaluated formulations and scalar target-dissimilarity scores and may add up to two ingredients without removing existing ingredients. Both methods use $T = 20$, $n_{\text{init}} = 5$, and 10 BO seeds per category.

Results. Adaptive expansion does not improve performance under the fixed $T = 20$ evaluation budget. On meat, the mean normalized distance at evaluation 20 is 0.36 for adaptive EGBO and 0.35 for one-shot EGBO. Adaptive EGBO performs better on 4/14 categories. On dairy, the corresponding values are 0.37 and 0.36, with adaptive EGBO performing better on 2/10 categories. The expert adds an average of 1.81 ingredients for meat and 1.56 for dairy. Thus, expanding the active set increases the BO dimension without producing a reliable reduction in the selection gap at this budget.

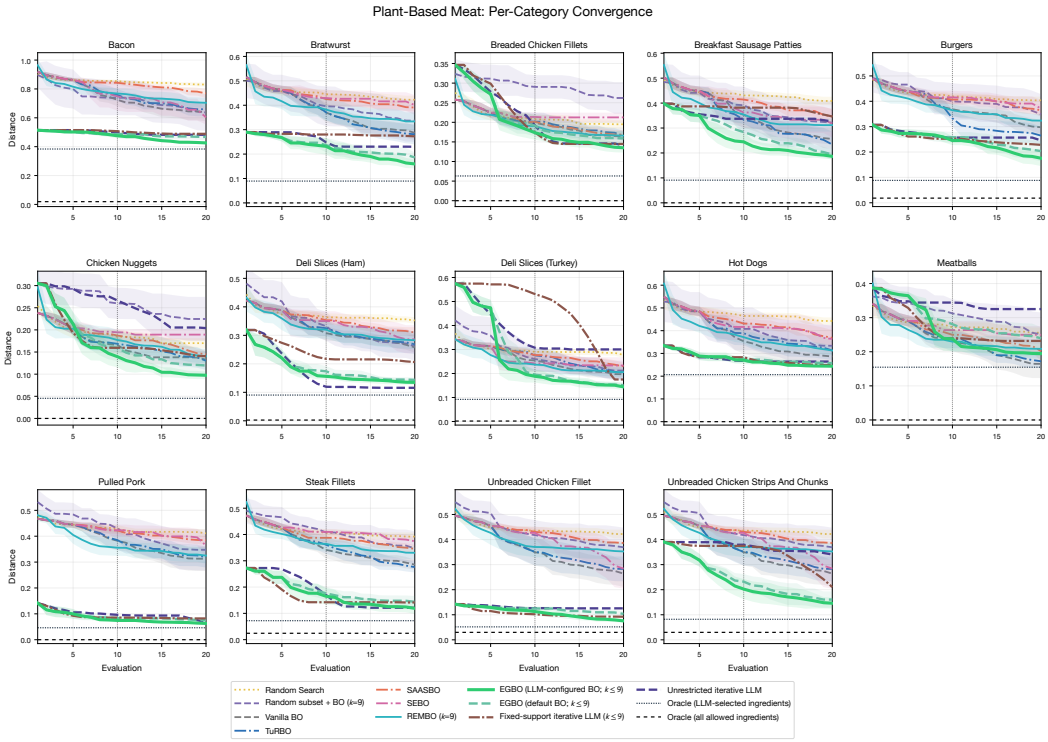

Figure 10: FORMULATEBENCH per-category convergence for the 14 meat categories. Lower is better. Curves show the mean best-so-far objective distance. The two EGBO variants share the same expert-selected ingredients and initialization and differ only in BO configuration. For EGBO and the conventional optimization baselines, shaded regions show 95% confidence intervals across 10 optimization seeds; each temperature-zero iterative LLM baseline contributes one cached trajectory per category. The horizontal lines show constrained NNLS oracles using either the LLM-selected ingredients or all allowed ingredients.

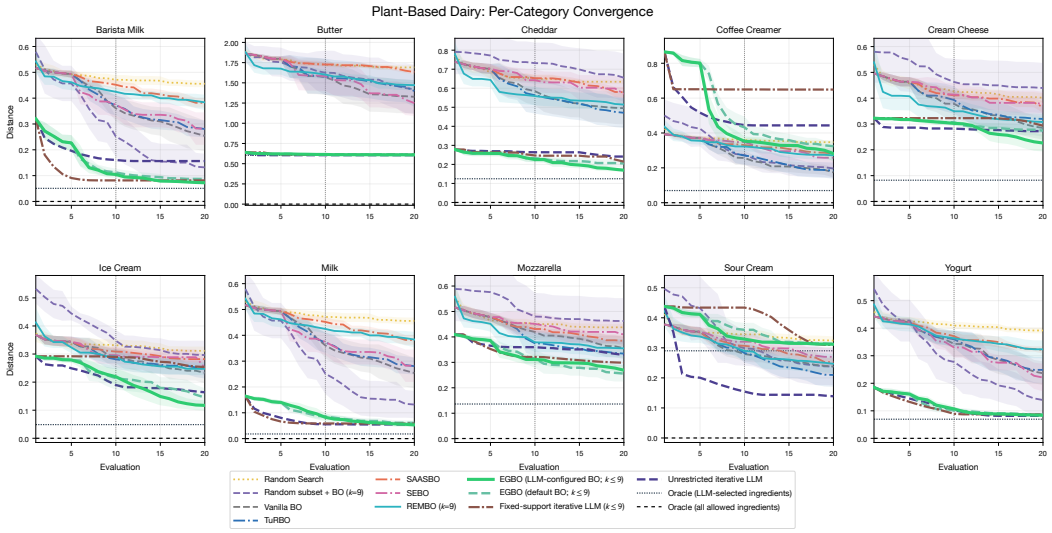

Figure 11: FORMULATEBENCH per-category convergence for the 10 dairy categories. Lower is better. Curves show the mean best-so-far objective distance. The two EGBO variants share the same expert-selected ingredients and initialization and differ only in BO configuration. For EGBO and the conventional optimization baselines, shaded regions show 95% confidence intervals across 10 optimization seeds; each temperature-zero iterative LLM baseline contributes one cached trajectory per category. The horizontal lines show constrained NNLS oracles using either the LLM-selected ingredients or all allowed ingredients.

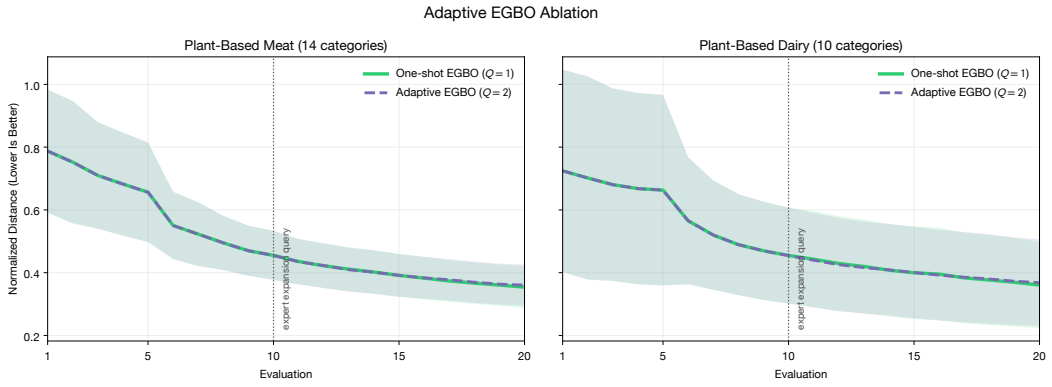

Figure 12: One-shot ($Q = 1$) and adaptive ($Q = 2$) EGBO on FORMULATEBENCH. The adaptive expert observes evaluated recipes and scalar target-dissimilarity feedback after evaluation 10 and may add up to two ingredients without removing existing ingredients. Lines average category-level means over 10 BO seeds; bands show 95% confidence intervals across categories. Adaptive expansion does not improve performance under the fixed $T = 20$ budget.

F.5 When does each strategy win?

We group the methods into three families: direct LLM optimization (the better of the fixed-support and unrestricted iterative LLM in each category), full-space BO (the better of vanilla BO and BO over all ingredients warm-started from the LLM's first recipe, which uses a Matérn-$5/2$ kernel with a long-lengthscale prior and isotropic lengthscales), and one-shot EGBO with LLM-configured BO. Losses are divided by the category's mean random-search loss at evaluation 1, and per-category values average over seeds. Figure 13(a) counts the categories in which each family has the lowest loss at each evaluation. To characterize the LLM's knowledge without new experiments, we use the two NNLS oracles: the *concentration error* is the normalized gap between the loss of the LLM's first recipe and $\mathrm{Oracle}_{S_{\mathrm{LLM}}}$, and the *ingredient-choice error* is the normalized gap between $\mathrm{Oracle}_{S_{\mathrm{LLM}}}$ and $\mathrm{Oracle}_{\mathrm{full}}$. For every LLM tested, most of the first recipe's error is concentration error: 0.54 versus 0.21 for ingredient choice with Claude Opus 4.7, 0.39 versus 0.18 with GPT-6.1 Sol, and 0.71 versus 0.17 with Gemini 4 Argon (Appendix F.8). Both correlations in Figure 13(b,c) persist when the most extreme category is removed ($\rho = 0.53$ and -0.45 without sour cream; 0.48 and -0.56 without coffee creamer) and within each domain (meat: 0.35 and -0.53; dairy: 0.64 and -0.75). These are correlations across 24 categories with uncorrected tests. With GPT-6.1 Sol and Gemini 4 Argon as the LLM (Appendix F.8), the ingredient-choice association replicates ($\rho = -0.52$ and -0.54), whereas the concentration association is weaker and not significant. Two caveats apply. First, EGBO spends evaluations 2–5 on local perturbations of the LLM recipe rather than on BO, which contributes to its lag at small budgets. Second, in coffee creamer, whose first recipe is far from optimal, full-space BO outperforms EGBO, suggesting that local initialization around a poor recipe can hurt. The analysis is reproduced from the released results by `formulateBench/analyze_regimes.py`.

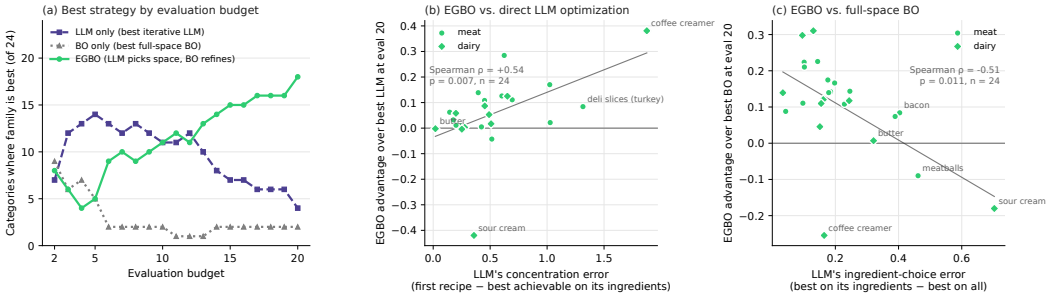

Figure 13: Which strategy wins on FORMULATEBENCH. **(a)** Number of categories (of 24) in which each family has the lowest normalized loss at each evaluation; evaluation 1 is omitted because all informed methods share the LLM's first recipe. **(b)** EGBO's advantage over the best iterative LLM arm at evaluation 20 against the LLM's concentration error. **(c)** EGBO's advantage over the best full-space BO arm at evaluation 20 against the LLM's ingredient-choice error. Positive advantages favor EGBO; lines are least-squares fits.

F.6 Observation noise

Sensory evaluations are noisy, whereas FORMULATEBENCH is deterministic. To test how noise affects each strategy, we rerun the two iterative LLM baselines and one-shot EGBO (LLM-configured BO) with additive Gaussian observation noise, $y_{\mathrm{obs}} = y + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where σ is 0.05 or 0.15 times the category's mean random-search loss at evaluation 1. All other settings match the main protocol, including the shared first recipe and EGBO's cached ingredient set. EGBO uses 10 seeds; each LLM baseline uses three independent noise draws per category at temperature zero. Because the best observed noisy score is biased toward lucky draws, we score every method by the true (noiseless) loss of the recipe it would recommend, defined as the recipe with the best observed score so far.

Under noise, direct LLM optimization nearly stops improving after evaluation 5. From evaluation 5 to 20, the unrestricted LLM improves by 0.139, 0.052, and 0.000 at $\sigma = 0$, 0.05, and 0.15, whereas EGBO improves by 0.302, 0.262, and 0.199 (Table 5, Figure 14). This is a failure of search, not only of selection. At $\sigma = 0.15$, the best true loss among all recipes the unrestricted LLM evaluates improves only from 0.644 to 0.637 between evaluations 5 and 20. EGBO outperforms the best LLM arm at evaluation 20 at every noise level (Wilcoxon signed-rank $p \leq 0.001$), and

	Evaluation 5			Evaluation 20			EGBO beats
Noise σ	Unrestr.	Fixed	EGBO	Unrestr.	Fixed	EGBO	best LLM
0	0.617	0.642	0.659	0.478	0.483	**0.357**	20/24
0.05	0.633	0.660	0.667	0.581	0.619	**0.404**	21/24
0.15	0.688	0.750	0.698	0.688	0.748	**0.499**	19/24

Table 5: True loss of the recommended recipe under observation noise (normalized; lower is better; mean over 24 categories) for the unrestricted and fixed-support iterative LLM and one-shot EGBO. The last column counts categories (of 24) in which EGBO beats the better LLM arm at evaluation 20. σ is a multiple of each category's initial random-search loss; $\sigma = 0$ is the main-text setting.

its deficit at evaluation 5 shrinks from -0.086 to -0.023 as noise increases (it narrows in 21/24 categories, paired $p = 10^{-4}$). At $\sigma = 0.15$, the two are statistically tied at evaluation 5. The mean advantage at evaluation 20 grows from 0.06 to 0.18, but this increase is concentrated in a subset of categories (it increases in 14/24, $p = 0.053$). Our recommendation rule is conservative for EGBO. At $\sigma = 0.15$, the best true recipe EGBO evaluates has loss 0.429, but the recipe it recommends has 0.499. Recommending by GP posterior mean narrows this slightly, to 0.489. Scoring each method by its own pick instead (the GP posterior-mean minimizer for EGBO, and for the LLM the recipe it states it would recommend) leaves the comparison unchanged, with EGBO better in 20/24 categories at evaluation 20. The sweep is reproduced by `formulateBench/run_noise_sweep.py` and analyzed by `formulateBench/analyze_noise_sweep.py`; cached LLM responses are released.

Telling the LLM the noise level. In a further arm, the iterative LLM's feedback ends with a note that scores include Gaussian measurement noise, stating its exact standard deviation ($\sigma = 0.15$). The note did not help. With Claude Opus 4.7, the better LLM arm's mean distance at evaluation 20 was 0.70 with the note versus 0.68 without (EGBO: 0.50), and the LLM tied or beat EGBO in 6/24 categories versus 7/24. With GPT-6.1 Sol, the values were 0.575 versus 0.568 (EGBO: 0.50), and 4/24 versus 6/24. Both comparisons use the first noise draw of each LLM arm (`formulateBench/analyze_noise_aware.py`).

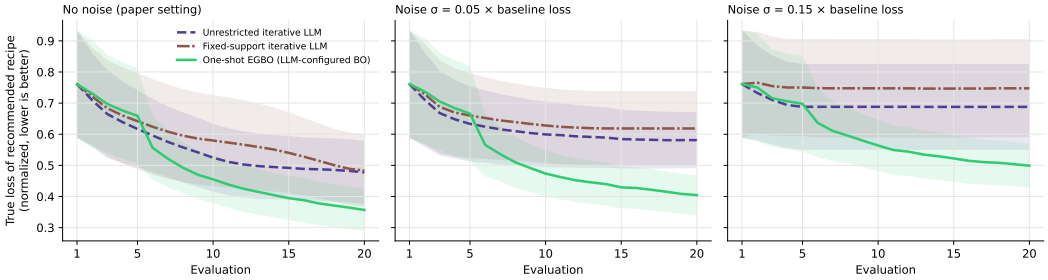

Figure 14: FORMULATEBENCH under observation noise. True loss of the recommended recipe (normalized; lower is better) for the two iterative LLM baselines and one-shot EGBO, without noise (left) and with Gaussian observation noise of standard deviation 0.05 (center) and 0.15 (right) times the category's initial random-search loss. Curves are means over 24 categories (EGBO: 10 seeds; LLM: three noise draws per category). Shaded bands are 95% CIs across categories and include between-category variation; the paired comparisons in the text are more informative.

F.7 Batched experiments

To mirror the laboratory deployment, which prepares and evaluates three recipes per session, we run 10 batches of 3 recipes (30 evaluations). For a given LLM, the first batch is identical for all LLM-based arms and consists of the LLM's first recipe and two seeded logit-normal perturbations of it ($\sigma = 0.35$, as in the local initialization of Appendix F.1). In each later batch, the iterative LLM arms are given all previous formulations and scores and asked for three formulations to be evaluated in parallel (the unrestricted arm from the full catalog with at most nine ingredients, the fixed-support arm from the LLM's first-recipe ingredients). The prompt is otherwise that of the one-at-a-time

protocol. EGBO fits the LLM-configured GP to all observations on the first-recipe ingredients and proposes three recipes with qLogNEI ($q = 3$, sequential greedy optimization). The conventional baselines do not use the LLM and search the full catalog. Random search draws three Dirichlet(1) recipes per batch. Vanilla BO starts from three simplex-initial points and then proposes batches of three with a BoTorch-default GP and qLogNEI. TuRBO does the same within a trust region (success tolerance 3, failure tolerance $\max(4, \lceil d/3 \rceil)$). Under noise, each evaluation returns the true distance plus Gaussian noise with standard deviation 0.15 times the category normalizer, and methods are scored by the true distance of the recipe with the best observed score. EGBO and the baselines use 10 seeds per category; each iterative LLM arm uses one temperature-zero trajectory. Batch diversity is the mean pairwise L_1 distance between the three recipes of a batch, averaged over batches 2–10. Table 6 summarizes the results after 30 evaluations. With Claude Opus 4.7, EGBO outperforms vanilla BO and TuRBO on 20/24 categories each without noise, and on 21/24 and 22/24 with noise. The LLM arms' batches are near-duplicates (mean pairwise L_1 distance 0.04–0.08), whereas EGBO's are spread out (0.43–0.47).

Table 6: Batched FORMULATEBENCH (10 batches of 3 recipes) after 30 evaluations: mean normalized distance over 24 categories (lower is better). "LLM" is the better of the two iterative LLM arms per category; "EGBO better" counts categories in which EGBO has the lower distance, with a two-sided Wilcoxon signed-rank p-value. Vanilla BO and TuRBO do not use the LLM.

Noise	LLM	LLM	EGBO	EGBO better	Vanilla BO	TuRBO
None	Claude Opus 4.7	0.44	0.34	19/24 ($p < 0.001$)	0.52	0.51
	GPT-6.1 Sol	0.31	0.32	9/24 ($p = 0.36$)		
	Gemini 4 Argon	0.32	0.35	10/24 ($p = 0.90$)		
$\sigma = 0.15$	Claude Opus 4.7	0.68	0.48	19/24 ($p < 0.001$)	0.70	0.73
	GPT-6.1 Sol	0.56	0.47	18/24 ($p < 0.001$)		
	Gemini 4 Argon	0.58	0.50	15/24 ($p = 0.12$)		

F.8 Other LLMs, one recipe at a time

We repeated the comparison with two other LLMs accessed through OpenRouter, GPT-6.1 Sol and Gemini 4 Argon. Each LLM proposes its own first recipe, which fixes EGBO's ingredient set. Otherwise, the protocol, prompts, and BO configuration are those of the main comparison (one LLM trajectory per category for each iterative LLM arm, ten EGBO seeds). Under observation noise ($\sigma = 0.15$ times the category normalizer, Appendix F.6), each method is scored by the true distance of the recipe with the best observed score. Table 7 reports results at evaluation 20 of the one-recipe-at-a-time protocol (Appendix F.2); batched results for GPT-6.1 Sol and Gemini 4 Argon are in Table 6. EGBO's final distance varies little across LLMs, whereas direct LLM optimization varies more. Without noise, GPT-6.1 Sol and Gemini 4 Argon match EGBO (differences not significant). Under noise, direct LLM optimization improves little after evaluation 5 for every LLM tested, and EGBO is better for Claude Opus 4.7 and GPT-6.1 Sol but not for Gemini 4 Argon, whose first proposals were already strong. Letting each LLM choose its own BO configuration, as Claude Opus 4.7 does in the main comparison, changed EGBO's mean final distance by at most 0.02. The ingredient-choice association of Appendix F.5 replicates with every LLM (Spearman $\rho = -0.52$ and -0.54 for GPT-6.1 Sol and Gemini 4 Argon, both $p < 0.01$). The concentration association, in contrast, is weaker and not significant ($\rho = 0.21$ and 0.31).

F.9 Do tool-using agents delegate?

The iterative LLM baselines propose recipes directly and have no tools. We tested whether an agent with tools would choose, on its own, to model its experiments statistically and thereby delegate refinement to an optimizer. We gave three LLMs (Claude Opus 4.7, GPT-6.1 Sol, and Gemini 4 Argon) two tools: `evaluate`, which submits one formulation (at most nine catalog ingredients) and returns its noisy score, and `python`, which runs code in a sandbox with no network access and no access to the benchmark code. The evaluated history is always available in the sandbox as a file. The task prompt is identical to the unrestricted iterative LLM's. Only its final paragraph differs, to state the budget (20 evaluations, including the LLM's first recipe) and describe the tools, including that "numpy, scipy, scikit-learn and pandas are installed." Nothing in the prompt mentions

Table 7: One recipe at a time: direct LLM optimization (better of the two iterative LLM arms) vs. EGBO at evaluation 20 across LLMs. Mean over 24 categories of the normalized distance, taking the better LLM arm per category (lower is better); "EGBO better" counts categories in which EGBO has the lower distance, with a two-sided Wilcoxon signed-rank p-value. Claude Opus 4.7 is the main comparison; under noise, its iterative LLM arms are averaged over three noise draws.

	No noise			Noise ($\sigma = 0.15$)		
LLM	LLM	EGBO	EGBO better	LLM	EGBO	EGBO better
Claude Opus 4.7	0.42	0.36	20/24 ($p < 0.001$)	0.68	0.50	19/24 ($p < 0.001$)
GPT-6.1 Sol	0.37	0.37	13/24 ($p = 0.55$)	0.57	0.50	18/24 ($p < 0.001$)
Gemini 4 Argon	0.34	0.39	9/24 ($p = 0.37$)	0.55	0.55	9/24 ($p = 0.29$)

Bayesian optimization, surrogate models, or noise. We ran 24 categories at noise level $\sigma = 0.15$ (Appendix F.6), one run per category, in a minimal tool-calling loop rather than a production agent harness, scored by the true distance of the best observed recipe. A second arm appends one sentence: "Measurements are noisy and each evaluation is expensive, so consider carefully how best to choose each experiment." For GPT-6.1 Sol, a third arm instead names BoTorch among the installed libraries. We classified each run from the code the agent executed (no code, analysis only, a fitted model such as a regression, or a Gaussian process) and checked the classification against 22 transcripts read in full.

Table 8: Tool-using agents on FORMULATEBENCH with observation noise ($\sigma = 0.15$; 24 categories, one run each). Columns count runs that wrote no code, fit any model of the data, or fit a Gaussian process, with the median evaluation (rounded) at which the first Gaussian process was fit; no run imported BoTorch. Distances are mean true normalized distances at evaluation 20 (lower is better). "No tools" is the unrestricted iterative LLM with the same prompt; EGBO uses the same first recipe.

LLM	Prompt	No code	Any model	GP	First GP	Agent	No tools	EGBO
Claude Opus 4.7	base	0	0	0	—	0.58	0.70	0.50
	+ hint	0	0	0	—	0.59		
GPT-6.1 Sol	base	20	4	1	9	0.48	0.58	0.50
	+ hint	15	7	4	12	0.46		
	+ BoTorch	12	10	6	13	0.51		
Gemini 4 Argon	base	0	24	20	13	0.66	0.57	0.55
	+ hint	1	23	22	10	0.66		

Table 8 summarizes the results. No agent ran a BO loop over its remaining budget. Claude Opus 4.7 wrote code in every run, but only to tabulate past results. It never fit a model, and under noise it anchored on the recipe with the luckiest score, re-evaluating and perturbing it ("All perturbations fail. Iter 5 remains best at 0.0236"). Under the base prompt, GPT-6.1 Sol usually wrote no code. When it fit a Gaussian process, it was competently specified (scaled inputs, a noise kernel, and a mean-minus-uncertainty selection rule), but it was fit once, late in the budget, to propose a single recipe, after which the agent returned to proposing recipes directly. Gemini 4 Argon fit models in nearly every run, but ad hoc, using linear regressions, line searches, and Gaussian processes whose posterior mean it minimized within narrow bounds it chose by hand. It had the worst final distance of the three agents, worse than its own iterative LLM baseline (base prompt, $p = 0.03$). The hint changed behavior only modestly, and naming BoTorch reduced the GPT-6.1 Sol runs without code from 20 to 12 but led no run to import it. The agents' final distances are not directly comparable with the iterative LLM baselines, because an agent keeps its own reasoning across turns and knows its budget. The GPT-6.1 Sol agent, for example, outperforms its no-tools baseline even in runs without code. We therefore do not attribute these gains to tool use. EGBO, in contrast, uses the same first recipe and budget, and no agent arm outperformed it significantly. The behavioral conclusion is that these agents could model their data but did not adopt modeling as a strategy.

G Real-World Deployment

G.1 Shared Setup

Problem. The candidate ingredient pool is a food-company catalog of $N \approx 88{,}000$ entries. Prior to optimization, the company's Food Scientist Agent (FSA, an LLM-based agentic framework) performs one-shot variable selection ($Q = 1$) yielding $d = 7$ active ingredients for yogurt and $d = 9$ for cottage cheese. All other $N - d$ ingredients are fixed at zero. Variables are encoded as continuous quantities with per-ingredient $[\min, \max]$ bounds taken from the company catalog. Property and quantity constraints are enforced via linear inequality constraints passed to `optimize_acqf`.

Budget and batching. Each campaign runs $T = 10$ iterations with a laboratory batch of $q = 3$ recipes per iteration, for 30 total recipes. The application draws scrambled Sobol batches from a pool of $2{,}048$ feasible candidates (Sobol seed equals the number of observations, for reproducibility) until at least $n_{\text{init}} = 5$ observations exist (see below for how each campaign started). Subsequent batches are produced by qLogNEI with `num_restarts` $= 20$, `raw_samples` $= 1024$, `sample_shape` $= [512]$, and `sequential` = True so each of the three batch members is selected greedily conditional on the previously chosen ones. The Streamlit application optionally enables an Input Warping kernel (`robust` = True) for non-stationarity; the campaigns reported here use the default unwarped configuration.

Starting recipe. The FSA's recipe was panel-scored before optimization ($f = 0.767$ for yogurt, 0.776 for cottage cheese). In the yogurt campaign, this observation seeded the application before iteration 1 and is not among the 30 recipes. Iterations 1–2 are Sobol batches and BO proposals begin at iteration 3. In the cottage cheese campaign, iteration 1 consisted of the FSA recipe (recipe 1.1) and two Sobol recipes generated by the application (recipes 1.2–1.3), all counting toward the 30-recipe budget. Iteration 2 is a Sobol batch, and BO proposals begin at iteration 3.

Utility. Sensory scores are entered through the Streamlit UI (`app.py`). `food_bo.py:_compute_utility` normalizes each attribute to $[0, 1]$ using the configured panel scale and converts it to a per-attribute utility according to the goal (max/min/target). The scalar objective is the weighted sum $\sum_i w_i u_i$ with uniform weights $w_i = 1/m$, matching the main-text utility definition. In the cottage cheese campaign the weights were entered as 0.17 each (summing to 1.02); see the footnote in Section 5.

Reproducibility. Sobol scrambling and a `torch.manual_seed` call both use `len(self.X_history)` so reruns at any iteration deterministically reproduce the proposed batch given the recorded panel scores. Campaign state is persisted to pickle and exported as JSON. We release the full optimization variables, ingredient lists, panel protocols, and per-iteration recipes alongside the code.

Compute. Each iteration's acquisition optimization runs in under a minute on a single CPU. The binding constraint is laboratory time ($\sim$4 hours per recipe batch), giving $\sim$40 hours per campaign.

G.2 Recipe Formulations

This appendix documents the output of FSA, final ingredient compositions and processing protocols for two plant-based products developed under the EGBO framework: the plant-based yogurt corresponding to the closed-loop deployment, and an analogous plant-based cottage cheese formulation. Quantities are reported as supplied to the laboratory. Ingredients held fixed during optimization are flagged as constants; all others were varied by EGBO.

G.2.1 Plant-Based Yogurt

Table 9 reports the FSA starting recipe (total batch mass 630.00 g) and the best formulation found by EGBO (iteration 7, recipe 3). Soy protein isolate and tricalcium phosphate were held constant across all iterations, serving as a functional protein backbone and a calcium fortificant, respectively. The remaining seven ingredients constitute the active EGBO design space.

The yogurt was prepared according to the following three-step protocol.

Step 1: Powder dispersion. All ingredients listed in Table 9, including the calcium fortificant, were combined and dissolved using a high-shear mixer at $3000\ \mathrm{RPM}$, $25\ ^\circ\mathrm{C}$, for $5\ \mathrm{min}$.

Step 2: Cooking. The dispersed yogurt base was cooked using a moist-heat boiling protocol in a covered pot (cover closure 90%) at $90\ ^\circ\mathrm{C}$ for $8\ \mathrm{min}$.

Step 3: Chilling. The cooked product was transferred to a refrigerator and chilled prior to sensory evaluation.

G.2.2 Plant-Based Cottage Cheese

Table 11 reports the ingredient composition of the plant-based cottage cheese formulation. The first two ingredients form the curd-forming substream, which is subsequently centrifuged to remove whey. The remaining seven ingredients form the cooked dressing, which is combined with the rinsed curds. The table reports the FSA starting recipe (recipe 1.1) and the best recipe found by EGBO (recipe 10.1). Weight fractions are reported as supplied to the process.

The cottage cheese was prepared according to the following seven-step protocol.

Step 1: Pasteurization. Soy milk was heat-treated for microbial-load reduction using a batch pasteurizer at $90\ ^\circ\mathrm{C}$ for $5\ \mathrm{min}$ with full cover closure (100%). Output: pasteurized soy milk.

Step 2: Acid coagulation. Pasteurized soy milk was combined with distilled vinegar in a mixing vessel operated at $200\ \mathrm{RPM}$, $75\ ^\circ\mathrm{C}$, for $2\ \mathrm{min}$ with 80% cover closure. Output: coagulated soy mixture.

Step 3: Curd separation. The coagulated mixture was separated by basket centrifugation at $800\ \mathrm{RPM}$ (50 Hz drive frequency), $25\ ^\circ\mathrm{C}$, for $3\ \mathrm{min}$ in a single pass. Outputs: soy curds; soy whey (discarded).

Step 4: Curd rinsing. Soy curds were washed with tap water in a washing machine at $10\ ^\circ\mathrm{C}$ for $1\ \mathrm{min}$ (single pass). Outputs: rinsed soy curds; spent rinse water.

Step 5: Dressing cooking. Coconut cream, tap water, lemon juice, salt, nutritional yeast, tapioca starch, and xanthan gum were combined and cooked using a moist-heat boiling protocol in a covered pot (cover closure 90%) at $90\ ^\circ\mathrm{C}$ for $8\ \mathrm{min}$. Output: cooked plant-based dressing.

Step 6: Combination. Rinsed soy curds and cooked dressing were combined in a planetary mixer at $60\ \mathrm{RPM}$, $20\ ^\circ\mathrm{C}$, for $2\ \mathrm{min}$. Output: unchilled plant-based cottage cheese.

Step 7: Chilling. The product was cooled to $4\ ^\circ\mathrm{C}$ in a refrigerator. Output: plant-based cottage cheese (final).

G.3 Sensory evaluation protocol

The utility scores reported in Section 5 were obtained via a structured descriptive sensory evaluation conducted by a trained panel of five participants. The procedure was held fixed across all $T = 10$ EGBO iterations to ensure that observed utility differences reflect genuine recipe-level differences. The sensory panel is treated as a noisy black-box oracle, where for a formulation $x \in \mathbb{R}^N$, the observed score is $y = f(x) + \varepsilon$, with ε capturing inter-panelist and session variability.

Panel composition and preparation. All panelists were members of a trained sensory panel with prior experience in structured attribute-based evaluation, and each session was preceded by a calibration briefing aligning the panel on the target attributes and on the 0–15 intensity scale. To ensure a neutral palate at the time of evaluation, panelists abstained from food consumption for at least two hours before the session, and from strong-flavored stimuli (spicy or acidic foods, alcohol, coffee, smoking) for at least six hours before the session.

Evaluation environment. Sessions were conducted in a dedicated sensory evaluation room designed to minimize external bias. The space was isolated from ambient odors and distractions, and environmental conditions (temperature, noise level, airflow) were held stable across sessions. Controlled lighting was applied to suppress visual bias on attributes such as color and surface appearance. Panelists worked in individual booths, did not communicate during the session, and did not discuss or compare samples. Each panelist evaluated samples in a predefined sequence determined prior to the session.

Calibration system. A Golden Standard reference sample was made available throughout each session as a continuous calibration anchor. Panelists were instructed to revisit the reference between samples to maintain a stable internal reference point, and printed 0–15 attribute scales were provided before the start of evaluation. This combination of a physical reference and an explicit scale was used to control for within-session and between-session drift in panelist perception.

Evaluation procedure. The protocol followed a comparative descriptive analysis approach: attribute-based, comparative against the Golden Standard, and non-hedonic (preference and liking are not recorded). Each sample was assessed via a fixed sequence of stages, each applied where relevant to the product class:

- **Visual assessment.** Density, surface characteristics, structural integrity, and overall first impression.
- **Aroma.** Identification of intensity and key aromatic notes under controlled exposure to avoid sensory fatigue.
- **Texture and mouthfeel.** Initial perception on the first spoon or bite, creaminess and slipperiness, graininess, and cohesion or structural behavior under deformation.
- **Flavor.** Basic taste profile (sweet, sour, salty, bitter, umami), flavor identity (e.g., dairy, fermented, clean, off-notes), balance, and aftertaste.
- **Overall assessment.** Holistic evaluation and identification of key strengths and weaknesses.

For the plant-based yogurt campaign, the four target attributes (Consistency, Creaminess, Tanginess, Uniformity) were drawn from the Texture and Flavor stages above and recorded on the 0–15 scale. For plant-based cottage the same was applied for the six target attributes (Homogeneity, Dispersion Phase, Slipperiness, Sourness, Dairy and Off-Notes).

Between-sample protocol. To preserve sensory clarity across samples, panelists cleansed their palate with water between evaluations, with neutral carriers (bread and crackers) available as an option, and observed a short reset period before proceeding to the next sample. The Golden Standard was re-tasted as needed during this interval to realign perception.

Objectivity, fatigue management, and performance principle. The panel operated under strict objectivity principles. Evaluation was based on the defined sensory attributes rather than personal preference, no response was treated as right or wrong, and consistency across samples was prioritized over individual bias. When sensory fatigue or attribute confusion arose, panelists followed a four-step recovery protocol: pausing evaluation, cleansing the palate, re-evaluating the Golden Standard, and then resuming testing. The overall protocol prioritized accuracy, consistency, and repeatability of the recorded scores; speed of evaluation was treated as secondary to data quality.

Data aggregation. Panelist scores were aggregated immediately after evaluation into a single set of attribute-level scores. Individual panelist responses were not retained, and no individual-level data were used in subsequent analysis. The utility values reported in the main text were computed from these aggregated scores using the scalar utility function defined in Section 5.

G.4 Yogurt campaign

Sensory convergence. Figure 15 shows the trajectory of all four sensory attributes. Consistency rose from 5 at baseline to 12 (exact target) by iteration 4, and remained within ± 1 of target for all but one subsequent recipe (13.5 for recipe 9.3). Creaminess increased from 3 to 5 (exact target), first

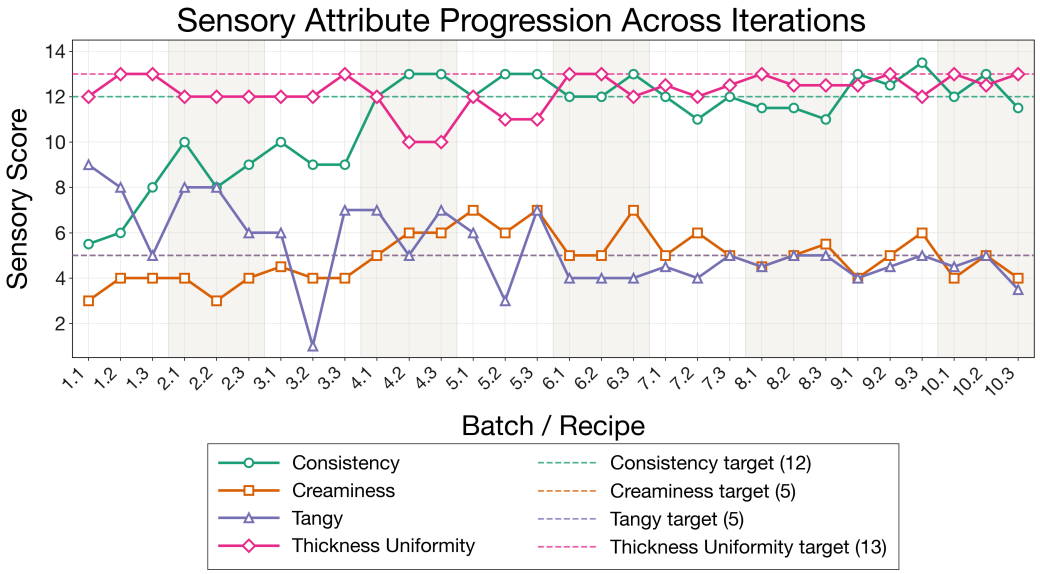

Figure 15: Sensory attribute trajectories across the 10 EGBO iterations (3 recipes per iteration) for the plant-based yogurt campaign. Solid lines show observed panel scores; dashed lines indicate the target value for each attribute. Consistency and Creaminess converge to their targets by iteration 7; Tanginess exhibits the highest volatility before settling at target; Uniformity remains near target throughout with minor drift.

reaching it at iteration 4 and stabilizing from iteration 7 onward. Tanginess, which required a decrease from 10 to 5, proved the most volatile attribute, dropping sharply in early iterations but oscillating between 3 and 7 through the middle of the campaign before converging to 5 at iteration 7. Uniformity, which started at 13 (on target), drifted to values between 10 and 13 during the exploration-heavy early iterations but recovered to 12.5–13 in the final batches. The best single recipe (iteration 7, recipe 3) achieved scores of $(12,\ 5,\ 5,\ 12.5)$, matching the target exactly on three of four attributes and falling only 0.5 points short on Uniformity.

Utility progression. Figure 16 reports the maximum utility per iteration and the cumulative best-so-far. The utility climbed steeply in the first four iterations, reaching 0.950 by iteration 4. A second jump to 0.983 occurred at iteration 6, and the campaign peak of 0.992 was reached at iteration 7. The best-so-far remained at or above 0.983 from iteration 6 onward, indicating that the optimizer had largely converged by the sixth batch and subsequent iterations refined rather than explored. The residual gap to perfect utility is $1 - 0.992 = 0.008$.

Ingredient concentration trajectories. Figure 17 shows how the seven active ingredients evolve over the campaign. Tap water and tapioca starch repeatedly take the same maximum values (42.0% and 2.6% of the recorded formulation; Table 9 rescales the best recipe to a 630 g batch) in most recipes from iteration 4 onward, consistent with binding upper bounds. The optimum may therefore lie outside the campaign's box, and widening these bounds is a natural next step.

Final formulation. Table 9 reports the FSA starting recipe and the best formulation found by EGBO (iteration 7, recipe 3). The formulation and preparation protocol are described in Appendix G.2.1.

Iteration photos. Figures 18–27 show the visual evolution of the yogurt across iterations.

G.5 Cottage cheese campaign

Sensory convergence. Figure 28 shows the trajectory of all six sensory attributes. Off-Notes, which required a decrease from 7 to 0, showed the most dramatic improvement, dropping steadily from 7 to 1.5 by the final iterations. Sourness decreased from 10 toward its target of 6.5, stabilizing in the

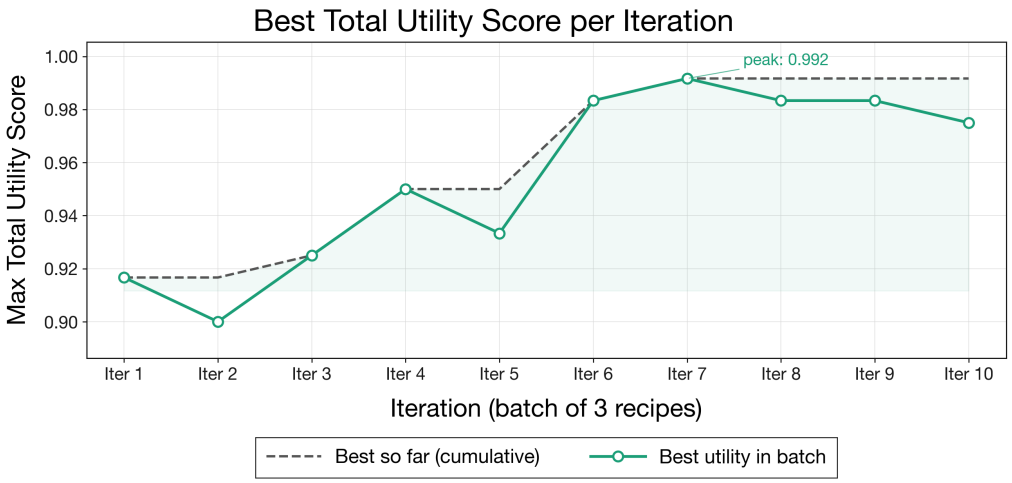

Figure 16: Best total utility score per iteration (solid) and cumulative best-so-far (dashed) across the 10 EGBO iterations. Utility rises from $f(x^{(0)}) = 0.767$ at baseline to a peak of 0.992 at iteration 7, with the best-so-far remaining at or above 0.983 from iteration 6 onward.

Table 9: Plant-based yogurt: ingredient composition of the FSA starting recipe ($f = 0.767$) and the best recipe found by EGBO (iteration 7, recipe 3; $f = 0.992$). Both recipes are scaled to a total batch mass of 630.00 g.

	Starting recipe		Best recipe		
Ingredient	Mass (g)	% w/w	Mass (g)	% w/w	Status
Coconut cream, canned, sweetened	340.00	53.97	311.92	49.50	varied
Tap water	237.75	37.74	267.27	42.42	varied
Tapioca starch	6.00	0.95	16.55	2.63	varied
Lemon juice, raw	20.00	3.17	14.05	2.23	varied
Nutritional yeast	10.00	1.59	7.18	1.14	varied
Soy protein isolate	6.30	1.00	6.30	1.00	constant
Xanthan gum	1.00	0.16	2.89	0.46	varied
Tricalcium phosphate	1.95	0.31	1.95	0.31	constant
Salt	7.00	1.11	1.89	0.30	varied
Total	630.00	100.00	630.00	100.00	

range 6–7 from iteration 7 onward. Dairy, which started at 3 against a target of 7, proved the most difficult attribute to improve, rising gradually to 5 by the final iterations but not fully reaching target (which is expected since this is a plant-based dairy recipe). Homogeneity improved from 4 to values consistently near 6 (target) in the second half of the campaign. Dispersion Phase and Slipperiness, both starting well above their targets of 6, converged toward target by iteration 8, though with residual volatility. The best single recipe (iteration 10, recipe 1) achieved a utility of 0.975, matching the target exactly on Homogeneity, Dispersion Phase and Sourness, with the largest residual gap on Dairy (5.0 vs. target 7).

Utility progression. Figure 29 reports the maximum utility per iteration and the cumulative best-so-far. The utility rose steadily from 0.776 at baseline, reaching 0.895 at iteration 4 and 0.941 at iteration 5. The best-so-far reached 0.958 at iteration 7 and continued climbing to the campaign peak of 0.975 at iteration 10, indicating that the optimizer had not fully plateaued and additional iterations could yield further gains.

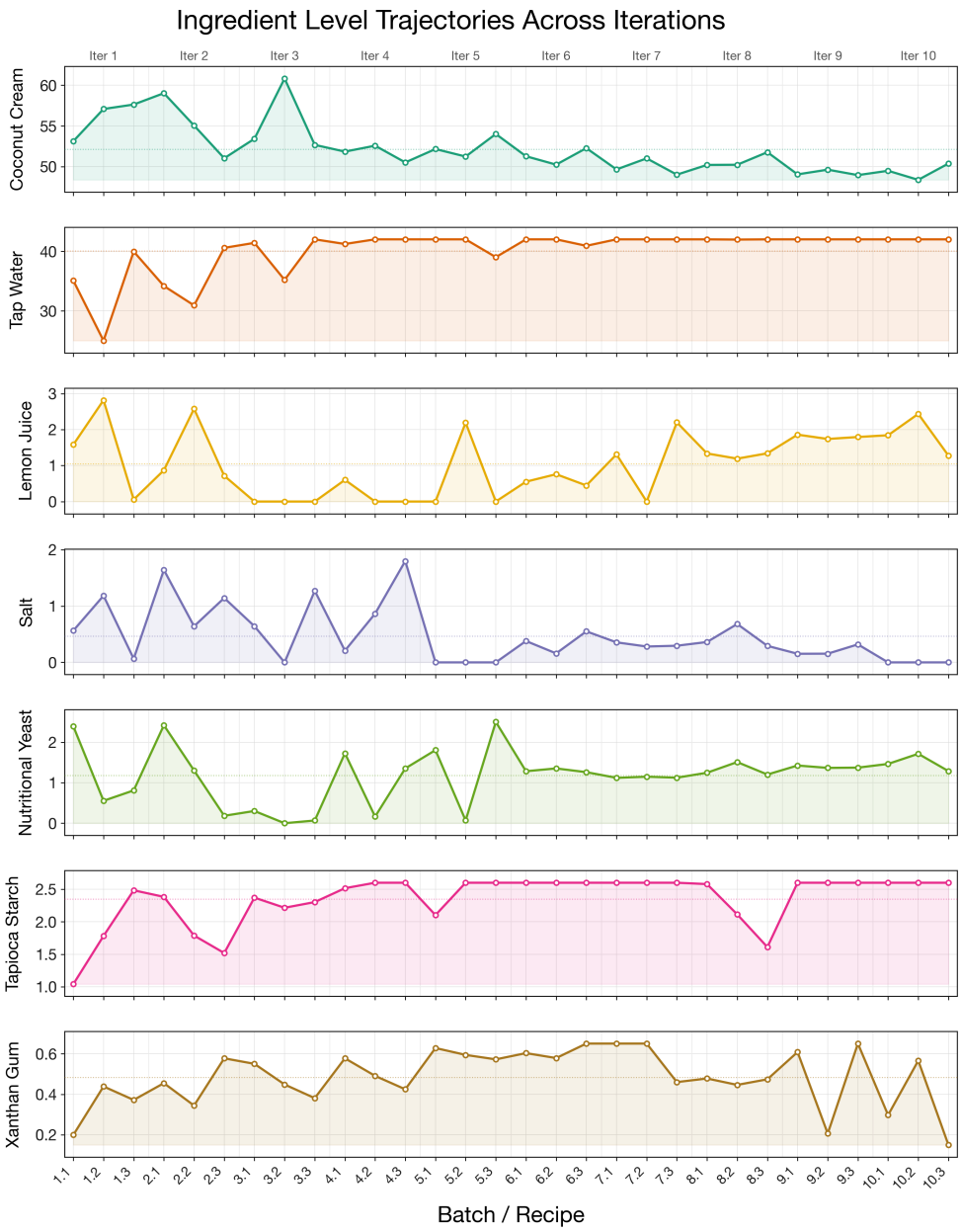

Figure 17: Ingredient concentration trajectories across the 10 EGBO iterations for the seven varying formulation dimensions. Each panel is independently scaled to reveal fine-grained adjustments: the dominant Coconut cream and Tap water rebalancing is visible alongside the smaller but consequential shifts in Lemon juice, Salt, Nutritional yeast, Tapioca starch, and Xanthan gum.

Final formulation. Table 10 reports the per-attribute errors, and Table 11 the FSA starting recipe and the best recipe found by EGBO (recipe 10.1). The formulation and preparation protocol are described in Appendix G.2.2.

G.6 Cross-campaign comparison

Figure 30 overlays the best-so-far utility curves for both campaigns. Both start from similar baselines generated by the FSA (0.767 for yogurt, 0.776 for cottage cheese) and follow a characteristic steep-then-plateau trajectory. Yogurt converges faster, reaching 0.983 by iteration 6 and peaking at 0.992



Figure 18: Plant-based yogurt: iteration 1.



Figure 19: Plant-based yogurt: iteration 2.



Figure 20: Plant-based yogurt: iteration 3.



Figure 21: Plant-based yogurt: iteration 4.



Figure 22: Plant-based yogurt: iteration 5.

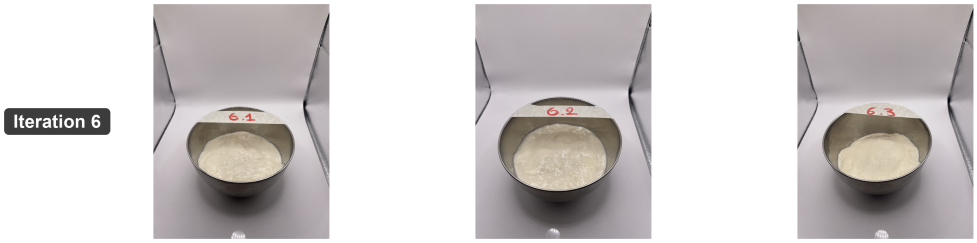

Figure 23: Plant-based yogurt: iteration 6.



Figure 24: Plant-based yogurt: iteration 7.



Figure 25: Plant-based yogurt: iteration 8.

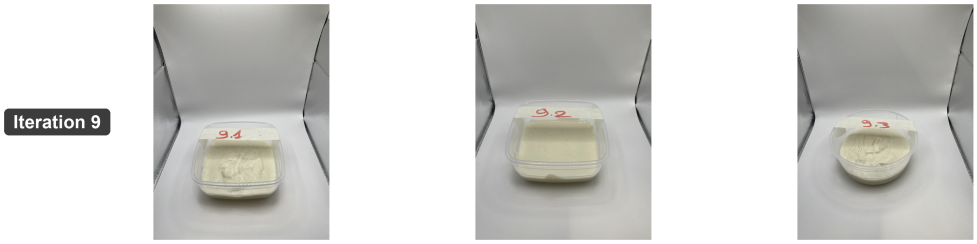

Figure 26: Plant-based yogurt: iteration 9.



Figure 27: Plant-based yogurt: iteration 10.

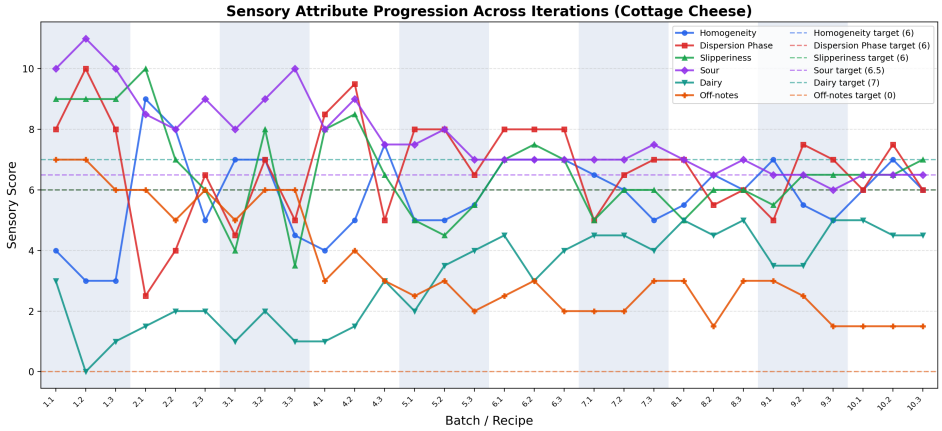

Figure 28: Sensory attribute trajectories across the 10 EGBO iterations (3 recipes per iteration, 30 total) for the plant-based cottage cheese campaign. Solid lines show observed panel scores; dashed lines indicate the target value for each attribute.

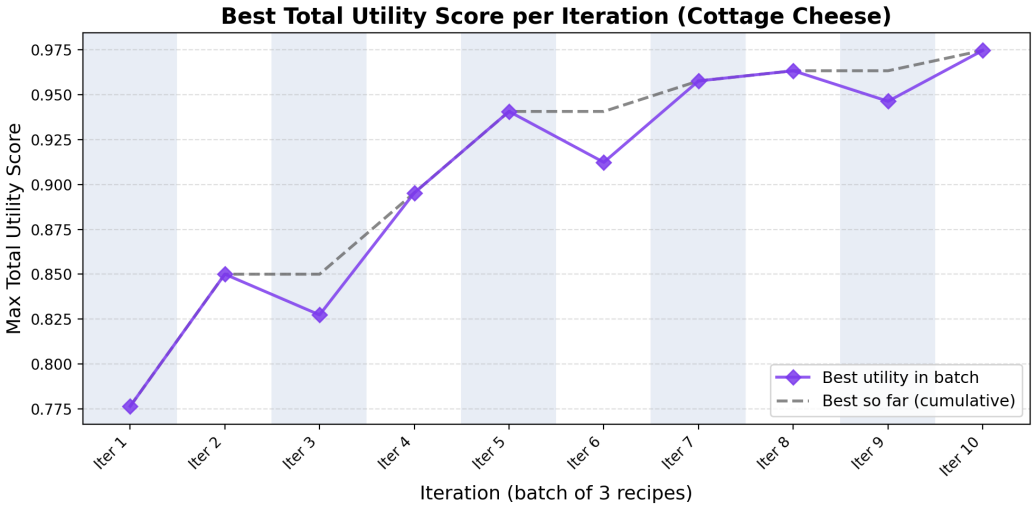

Figure 29: Best total utility score per iteration (solid) and cumulative best-so-far (dashed) across the 10 EGBO iterations for the cottage cheese campaign. Utility rises from $f(x^{(0)}) = 0.776$ at baseline to a peak of 0.975 at iteration 10.

at iteration 7, while cottage cheese reaches 0.958 by iteration 7 and peaks at 0.975 at iteration 10. The slower convergence of cottage cheese is consistent with its higher-dimensional sensory objective ($m = 6$ vs. $m = 4$) and larger active ingredient set ($d = 9$ vs. $d = 7$), both of which increase the effective optimization difficulty. Nevertheless, both campaigns achieved substantial improvements — 29% and 26% respectively — within the same 10-iteration budget, suggesting that EGBO transfers across product categories with different sensory structures and ingredient spaces, considering also that it started from a > 0.75 utility.

G.7 EGBO vs. Human Food Scientist

G.7.1 Protocol

The EGBO vs. HFS comparison (Section 5.3) was conducted under the following instructions and constraints, designed to ensure a controlled comparison while respecting the inherently different workflows of algorithmic and manual formulation.

Table 10: Plant-based cottage cheese real-world validation. The best recipe (iteration 10, recipe 1) shows the largest gains on Off-Notes and Sourness, with Dairy remaining the hardest attribute to reach. Overall utility improved 26% relative to baseline. Utilities use the weights $w_j = 0.17$ described in Section 5, so the maximum is 1.02.

Attribute	Target	Baseline	Best	Error	
				Before	After
Homogeneity	6	4	6	2	0
Dispersion Phase	6	8	6	2	0
Slipperiness	6	9	6.5	3	0.5
Sourness	6.5	10	6.5	3.5	0
Dairy	7	3	5	4	2
Off-Notes	0	7	1.5	7	1.5
Utility $f(x)$	1.020	0.776	0.975	0.244	0.045

Table 11: Plant-based cottage cheese: ingredient composition of the FSA starting recipe (recipe 1.1; $f = 0.776$) and the best recipe found by EGBO (recipe 10.1; $f = 0.975$). Curd-substream ingredients (top) are coagulated and centrifuged before being recombined with the cooked dressing (bottom).

Ingredient	Starting recipe % w/w	Best recipe % w/w	Substream
Soy milk, unsweetened, plain, shelf stable	63.12	70.00	curd
Distilled vinegar	6.00	4.10	curd
Tap water	10.38	8.00	dressing
Coconut cream, canned, sweetened	18.06	15.81	dressing
Lemon juice, raw	0.55	0.40	dressing
Nutritional yeast	0.68	0.50	dressing
Salt	0.49	0.59	dressing
Tapioca starch	0.68	0.57	dressing
Xanthan gum	0.04	0.03	dressing
Total	100.00	100.00	

HFS profile. HFS is a professional food scientist and chef with prior experience in recipe formulation but no prior experience specifically with plant-based dairy products. The operator was not given access to the EGBO system, its suggestions, or its results at any point during the experiment. HFS participated voluntarily as a professional collaborator; the comparison was designed in mutual agreement and HFS was not a research subject.

Shared constraints.

1. **Sensory targets.** HFS was given the same four sensory objectives used in the EGBO campaign, assessed on a 0 to 15 scale: Consistency (target 12), Creaminess (target 5), Tanginess (target 5), and Uniformity (target 13).
2. **Utility metric.** The scalar utility score (Equation 6) was computed for each evaluated formulation and reported to the operator after each evaluation.
3. **Plant-based constraint.** All ingredients must be plant-based; no animal-derived ingredients were permitted.
4. **Time tracking.** HFS documented the time spent on each trial or iteration, including preparation, cooling, tasting, and decision-making time.
5. **Documentation.** All ingredient quantities, preparation steps, and sensory observations were recorded for each trial.

Experiment 1: full workflow comparison.

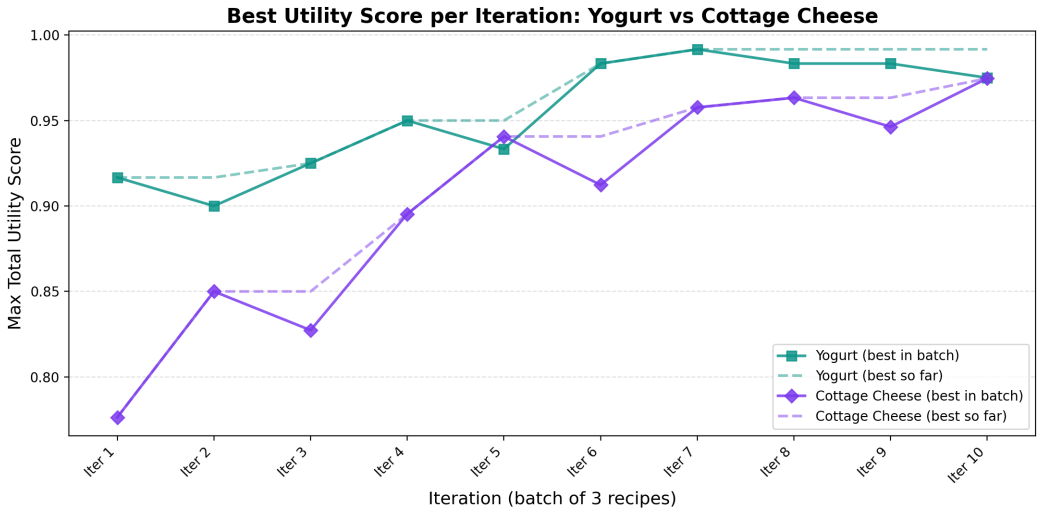

Figure 30: Comparison of best-so-far utility across both real-world campaigns. Yogurt converges faster (peaking at 0.992 by iteration 7), consistent with its lower-dimensional sensory objective (m=4, d=7). Cottage cheese converges more slowly (peaking at 0.975 at iteration 10), reflecting its higher-dimensional objective (m=6, d=9).

1. **Objective.** Develop, from scratch, a plant-based yogurt recipe that matches the target sensory profile as closely as possible.
2. **No starting recipe.** The operator was free to choose any plant-based ingredients and any preparation method.
3. **No budget constraint.** There was no limit on the number of trials. The operator was instructed to continue until satisfied with the recipe or until further progress seemed unlikely.
4. **Trial definition.** A trial is defined as a complete attempt starting from raw ingredients. If the operator modifies a formulation mid-preparation, this is considered part of the same trial. If the operator stops and starts over from the beginning, a new trial begins. Trials abandoned before producing a tasteable result were recorded as failed trials.
5. **Evaluation.** Formulations that the operator deemed sufficiently promising were evaluated for sensory scores on the four target attributes and the corresponding utility was computed and shared with the operator.
6. **Workflow.** The scientist followed a conventional iterative workflow, preparing a candidate formulation, letting it cool, tasting and evaluating it, and then deciding on the next modification or restarting.

Experiment 2: same-start refinement comparison.

1. **Objective.** Optimize the FSA's starting recipe ($f(x^{(0)}) = 0.767$) toward the same four sensory targets.
2. **Fixed starting point.** The operator was given the exact recipe (ingredient list and concentrations) produced by the FSA, the same starting recipe used by EGBO.
3. **Budget.** Maximum of 10 iterations, matching the EGBO budget.
4. **One recipe per iteration.** The scientist chose to produce one recipe per iteration, judging a batch of 3 independent variants infeasible because each formulation decision depended sequentially on the sensory evaluation of the previous one.
5. **Sensory feedback.** After each iteration, the formulation was evaluated by the sensory panel on the four target attributes, and the scores and resulting utility were reported to the operator before the next iteration began.
6. **Permitted modifications.** The operator was free to adjust any ingredient concentration, add new plant-based ingredients, or remove ingredients, subject to the plant-based constraint.

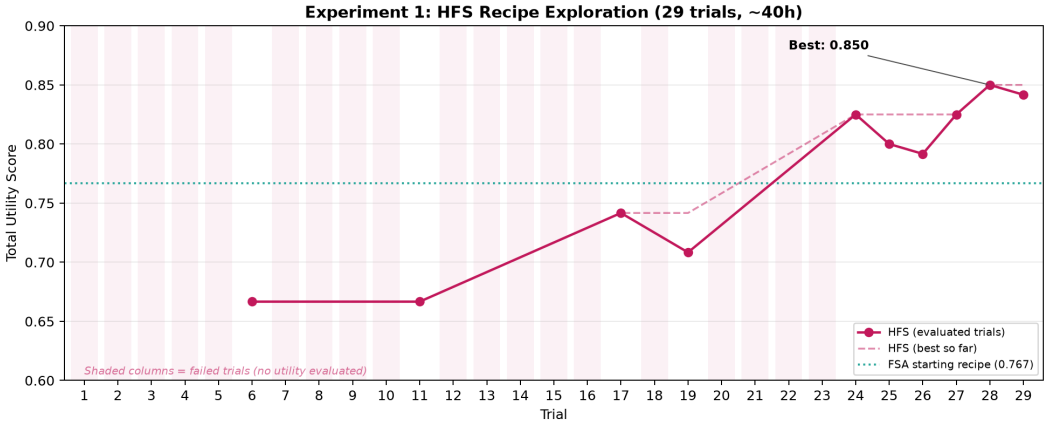

Figure 31: Experiment 1: HFS recipe exploration. The operator executed 29 trials over ∼40 h, of which 10 produced formulations evaluated for utility. The dotted line indicates the utility of the FSA's starting recipe (0.767), produced in negligible time. HFS's best result (0.850, trial 28) required the full 40-hour exploration budget.

7. **Time per iteration.** Each iteration consisted of recipe design, preparation, cooling, and sensory evaluation, requiring approximately 4 hours.

Key differences between EGBO and HFS workflows. Table 12 summarizes the structural differences between the two workflows.

Table 12: Structural differences between EGBO and HFS workflows in Experiment 2.

	EGBO	HFS
Recipes per iteration	3 (batch)	1 (sequential)
Total recipes in 10 iter	30	10
Recipe design time	Seconds	Hours
Decision basis	GP surrogate	Expert judgment
Regression possible	Unlikely	Observed

The batch vs. sequential distinction is not an artificial handicap imposed on HFS but a genuine consequence of the two workflows. EGBO's acquisition function can propose multiple diverse points in a single optimization step because it reasons over a global surrogate model. HFS, by contrast, reasons sequentially. Each modification is informed by the sensory outcome of the previous one, and designing three independent variants in parallel would require the operator to mentally maintain three separate hypotheses about ingredient interactions without feedback.

G.7.2 Results

Figures 31 and 32 show the utility trajectories for Experiments 1 and 2; Section 5.3 discusses the results.

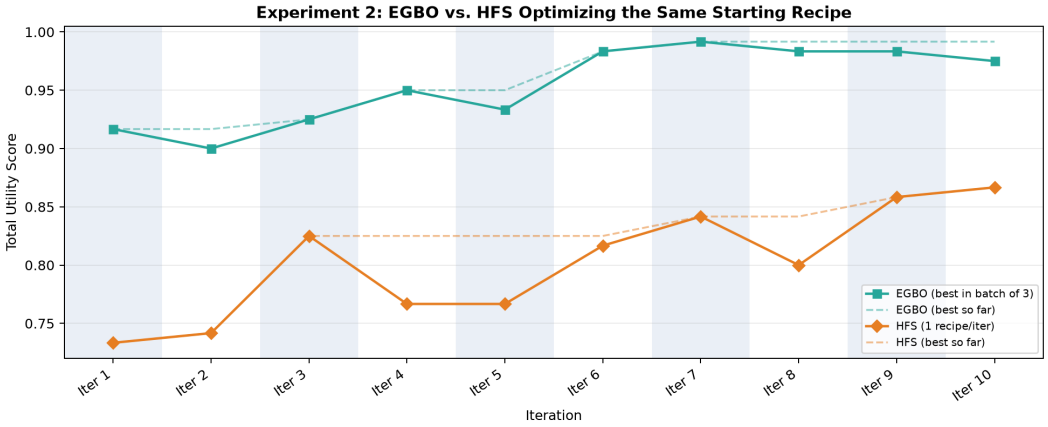

Figure 32: Experiment 2: EGBO vs. HFS, both optimizing the same starting recipe ($f(x^{(0)}) = 0.767$) over 10 iterations. EGBO proposes 3 recipes per iteration; HFS produces 1. EGBO reaches 0.992 by iteration 7, while HFS reaches 0.867 by iteration 10, a 14.4% gap.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: We make three claims. First, we formalize expert-guided Bayesian optimization, characterize its performance theoretically, and evaluate its performance under controlled expert quality on a classic test function: Hartmann6. Second, we introduce a benchmark, FORMULATEBENCH for *in silico* prototyping of formulation design algorithms, and find that, in batched experiments mirroring our laboratory deployment, EGBO outperforms conventional BO, while the strongest LLMs match it without observation noise but largely stop improving under noise; we also test whether LLM agents with code execution run BO on their own (Appendix F.9). Third, we deploy EGBO for two real-world dairy formulation projects, and find that sensory utility improves by 29% and 26% in 10 iterations. Moreover, EGBO's performance is competitive with or exceeds that of an expert food scientist. All of these claims are supported in the main text.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions are in Section 3.1, and proofs are in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All optimization code, recipes, ingredient lists, panel protocols, and per-iteration results are released in Appendix G and the attached code. The expert system used for initial variable selection in the real-world deployment is a commercially available LLM-based agent developed by our industry partner (the Food Scientist Agent, FSA; Section 5). The FSA's output (the selected ingredients) and every recipe evaluated during BO refinement, with its panel scores, are released, so the BO refinement stage can be audited and its analyses reproduced without access to the FSA.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes. The code, benchmark data, and frozen experiment records are publicly released. The repository includes entry-point scripts for all three experimental settings (Hartmann6, FORMULATEBENCH, real-world deployment), `requirements.txt` files with pinned library versions, a README with reproduction instructions, and a single script that verifies the paper's headline numbers and regenerates its figures from the released records. FORMULATEBENCH results are reproducible without API access via the cached LLM trajectories. The real-world deployment data (per-recipe ingredient levels, aggregated panel scores, and utilities) are included; the industry partner's proprietary ingredient catalog is not.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental parameters are stated in Section 4, Section 5 or Appendices F, E, G.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The synthetic (Hartmann6) experiments report 95% confidence intervals (mean ± 1.96 standard errors) over 10 random seeds (20 for the Q-sweep). FORMULATEBENCH aggregate figures report 95% confidence intervals across category-level mean trajectories, and per-category figures report 95% confidence intervals across 10 optimization seeds; each temperature-zero LLM baseline is a single cached trajectory per category. The real-world deployments (yogurt, cottage cheese) are single-run campaigns due to the cost of laboratory iterations (~ 40 hours each), so they carry no error bars; repeated campaigns were not feasible. We report per-iteration and per-recipe scores to allow readers to assess variability within each campaign. The FORMULATEBENCH regime and noise analyses (Appendices F.5 and F.6) report paired Wilcoxon signed-rank tests and Spearman correlations across the 24 categories; the noise sweep uses 10 seeds for EGBO and three noise draws per category for each LLM baseline. The batched experiments, the comparison with other LLMs, and the tool-using-agent test (Appendices F.7–F.9) use 10 seeds per category for EGBO and the conventional baselines and one trajectory per category for each LLM or agent arm, with paired Wilcoxon signed-rank tests across categories.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. Compute resources, hardware, per-run and total wall time, memory, and laboratory time for all experiments are documented in Appendix D.1, including a summary table and per-experiment paragraphs. The full reproduction of every numeric result requires approximately 1.8×10^2 CPU-hours on a single workstation; no GPU is required. Rerunning

the LLM-based experiments additionally requires approximately 19,000 LLM API calls in total (about 5,500 for the observation-noise sweep, 9,000 for the batched, other-LLM, and noise-aware experiments, and 4,800 for the tool-using-agent test); all LLM responses and agent transcripts are released, so the reported analyses run without API access.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes. The sensory evaluation was conducted by trained panelists employed by the industry partner as part of their professional role; panelists were compensated through their regular employment and were informed of relevant allergens prior to tasting. All ingredients used are GRAS-classified and prepared under established food safety protocols. The human food scientist comparison involved a professional who volunteered their expertise as a collaborator; they were not a research subject. No personally identifiable information about panelists or the food scientist is reported. The research aims to accelerate sustainable protein development, and we discuss both positive societal impacts and potential risks of misuse in Section 6. We release the code, benchmark data, and deployment recipes to support reproducibility.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: These are discussed in Section 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not believe our algorithms pose high risk of misuse. The released assets are optimization code and a nutritional benchmark; they do not include trained models or sensitive data. Potential dual-use risks are discussed in Section 6.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. USDA FoodData Central is a U.S. government work in the public domain (CC0 1.0) and is cited with its access URL. The NECTAR plant-based ingredient dataset is cited and used in accordance with its published terms. BoTorch is an open-source library released under the MIT License. Claude Opus 4.7, used for LLM-guided variable selection in FORMULATEBENCH, was accessed via the OpenRouter API in accordance with Anthropic's and OpenRouter's terms of use. The proprietary Food Scientist Agent used in the real-world deployment is a commercially available system used with permission from the developing company, our industry partner. Our code is released under the MIT License; the data it uses keep their original terms, which are listed in the repository's `LICENSE` and `DATA.md` files.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes. We introduce two new assets. (1) FORMULATEBENCH, a benchmark suite of 24 plant-based formulation tasks. The benchmark construction is documented in Section 4.2 and Appendix F.1, including the data sources (USDA FoodData Central, NECTAR), the objective function definition, the ingredient ground sets, nutritional targets per category, and the LLM prompt used for variable selection. The cached LLM selections are included in the released code for reproducible results without API access. (2) The EGBO codebase, including all optimization code, baseline implementations, and the Streamlit application used in the real-world deployment. Entry points, hyperparameters, and library version pins are documented in Appendix D. The code is released under the MIT License, and data terms are documented in the repository, with a README covering installation, reproduction of all experiments, and intended use. Limitations of FORMULATEBENCH - in particular its use of nutritional distance rather than sensory similarity as the primary objective - are discussed in Section 6.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Yes. The full sensory evaluation protocol, including panel composition, calibration procedures, evaluation stages, between-sample palate cleansing, and the attribute scoring scale, is documented in Appendix G.3. The full instructions given to the professional human food scientist, including the objectives, constraints, and procedures for both the from-scratch exploration and the same-start refinement phases, are documented in Appendix G.7.1. The sensory panelists were compensated through their regular salaries as employees of the food company, as is standard; the sensory evaluation was part of their regular professional duties. The human food scientist participated voluntarily as a professional collaborator and was not separately compensated for this study.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The sensory evaluation was conducted by trained panelists at the industry partner as part of their standard product development workflow. All ingredients used are classified as Generally Recognized as Safe (GRAS) and were prepared under the partner's established food safety protocols. Consistent with safety standards, participants were notified of relevant allergens before sample consumption, though full ingredient transparency was withheld to minimize sensory bias. No personally identifiable information about panelists was collected; individual panelist scores were used only to compute panel medians, and only aggregated scores are reported and released. As this activity constituted routine product evaluation rather than human subjects research, formal IRB review was not required under the applicable institutional guidelines. The human expert comparison involved an independent professional food scientist engaged by the industry partner to develop a plant-based yogurt formulation. The food scientist's participation was conducted under a voluntary collaboration arrangement with the industry partner. The food scientist is not identified by name in this paper. As the food scientist's role was that of a domain expert providing a professional work product rather than a research subject, this component does not constitute human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The expert in EGBO can be an LLM, and we use Claude Opus 4.7 as the expert in our main FORMULATEBENCH experiments, with GPT-6.1 Sol and Gemini 4 Argon as the expert and iterative LLM in additional comparisons, and all three as tool-using agents (Appendix F.9). We also use an LLM-based agent (FSA) in the real-world deployment. Claude Code and OpenAI Codex were used for coding support, code auditing, and editing; all generated code and text were reviewed and validated by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.