Agentic Numeric Optimization for Nutrition Planning

Kritik Soman, Member IEEE [1]

Abstract

We introduce *Agentic Numeric Optimization* (ANO), a framework for nutrition planning that combines a Large Language Model (LLM) orchestrator with a numeric optimizer for meal selection and quantity assignment. Given a free-text user profile query, ANO computes dietary targets, retrieves matching ingredients from the OpenNutrition dataset, builds a structured diet specification, and calls the optimizer to assign ingredient quantities. To measure the benefit of solver-backed planning, we compare ANO against both a Baseline Agent (BA), which shares the same target-computation and retrieval tools but relies on iterative reasoning guided by a nutrient calculator, and a deterministic No-LLM baseline that uses fixed templates and heuristics with the same optimizer. We evaluate all three planners on synthetically generated benchmarks based on the Canada Health Dietary Reference Intakes (DRI) and MyFitnessPal (MFP) style nutrition targets. Relative to BA, ANO substantially improves calorie adherence: on DRI, the share of profiles within 10 kcal improves from 9.0% to 79.5%, while on MFP it improves from 3.1% to 78.0%; the share above 100 kcal falls from 74.8% to 6.8% on DRI and from 79.7% to 6.8% on MFP. While the No-LLM baseline is numerically stronger on several bins, and BA is stronger on palatability, ANO is competitive on cuisine alignment.

1. Introduction

LLM agents are increasingly being used for scheduling, travel, and diet planning because they can interpret and invoke multiple tools in sequence. Such tasks also require numeric optimization, LLMs fall short because their outputs are not constrained to satisfy strict numeric bounds.

[1]Independent Researcher. Correspondence to: Kritik Soman <kritik.soman@ieee.org>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

Planning a nutritionally valid diet requires meeting calorie, macronutrient, and fibre targets. Nutrition guidelines are defined by organizations such as the Canada Health Dietary Reference Intakes (DRI) (Health Canada, 2023) and fitness apps like MyFitnessPal (MFP) (MyFitnessPal, 2026), which use calorie targets derived from the Mifflin–St Jeor equation (Mifflin et al., 1990). DRI targets include calories, macronutrient ranges, and fibre, and they are computed from demographic inputs (such as age and sex) together with physiological inputs (such as activity levels, pregnancy, or lactation status). These prescribe nutrition for maintaining body weight. In contrast, MFP lacks fibre and is based on the user's fitness goal, such as weekly weight-change rate and carbohydrate-fat preference. The importance of tracking and maintaining dietary intake is well established. Small sustained calorie mismatches can affect long-run body weight. For example, it is estimated that each persistent 100 kJ/day energy imbalance eventually changes average body weight by about 1 kg, so a sustained 100 kcal/day deficit would correspond to an eventual steady state reduction of roughly 4.2 kg (Hall et al., 2011).

Tracking dietary intake with apps such as MyFitnessPal or Cal AI (Cal AI, 2026) requires the user to manually search for a food item or scan it using the mobile camera. The meal plans in such apps are generally predefined by nutritionists or based on popular diets that may not be personalized to the user's preference. They also do not take into account availability of ingredients to the user. The nutrient calculation is done based on the specific ingredient and quantity logged by the user.

Planning dietary intake solely with LLMs helps users create plans depending on their preferences but the nutrient calculation done by LLMs is not based on specific ingredients used in the meal. This could be addressed by grounding it with a tool for adding the exact macronutrient information into the context and a separate tool to calculate the total. We refer to this approach as Base Agent (BA), where the LLM is responsible for reasoning, identifying the ingredients and quantities needed. Since LLMs struggle to reason and enforce strict numeric constraints, we propose to incorporate an optimizer tool that helps address this issue for creating personalized multi-day meal plans, which leads to nutrient deficit, surplus, or balance based on the user's preference. This would be more reliable than manual logging, as the

nutrient calculation in the meal plan is based on specific ingredients retrieved from a database and quantities computed by an optimizer. We define this approach as Agentic Numeric Optimization (ANO), where the LLM generates the search space for the optimizer. To isolate the benefit of the LLM-generated search space, we also include a No-LLM planner that removes the LLM from the planning loop and instead uses cuisine-aware fixed templates and deterministic heuristics to construct the optimizer search space. The objective function is predefined in the optimizer tool but the target and constraints are retrieved through the same tools. To the best of our knowledge, this is the first work to use an LLM to construct the optimization search space itself, and to evaluate it on nutrition planning tasks with explicit numeric targets and constraints. Example conversations comparing BA, ANO, and No-LLM heuristics on a representative user query are shown in Figures 2a, 2b, and 2c. The BA agent fails to satisfy all the dietary targets (highlighted in red), the ANO agent satisfies the targets with the optimizer tool, and the No-LLM heuristics also satisfy the targets but with weaker cuisine alignment and palatability (highlighted in red).

In ANO, an agentic orchestrator reads a user query in natural language, computes query-specific targets, and retrieves candidate ingredients to build a structured diet specification. It then calls a Mixed-Integer Linear Program (MILP) solver to assign ingredient quantities within specified bounds; if no valid plan is found, the orchestrator adjusts the ingredient choices and retries. To define the ingredient search space, we use the OpenNutrition 2025 dataset (OpenNutrition, 2025). It contains attributes such as ingredient name, alternate names, description, and nutrient values per 100 g. We use these fields to create a search space and then optimize calories, protein, carbohydrate, and fat for both DRI and MFP, with fibre added for DRI.

2. Related Work

Planning is a central capability of agentic systems. It may be achieved through Chain-of-Thought (CoT) prompting, where the LLM performs internal step-by-step reasoning before answering (Wei et al., 2022), ReAct (Yao et al., 2023), which interleaves reasoning with external actions so the plan can adapt after each observation, and ReWOO (Xu et al., 2023), which first writes an explicit multi-step workflow with tool calls and then executes it with less repetitive replanning.

LLMs have been used to convert natural-language problem descriptions into optimization formulations in work on auto-formulation and in the NL4Opt competition (Ramamonjison et al., 2022; 2023). Such agents have also been developed for domains such as travel planning (Zhai & Awasthi, 2025), where an LLM first generates an initial itinerary, after which the system grounds it with real-world data, augments it with retrieved substitute activities, and refines it with a two-stage optimizer. In contrast, ANO uses the LLM to construct the optimization search space itself and solves a nutrition-specific MILP over ingredient choices and quantities to satisfy explicit dietary targets.

Diet optimization predates LLMs by decades and is deeply rooted in operations research and recommender-system methods such as in Stigler's cost-of-subsistence formulation (Stigler, 1945). Later work extends linear programming to personalized diet design (Gazan et al., 2018; van Dooren, 2018). Retrieval- and ranking-based food recommendation pipelines have used structured user and food representations together with explicit objective trade-offs (Min et al., 2019; Trattner & Elsweiler, 2017).

Recent studies use LLMs for nutrition support and diet recommendation (Kirk et al., 2023; Papastratis et al., 2024), and hybrid systems combine LLM reasoning with retrieval or external planning modules for diet generation (Damette et al., 2025; Khamesian et al., 2025). Empirical evaluations of LLM-based nutrition recommendation systems report persistent quantitative errors in calorie and macronutrient guidance, including in clinically sensitive renal and dialysis settings (Bergling et al., 2025; Wang et al., 2024). This motivates our focus on explicit numeric optimization for nutrition planning with ANO.

3. Our Approach

Figure 1 summarizes the three approaches namely, (A) No-LLM, (B) ANO, and (C) BA using a sequence diagram. Given a free-text user profile (shown in step 1 of Figure 1), the orchestrators execute the tool calls sequentially: compute targets (step 1.1.2 and 1.2.2), retrieve candidate ingredients (step 1.1.4 and 1.2.4), build a structured diet specification (step 1.1.5 and 1.2.5), invoke the quantity optimizer, and, if needed, revise (step 1.1.7 and 1.2.7) the specification before presenting a plan (step 1.1.9 and 1.2.9). In the following subsections, we describe each tool in turn, highlighting the shared components and the key difference in quantity optimization.

3.1. Orchestrator

Both ANO and BA use the same base LLM Orchestrator and the same system prompt (Appendix 9.4). The prompt also defines the tool APIs and provides examples of how to use them. The only difference is that the ANO prompt includes additional instructions and examples for how to build a structured diet specification and call the optimizer, whereas the BA prompt includes instructions and examples for how to call the nutrient calculator iteratively to adjust ingredient choices and quantities.

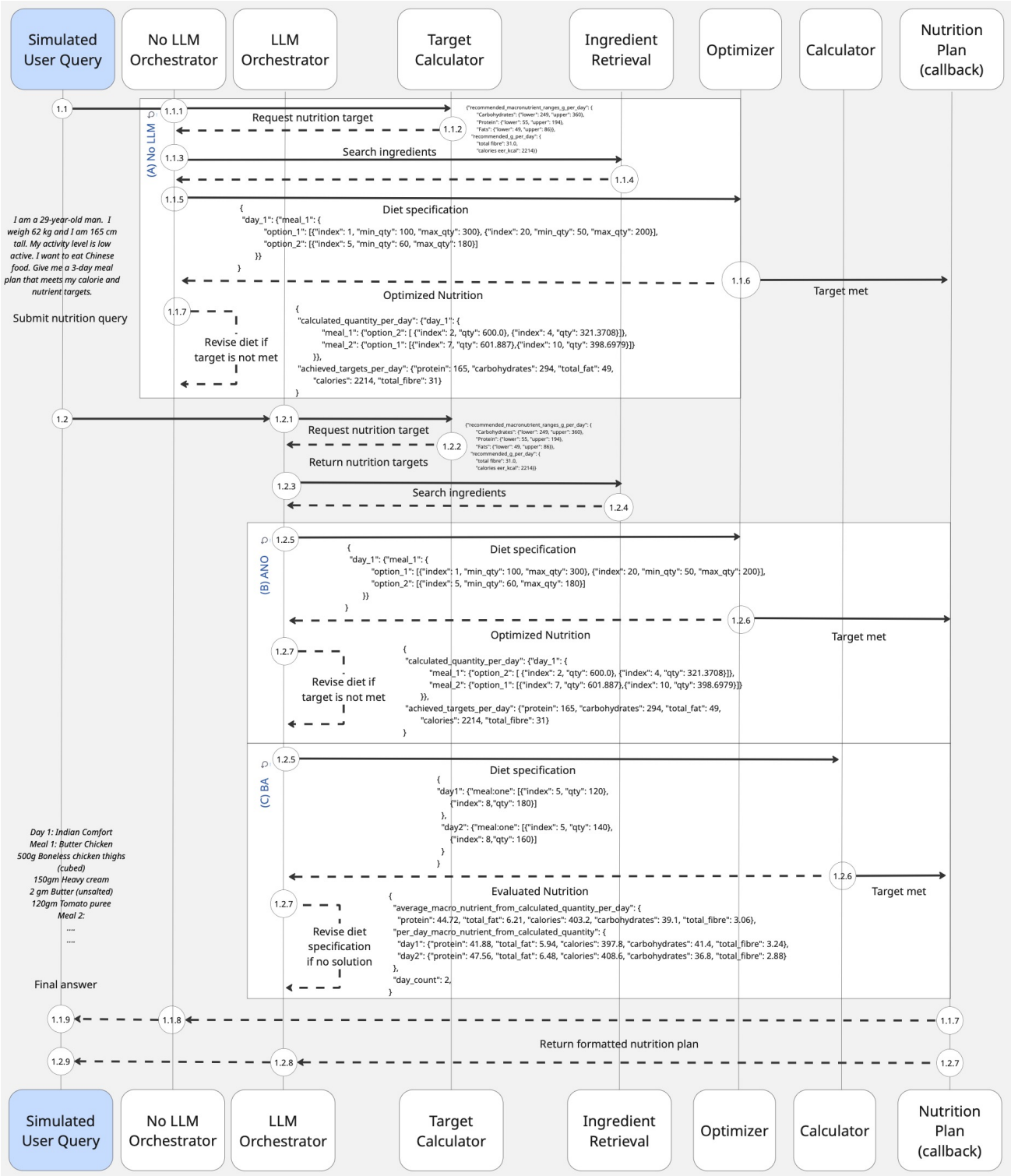


Figure 1. Sequence diagram comparing all the approaches.

The No-LLM orchestrator removes the language model from the planning loop. It first parses the user profile, cuisine, and requested number of days, then computes the nutrition targets and retrieves candidate ingredients using the same target calculators and ingredient retrieval tool as the other planners. Instead of asking an LLM to build a structured diet

specification, it uses fixed meal templates and hand-written heuristics to construct the search space for the optimizer. The heuristic rules are described in Appendix 9.3.

For DRI, the heuristic planner selects breakfast, lunch, and dinner templates and assigns quantity ranges based on ingredient type and age. For MFP, it uses the same template structure but additionally adapts quantity ranges according to the parsed goal and carbohydrate/fat preference. In both cases, the resulting heuristic search space is passed to the same optimizer, with up to three retries if a feasible plan is not found.

3.2. Target Calculator Tool

The DRI target tool takes age, gender, body size, activity level, and, when applicable, pregnancy or lactation status, and returns point targets for average daily calories (EER) and fibre, along with g/day ranges for protein, carbohydrate, and fat. An example output from the tool is shown in Step 1.1.2 and 1.2.2 of Figure 1.

Likewise, the MFP macro tool takes age, gender, body size, activity level, fitness goal, weekly weight-change rate, and carbohydrate/fat preference, and then returns point-valued targets for calories, protein, carbohydrate, and fat.

We skip other nutrient targets such as vitamins and minerals, which are not consistently available across all ingredients in the dataset and would require more complex handling of missing values. We also do not model user preferences such as cuisine or ingredient dislikes as explicit constraints in the optimization, but we do measure cuisine alignment and palatability as post-hoc metrics.

3.3. Ingredient Retrieval Tool

Let a_i^k be the per-100g nutrient k for ingredient i, where k belongs to the set of nutrients {calories, protein, carbohydrate, fat, fibre}. We preprocess OpenNutrition by removing nutritionally invalid entries such as those with missing required energy/fat/carbohydrate/protein fields, values outside per 100g bounds, macro sums above 100g, or calorie entries that fail the following consistency check (all per 100g):

$$0.6\,(9a_i^{\text{fat}} + 4a_i^{\text{carb}} + 4a_i^{\text{prot}}) \le a_i^{\text{cal}} \le 1.4\,(9a_i^{\text{fat}} + 4a_i^{\text{carb}} + 4a_i^{\text{prot}}).$$

Then, an ingredient retrieval tool is created that performs generative search on this dataset (OpenNutrition, 2025). It receives regex patterns from the LLM for ingredient name, alternate names, nutrition per 100 g, and description fields. For textual fields (such as name, alternate names, and description), we use a system prompt that asks the LLM to generate regex patterns in decreasing order of specificity, so that exact or near-exact matches are preferred. For example, the LLM might generate a pattern like *chicken.*breast|chicken|poultry* to capture the user's intent for "chicken breast" while still allowing for more general matches. For fields with numeric values (such as nutrition per 100 g), we prompt it to generate lower and upper bounds for relevant nutrients. The retrieval tool then executes a SQL query that applies these regex patterns and numeric bounds to the relevant fields in the database. The returned records include the ingredient name, nutrient values per 100 g, and the conversation-based ingredient index (as shown in Step 5 of Figure 1).

For example, if the user asks for a chicken-breast ingredient with at least 20 g of protein and at most 10 g of fat per 100 g, the LLM would issue a retrieval-tool call of the form:

```
search_ingredient(
  name_regex="chicken.*breast|chicken|poul
  ↪  try",
  nutrient_bounds={
    "protein": {"min": 20},
    "fat": {"max": 10}
  },
  limit=5
)
```

The retrieval tool then compiles the following SQL query for this request.

```
SELECT name, alternate_names,
↪  nutrition_100g, description
FROM OpenNutrition
WHERE (name ~*
↪  'chicken.*breast|chicken|poultry'
       OR alternate_names ~* 'chicken.*bre
       ↪  ast|chicken|poultry')
  AND substring(nutrition_100g::text FROM
  ↪  '"protein"\s*:\s*([0-9.]+)')::double
  ↪  precision >= 20
  AND substring(nutrition_100g::text FROM
  ↪  '"fat"\s*:\s*([0-9.]+)')::double
  ↪  precision <= 10
LIMIT 5;
```

Here the regex over `name` and `alternate_names` handles lexical matching, while the `substring(... FROM ... )::double precision` expressions extract nutrient values from the stored `nutrition_100g` field so they can be compared numerically. In effect, generative ingredient search combines regex-based lexical retrieval over product metadata with SQL predicates over extracted nutrient values, and then returns the top K matches for downstream planning.

3.4. Plan Assessment Tool

The three planners differ in how they construct the meal search space and assign quantities. ANO uses the LLM to

author the search space, BA uses the LLM together with the nutrient calculator to revise the search space iteratively, and the No-LLM baseline uses fixed meal templates and deterministic heuristics with the same optimizer. The final solution is the best of three tool retries enforced deterministically on infeasible solutions for each planner.

3.4.1. BA

The plan-assessment tool is a nutrient calculator that takes as input a proposed set of ingredient indices together with their assigned quantities. It returns the achieved nutritional totals for the proposed plan, including calories, protein, carbohydrate, fat, and fibre for the relevant nutrition target. The calculation is deterministic: for each selected ingredient, this tool reads nutrient values per 100 g from the tool-call history using the ingredient index, rescales them by the proposed quantity, computes the daily total for each day, and then averages across all days to obtain plan-level totals.

3.4.2. ANO

The LLM builds a structured diet specification that defines the discrete diet options z_{dmo} and continuous quantity intervals $q_{d,m,o,i}$ for the optimizer (shown in Step 6.1 of Figure 1(A)). Let x denote the conversation and let $\tau(x)$ denote the nutrition target. The LLM builds a solver specification

$$\begin{aligned} s(x) = \big(&\mathcal{D},\, \mathcal{M},\, \{\mathcal{O}_{d,m}\},\, \{\mathcal{I}_{d,m,o}\}, \\ &\{L_{d,m,o,i}\},\, \{U_{d,m,o,i}\},\, \tau(x)\big), \end{aligned} \tag{1}$$

where $\mathcal{D}$ is the set of days, $\mathcal{M}$ is the set of diet slots for each day, $\mathcal{O}_{d,m}$ is the set of diet options for slot (d, m), $\mathcal{I}_{d,m,o}$ is the set of retrieved ingredient indices for option o, and $L_{d,m,o,i}, U_{d,m,o,i}$ are the lower and upper quantity bounds per ingredient. The important point is that the LLM chooses which discrete candidates and continuous intervals the optimizer is allowed to explore.

Next, we formalize the ingredient search space generated by the LLM. For day d and diet slot m with options $\mathcal{O}_{d,m}$, each option o containing items $\mathcal{I}_{d,m,o}$ and bounds $(L_{d,m,o,i}, U_{d,m,o,i})$, the ingredient search space authored by the LLM is defined by the option-selection and quantity constraints:

$$\begin{aligned} &\sum_{o \in \mathcal{O}_{d,m}} z_{dmo} = 1, \quad \forall\, d \in \mathcal{D},\, m \in \mathcal{M}, \\ &L_{d,m,o,i} z_{dmo} \le q_{d,m,o,i} \le U_{d,m,o,i} z_{dmo}, \\ &\qquad\qquad \forall\, d, m,\ o \in \mathcal{O}_{d,m},\ i \in \mathcal{I}_{d,m,o}. \end{aligned} \tag{2}$$

The first equation in (2) ensures that exactly one option is selected for each diet slot, while the second links the quantity variables to the selected options, enforcing that quantities are zero when an option is not selected and within specified bounds when it is.

Each day is optimized independently using the same targets, and per-day optimization jobs are executed in parallel to reduce end-to-end latency. For each day d, we compute the achieved values,

$$\hat{T}_{k,d} = \sum_{m,o,i} a_i^k q_{d,m,o,i}, \quad \forall d \in \mathcal{D}, \tag{3}$$

where $q_{d,m,o,i}$ are the quantities optimized for day d. After optimizing all days, we compute average achieved values across days,

$$\hat{T}_k = \frac{1}{D} \sum_d \hat{T}_{k,d}. \tag{4}$$

For DRI, let $\mathcal{K}_r \subseteq \{\text{protein, carbs, fat}\}$ denote the range-constrained nutrients with lower and upper bounds R_k^{L} and R_k^{U}, and let $\mathcal{K}_p = \{\text{calories, fibre}\}$ denote the point-target nutrients with targets T_k.

For MFP, let $\mathcal{K}_{\mathrm{mfp}} = \{\text{calories, protein, carbs, fat}\}$ denote the point-target nutrient set with targets T_k. For each point-target nutrient and each day, we introduce nonnegative positive and negative deviation variables $d_{k,d}^+, d_{k,d}^-$ and optimize their weighted sum using weights $w_k = \frac{\alpha_k}{\max(T_k, 1)}$. When referring to individual nutrients, we write $T_{\text{cal}}, T_{\text{fib}}, T_{\text{prot}}, T_{\text{carb}}, T_{\text{fat}}$ and the corresponding achieved values $\hat{T}_{\text{cal}}, \hat{T}_{\text{fib}}, \hat{T}_{\text{prot}}, \hat{T}_{\text{carb}}, \hat{T}_{\text{fat}}$.

We provide the entire conversation history as input to the plan assessment tool so that it can use the target and ingredient information without any hallucination. The only inputs for this tool that can hallucinate are the ingredient indices and quantity ranges. For assessing whether a diet plan is acceptable, we define explicit feasibility thresholds. For each calorie or nutrient threshold, we approximate the smallest practically meaningful quantity using the most calorie- or nutrient-dense ingredient in the dataset. As an example, half a teaspoon of the most carbohydrate-dense ingredient (sugar) contains 2.1 g of carbohydrate, so we set the carbohydrate deviation threshold to 2 g.

3.5. Nutrition Plan Callback

After optimization, the planner returns a structured callback that includes the selected ingredients, their optimized gram quantities, aggregate nutrient totals, and a feasibility flag. If a feasible meal plan is found, the callback is used to directly construct the user message. On the other hand, if a feasible meal plan is not found, the tool output is passed to the LLM so that it can retry with a different diet specification or conclude the conversation by expressing the infeasibility.

4. Experiments

We implement all agent variants and the evaluation tests in the Agent Development Kit (ADK) (Google, 2026). All ex-

Component	DRI optimization	MFP optimization
Target source	Targets are taken from the DRI tool output. Returned fields include daily calorie and fibre point targets, and protein, carbohydrate, and fat constraints that may be provided as hard Acceptable Macronutrient Distribution Range (AMDR) constraints.	Targets are taken from the MFP macro tool output. Returned fields include daily point targets for calories, protein, carbohydrate, and fat, with no AMDR-style hard ranges.
Range-constrained nutrients	$\mathcal{K}_r \subseteq \{\text{protein, carbs, fat}\},$ $R_k^{\mathrm{L}} \le \hat{T}_{k,d} \le R_k^{\mathrm{U}}, \quad \forall k \in \mathcal{K}_r,\ d \in \mathcal{D}.$	None
Point-target nutrients	$\mathcal{K}_p = \{\text{calories, fibre}\},$ $\hat{T}_{k,d} - d_{k,d}^{+} + d_{k,d}^{-} = T_k, \quad \forall k \in \mathcal{K}_p,\ d \in \mathcal{D}.$	$\mathcal{K}_{\mathrm{mfp}} = \{\text{calories, protein, carbs, fat}\},$ $\hat{T}_{k,d} - d_{k,d}^{+} + d_{k,d}^{-} = T_k, \quad \forall k \in \mathcal{K}_{\mathrm{mfp}},\ d \in \mathcal{D}.$
Objective	$\min \sum_{k \in \mathcal{K}_p} w_k (d_{k,d}^{+} + d_{k,d}^{-}).$ with normalized weights $w_k = \frac{\alpha_k}{\max(T_k, 1)}$, where $(\alpha_{\text{cal}}, \alpha_{\text{prot}}, \alpha_{\text{carb}}, \alpha_{\text{fat}}, \alpha_{\text{fib}}) = (100, 25, 3, 3, 20)$.	$\min \sum_{k \in \mathcal{K}_{\mathrm{mfp}}} w_k (d_{k,d}^{+} + d_{k,d}^{-}).$ with normalized weights $w_k = \frac{\alpha_k}{\max(T_k, 1)}$, where $(\alpha_{\text{cal}}, \alpha_{\text{prot}}, \alpha_{\text{carb}}, \alpha_{\text{fat}}) = (100, 25, 3, 3)$.
Post optimization feasibility check *(applied to averaged results)*	$\lvert\hat{T}_{\text{fib}} - T_{\text{fib}}\rvert \le 5\,\text{g},$ $R_k^{\mathrm{L}} \le \hat{T}_k \le R_k^{\mathrm{U}}, \quad \forall k \in \mathcal{K}_r.$	$\lvert\hat{T}_{\text{cal}} - T_{\text{cal}}\rvert \le 10\,\text{kcal},$ $\lvert\hat{T}_{\text{prot}} - T_{\text{prot}}\rvert \le 5\,\text{g},$ $\lvert\hat{T}_{\text{carb}} - T_{\text{carb}}\rvert \le 2\,\text{g},$ $\lvert\hat{T}_{\text{fat}} - T_{\text{fat}}\rvert \le 5\,\text{g}.$

Table 1. MILP formulation across DRI and MFP nutrition guidelines.

Factor	Category	N
Gender	Male	113
	Female	227
Age group	Young (≤ 50)	284
	Old (> 50)	56
Activity level	Inactive	71
	Low active	112
	Active	82
	Very active	75
Pregnancy status	Not pregnant	284
	Pregnant	56
Gestation stage (pregnant only)	First half (< 20 weeks)	19
	Second half (≥ 20 weeks)	37
Lactation status	None	292
	0–6 months postpartum	25
	7–12 months postpartum	23
Prepregnancy BMI class (pregnant only)	Underweight	16
	Normal weight	19
	Overweight	7
	Obese	14

(a) DRI target-finder category counts ($N = 340$)

Factor	Category	N
Gender	Male	161
	Female	159
Goal	Lose weight	107
	Maintain	107
	Gain weight	106
Activity level	Sedentary	70
	Lightly active	60
	Moderately active	65
	Very active	66
	Extra active	59
Carb/fat preference	Balanced	126
	Lower carb	75
	Higher carb	58
	Higher protein	61
Weekly rate (lb/week)	0.0	107
	0.25	8
	0.5	62
	0.75	15
	1.0	51
	1.5	25
	2.0	52
Age band	< 19	58
	19–29	43
	30–49	122
	≥ 50	97
Gender $\times$ goal	male / lose weight	54
	male / maintain	54
	male / gain weight	53
	female / lose weight	53
	female / maintain	53
	female / gain weight	53

(b) MFP macro category counts ($N = 320$)

Table 2. Category counts for both generated datasets.

periments use `gemini-3-flash-preview` except for the cuisine-alignment and palatability judge, which uses `gemini-2.5-flash`. Both of these models have reasoning on by default. Prompt caching is used for both ANO and BA to reduce cost. For the optimizer, we use (Google, 2025) OR-Tools with the MILP solver. We allow up to three retries for each planner if a feasible solution is not found, and we report the best solution across all retries.

4.1. Benchmarks

We evaluate No-LLM, BA, and ANO on both DRI and MFP targets. Two synthetically generated benchmark datasets (Table 2) are produced deterministically for this evaluation. Across both targets, common inputs are gender, age, weight, height, and activity level; the DRI benchmark additionally includes pregnancy status, gestation weeks, pre-pregnancy weight, and lactation status, whereas the MFP benchmark additionally includes goal, weekly weight-change rate, and carbohydrate-fat preference. To generate diverse cuisine preferences, we define cuisine values as Indian, Chinese, Mexican, Thai, Japanese, Mediterranean, Italian, Korean, American, and Middle Eastern.

The first is a 340-profile DRI-based dataset built as the union of a 140-profile core set and a 200-profile random valid set. The core set consists of manually specified scenarios chosen to cover the main DRI rule cases and important cutoff values across all inputs. The 200-profile random valid set is generated separately by randomly sampling valid profiles drawn across the same attributes. Table 2(a) reports the resulting category distribution.

The second is a 320-profile MFP-based dataset built as the union of a 122-profile core set and a 198-profile random valid set. Its construction also begins with an initial set of manually specified profiles chosen to cover all inputs. The remainder is then filled with randomly sampled valid MFP profiles while keeping counts balanced across gender$\times$goal strata. Table 2(b) reports the final category counts for this MFP dataset.

Each profile is randomly assigned one cuisine and a day target from $\{3, 5, 7\}$. It is then converted into a natural language user query. Example DRI and MFP queries are shown below.

DRI: *"I am a 30-year-old woman. I weigh 66 kg and I am 165 cm tall. My activity level is low active. I am breastfeeding (0–6 months postpartum). I want to eat Mediterranean food. Give me a 5-day diet plan that meets my calorie and nutrient targets."*

MFP: *"I am a 40-year-old man. I weigh 84 kg and I am 177 cm tall. My activity level is moderately active. I want to lose weight at about 1 lb per week. I'd like a higher protein macro split. I want to eat Mexican food. Give me a 3-day diet plan that meets my calorie and macro targets."*

4.2. Evaluation Metrics

We compute the following per-conversation metrics, and then aggregate them.

- **Calorie deviation (kcal)**: $|\hat{T}_{\text{cal}} - T_{\text{cal}}|$.
- **Min daily calorie (kcal)**: the minimum achieved daily calories within a conversation.
- **Max daily calorie (kcal)**: the maximum achieved daily calories within a conversation.
- **Fibre deviation (g)**: $|\hat{T}_{\text{fib}} - T_{\text{fib}}|$ for the DRI fibre point target. Reported as "N/A" for MFP profiles, which have no fibre target in $\mathcal{K}_{\text{mfp}}$.
- **Protein / carbohydrate / fat outside-range deviation (g)**: the absolute deviation of $\hat{T}_k$ below R_k^{L} or above R_k^{U} for $k \in \mathcal{K}_r$, and zero when $R_k^{\text{L}} \leq \hat{T}_k \leq R_k^{\text{U}}$. For MFP, where macronutrients are point-valued targets in $\mathcal{K}_{\text{mfp}}$, this reduces to $|\hat{T}_k - T_k|$.
- **Cuisine alignment score**: an LLM-judge metric that compares the requested cuisine in the user prompt with the returned diet plan on a per-day basis. The judge assigns each detected day a score in $\{1.0, 0.7, 0.4, 0.0\}$ for strongly aligned, mostly aligned, partially aligned, and not aligned, respectively, and the final conversation-level score is the average across the requested number of days. Although this metric is not modeled as part of the optimization, we use it to measure the extent to which the solver's numeric precision comes at the cost of cuisine alignment.
- **Palatability score**: an LLM-judge score of perceived meal quality and coherence of the generated plan, aggregated at conversation level. The judge scores each detected day on a scale of 0 to 1.0, where 1.0 represents highly appealing and coherent meals with good variety and realistic pairings, 0.7 represents generally appealing meals with minor issues, 0.4 represents edible but weak, repetitive, awkward, or poorly paired meals, and 0.0 represents clearly unappealing, incoherent, or unrealistic meals. The conversation-level score is the average across the detected days.
- **Tool retries**: the number of plan-revision attempts before termination (bounded to at most three retries).
- **Latency (seconds)**: end-to-end conversation runtime reported with percentile summaries (p50 and p95).

All the targets above were computed from step 1.1.2, 1.2.2 and step 1.1.6, 1.2.6 of the pipeline in Figure 1.

Canada Health DRI			
Metric	**No-LLM**	**BA**	**ANO**
Calorie deviation $\leq$ 10 kcal	84.7%	9.0%	**79.5%**
Calorie deviation 10–50 kcal	8.2%	7.2%	10.4%
Calorie deviation 50–100 kcal	4.4%	9.0%	**3.3%**
Calorie deviation $>$ 100 kcal	2.6%	74.8%	**6.8%**
Protein deviation $\leq$ 5 g	100.0%	99.3%	**100.0%**
Protein deviation 5–25 g	0.0%	0.3%	**0.0%**
Protein deviation 25–50 g	0.0%	0.0%	0.0%
Protein deviation $>$ 50 g	0.0%	0.3%	**0.0%**
Carbohydrate deviation $\leq$ 2 g	99.7%	42.7%	**97.0%**
Carbohydrate deviation 2–5 g	0.0%	0.7%	1.0%
Carbohydrate deviation 5–50 g	0.3%	28.7%	**1.7%**
Carbohydrate deviation $>$ 50 g	0.0%	28.0%	**0.3%**
Fat deviation $\leq$ 5 g	98.0%	67.6%	**98.3%**
Fat deviation 5–10 g	1.0%	11.3%	**1.0%**
Fat deviation 10–25 g	0.7%	14.3%	**0.7%**
Fat deviation $>$ 25 g	0.3%	6.8%	**0.0%**
Dietary fiber deviation $\leq$ 5 g	90.6%	29.6%	**84.1%**
Dietary fiber deviation 5–10 g	7.7%	22.0%	**10.8%**
Dietary fiber deviation 10–15 g	1.0%	19.9%	**1.7%**
Dietary fiber deviation $>$ 15 g	0.7%	28.5%	**3.4%**
Min daily calorie $<$ 1000 kcal	13.6%	14.8%	**10.8%**
Min daily calorie 1000–1400 kcal	4.2%	6.2%	**4.4%**
Min daily calorie $>$ 1400 kcal	82.2%	79.1%	**84.8%**
Max daily calorie $>$ 3800 kcal	0.9%	2.5%	**0.6%**
Max daily calorie 3400–3800 kcal	2.4%	8.6%	**5.4%**
Max daily calorie $<$ 3400 kcal	96.7%	88.9%	**94.0%**
Cuisine alignment score $<$ 0.25	72.6%	6.5%	12.9%
Cuisine alignment score 0.25–0.50	22.9%	2.4%	11.8%
Cuisine alignment score 0.50–0.75	4.4%	21.9%	28.5%
Cuisine alignment score $\geq$ 0.75	0.0%	69.2%	46.8%
Palatability score $<$ 0.25	75.0%	3.0%	11.8%
Palatability score 0.25–0.50	23.5%	2.4%	29.7%
Palatability score 0.50–0.75	1.5%	34.6%	39.1%
Palatability score $>$ 0.75	0.0%	60.1%	19.4%
Tool retries $=$ 0	34.4%	83.4%	68.8%
Tool retries $=$ 1	16.2%	8.9%	23.5%
Tool retries $=$ 2	49.4%	7.4%	**6.2%**
Latency p50 (s)	8.9	24.3	39.8
Latency p95 (s)	104.3	60.8	102.4

Table 3. Evaluation metrics for No-LLM, BA, and ANO on Canada Health DRI, including palatability, retries, and latency metrics.

MyFitnessPal			
Metric	**No-LLM**	**BA**	**ANO**
Calorie deviation $\leq$ 10 kcal	100.0%	3.1%	**78.0%**
Calorie deviation 10–50 kcal	0.0%	5.3%	8.1%
Calorie deviation 50–100 kcal	0.0%	11.9%	**7.2%**
Calorie deviation $>$ 100 kcal	0.0%	79.7%	**6.8%**
Protein deviation $\leq$ 5 g	49.7%	20.3%	**62.3%**
Protein deviation 5–25 g	19.4%	50.0%	**32.6%**
Protein deviation 25–50 g	13.8%	20.3%	**3.4%**
Protein deviation $>$ 50 g	17.2%	9.4%	**1.7%**
Carbohydrate deviation $\leq$ 2 g	19.7%	3.4%	**12.7%**
Carbohydrate deviation 2–5 g	11.9%	5.3%	11.0%
Carbohydrate deviation 5–50 g	50.6%	42.2%	60.2%
Carbohydrate deviation $>$ 50 g	17.8%	49.1%	**16.1%**
Fat deviation $\leq$ 5 g	41.9%	20.9%	**41.5%**
Fat deviation 5–10 g	17.2%	14.7%	21.6%
Fat deviation 10–25 g	27.2%	36.9%	**24.6%**
Fat deviation $>$ 25 g	13.8%	27.5%	**12.3%**
Min daily calorie $<$ 1000 kcal	0.0%	5.6%	**1.7%**
Min daily calorie 1000–1400 kcal	8.4%	15.9%	**10.6%**
Min daily calorie $>$ 1400 kcal	91.6%	78.4%	**87.7%**
Max daily calorie $>$ 3800 kcal	3.1%	7.8%	**3.4%**
Max daily calorie 3400–3800 kcal	6.9%	5.9%	6.4%
Max daily calorie $<$ 3400 kcal	90.0%	86.2%	**90.3%**
Cuisine alignment score $<$ 0.25	72.8%	6.2%	8.4%
Cuisine alignment score 0.25–0.50	19.4%	6.2%	6.3%
Cuisine alignment score 0.50–0.75	7.8%	21.9%	16.0%
Cuisine alignment score $\geq$ 0.75	0.0%	65.6%	**69.2%**
Palatability score $<$ 0.25	80.0%	6.2%	11.0%
Palatability score 0.25–0.50	19.7%	8.1%	11.8%
Palatability score 0.50–0.75	0.3%	44.7%	45.1%
Palatability score $>$ 0.75	0.0%	40.9%	32.1%
Tool retries $=$ 0	2.5%	65.6%	**77.2%**
Tool retries $=$ 1	5.0%	24.1%	**19.4%**
Tool retries $=$ 2	92.5%	9.4%	**2.1%**
Latency p50 (s)	10.7	28.2	30.5
Latency p95 (s)	116.5	75.8	105.2

Table 4. Evaluation metrics for No-LLM, BA, and ANO on MyFitnessPal, including palatability, retries, and latency metrics.

5. Results and Analysis

In Figure 3, we visualize the PCA projections of nutritional information for meals generated by ANO across the DRI and MFP benchmarks using TensorBoard (Wongsuphasawat et al., 2018). Each point in the plot represents a meal. The proximity of points indicates similarity in nutrient profiles.

Tables 3 and 4 show two consistent patterns. First, ANO substantially improves numeric adherence relative to BA on both benchmarks. On DRI, ANO raises the share of profiles within 10 kcal from 9.0% to 79.5% and reduces the share above 100 kcal from 74.8% to 6.8%. It also improves carbohydrate deviation within 2 g from 42.7% to 97.0%, fat deviation within 5 g from 67.6% to 98.3%, and fibre deviation within 5 g from 29.6% to 84.1%. On MFP, the same trend holds: ANO increases the share within 10 kcal from 3.1% to 78.0%, protein deviation within 5 g from 20.3% to 62.3%, and reduces the share above 100 kcal from 79.7% to 6.8%. It also shrinks large macronutrient error bins relative to BA, for example reducing carbohydrate deviation above 50 g from 49.1% to 16.1% and fat deviation above 25 g from 27.5% to 12.3%.

Second, the results show that the No-LLM baseline is still strongest on several bins. On DRI, No-LLM achieves the highest share within 10 kcal (84.7%), within 2 g carbohydrate deviation (99.7%), and within 5 g fibre deviation (90.6%), while ANO is close on fat. On MFP, No-LLM remains strongest on calorie adherence, with 100.0% of profiles within 10 kcal and 0.0% above 100 kcal, although ANO outperforms No-LLM on protein and on the largest carbohydrate and fat error bins. This indicates that deterministic template-and-heuristic search-space construction can define a highly feasible optimization region, even though it is less flexible than LLM-authored search spaces.

Preference quality remains a trade-off. BA remains strongest on palatability on both datasets: in the top bin ($>$ 0.75), BA reaches 60.1% on DRI and 40.9% on MFP, compared with 19.4% and 32.1% for ANO, and 0.0% for No-LLM. For cuisine alignment, BA is strongest on DRI (top-bin 69.2% versus 46.8% for ANO), while ANO is strongest on MFP (69.2% versus 65.6% for BA); No-LLM remains lowest in both cases.

The retry and latency statistics are mixed. ANO reduces the share of runs requiring two retries relative to BA on both DRI (6.2% versus 7.4%) and MFP (2.1% versus 9.4%). However, BA has the best p95 latency on both datasets

(60.8 s on DRI and 75.8 s on MFP), while No-LLM has the best p50 latency (8.9 s on DRI and 10.7 s on MFP). Overall, the three-way comparison indicates no single planner dominates all axes: ANO improves numeric adherence over BA, No-LLM remains highly robust on several numeric bins, and BA remains strongest on palatability.

6. Conclusion

We presented ANO, an agentic nutrition-planning framework that combines an LLM orchestrator with a numeric optimizer for quantity assignment. Across both evaluated nutrition targets, ANO improves numeric adherence relative to BA by large margins, while the No-LLM baseline helps isolate the contribution of LLM-authored search-space design under the same optimizer backend.

These results show that making the feasible region explicit and delegating quantity selection to an MILP solver can materially improve adherence over an iterative LLM-only revision loop. At the same time, the full comparison shows persistent trade-offs: No-LLM remains strongest on several numeric bins, and BA remains strongest on palatability. Future work involves improving retrieval and ingredient-set construction so that cultural and culinary intent can be preserved while maintaining strong numeric feasibility.

7. Ethics and Compliance

This paper studies only experiments on synthetic user profiles generated by manually specifying different scenarios and randomly sampling valid input combinations. It does not involve human subjects research and is not deployed on real users. The paper does not include any personally identifiable information or sensitive data. The nutrition targets and ingredient data used in the experiments are based on publicly available sources. The paper does not make any medical claims or provide personalized medical advice, and it is intended for research purposes only. The authors do not have any conflicts of interest to disclose.

8. Code and Data Availability

The code and data used in this work are available at *this* URL.

Impact Statement

This paper presents work on nutrition-planning agents that combine LLM orchestration with explicit numeric optimization. The main anticipated benefit is improved adherence to stated calorie and macronutrient targets relative to direct generation alone. At the same time, such systems should not be treated as medical devices or as substitutes for clinical judgment, particularly for populations with disease-specific dietary requirements. Our experiments are limited to synthetic profiles and publicly available ingredient data, and the paper is intended as a study of planning methodology rather than a source of personalized medical advice.

Although we implemented and evaluated this framework in nutrition planning, its core design is general and can be applied to other rigid, numerically constrained planning domains such as logistics and scheduling. We view this work as a general method for combining language-guided search-space construction with explicit optimization.

Acknowledgment

We thank the reviewers for pointing out several important issues in an earlier version of this work, including the temporal modeling mismatch in multi-day averaging, the absence of a traditional Operations Research baseline to better isolate the value of the LLM generated search space, and the need to report latency and tool retry rates. These concerns have been addressed in the revised manuscript.

References

Bergling, K., Wang, L. C., Shivakumar, O., et al. From bytes to bites: Application of large language models to enhance nutritional recommendations. *Clinical Kidney Journal*, 18(4):sfaf082, 2025. doi: 10.1093/ckj/sfaf082.

Cal AI. Cal ai, 2026. URL https://www.calai.app/. Official website.

Damette, N., Tchappi, I., Mualla, Y., Markovich, R., Najjar, A., and Abbas-Turki, A. Enhancing personalized nutrition: Towards a hybrid intelligence approach with llm-powered meal planning. In *Advances in Explainability, Agents, and Large Language Models*, pp. 1–17. Springer, Cham, Switzerland, 2025. doi: 10.1007/978-3-031-89103-8_1.

Gazan, R., Brouzes, C. M. C., Vieux, F., Maillot, M., Lluch, A., and Darmon, N. Mathematical optimization to explore tomorrow's sustainable diets: A narrative review. *Advances in Nutrition*, 9(5):602–616, 2018. doi: 10.1093/advances/nmy049.

Google. OR-Tools - Google Optimization Tools, 2025. URL https://developers.google.com/optimization. Official documentation and software repository.

Google. Agent development kit, 2026. URL https://adk.dev/. Official documentation.

Hall, K. D., Sacks, G., Chandramohan, D., Chow, C. C., Wang, Y. C., Gortmaker, S. L., and Swinburn, B. A.

Quantification of the effect of energy imbalance on bodyweight. *The Lancet*, 378(9793):826–837, 2011. doi: 10.1016/S0140-6736(11)60812-X.

Health Canada. Dietary reference intakes tables, 2023. URL https://www.canada.ca/en/health-canada/services/food-nutrition/healthy-eating/dietary-reference-intakes/tables.html. Accessed: 2026-03-15.

Khamesian, S., Arefeen, A., Carpenter, S. M., and Ghasemzadeh, H. NutriGen: Personalized meal plan generator leveraging large language models to enhance dietary and nutritional adherence, 2025. URL https://arxiv.org/abs/2502.20601.

Kirk, D., Van Eijnatten, E., and Camps, G. Comparison of answers between ChatGPT and human dieticians to common nutrition questions. *Journal of Nutrition and Metabolism*, 2023:1–9, 2023. doi: 10.1155/2023/5548684.

Mifflin, M. D., St Jeor, S. T., Hill, L. A., Scott, B. J., Daugherty, S. A., and Koh, Y. O. A new predictive equation for resting energy expenditure in healthy individuals. *The American Journal of Clinical Nutrition*, 51(2):241–247, 1990. doi: 10.1093/ajcn/51.2.241.

Min, W., Jiang, S., Liu, L., Rui, Y., and Jain, R. C. A survey on food computing. *ACM Computing Surveys*, 52(5):1–36, 2019.

MyFitnessPal. Myfitnesspal, 2026. URL https://www.myfitnesspal.com/. Official website.

OpenNutrition. Opennutrition dataset, 2025. URL https://www.opennutrition.app. Official dataset and project website.

Papastratis, I., Konstantinidis, D., Daras, P., and Dimitropoulos, K. AI nutrition recommendation using a deep generative model and ChatGPT. *Scientific Reports*, 14(1), 2024. doi: 10.1038/s41598-024-65438-x.

Ramamonjison, R., Li, H., Yu, T., He, S., Rengan, V., Banitalebi-Dehkordi, A., Zhou, Z., and Zhang, Y. Augmenting operations research with auto-formulation of optimization models from problem descriptions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 29–62, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-industry.4. URL https://aclanthology.org/2022.emnlp-industry.4/.

Ramamonjison, R., Yu, T., Li, R., Li, H., Carenini, G., Ghaddar, B., He, S., Mostajabdaveh, M., Banitalebi-Dehkordi, A., Zhou, Z., and Zhang, Y. NL4Opt Competition: Formulating optimization problems based on their natural language descriptions. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pp. 189–203. PMLR, 2023. URL https://proceedings.mlr.press/v220/ramamonjison23a.html.

Stigler, G. J. The cost of subsistence. *Journal of Farm Economics*, 27(2):303–314, 1945.

Trattner, C. and Elsweiler, D. Food recommender systems: Important factors and promising directions. *arXiv preprint arXiv:1711.02760*, 2017.

van Dooren, C. A review of the use of linear programming to optimize diets, nutritiously, economically and environmentally. *Frontiers in Nutrition*, 5:48, 2018. doi: 10.3389/fnut.2018.00048.

Wang, L. C., Zhang, H., Ginsberg, N., Ban, A. N., Kooman, J. P., and Kotanko, P. Application of ChatGPT to support nutritional recommendations for dialysis patients: A qualitative and quantitative evaluation. *Journal of Renal Nutrition*, 34(6):477–481, 2024. doi: 10.1053/j.jrn.2024.09.001.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Mané, D., Fritz, D., Krishnan, D., Viégas, F. B., and Wattenberg, M. Tensorboard: A visualization toolkit for machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):765–774, 2018. doi: 10.1109/TVCG.2017.2744873.

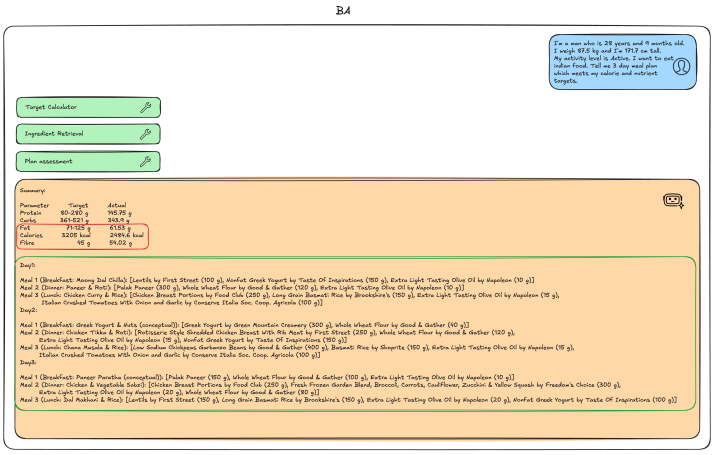
Xu, B., Peng, Z., Lei, B., Mukherjee, S., Liu, Y., and Xu, D. ReWOO: Decoupling reasoning from observations for efficient augmented language models, 2023. Preprint.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

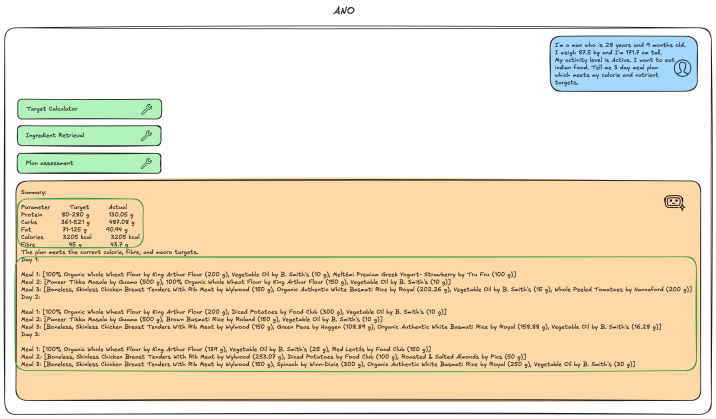
Zhai, A. and Awasthi, P. Optimizing LLM-based trip planning, 2025. URL https://research.google/blog/optimizing-llm-based-trip-planning/. Google Research Blog.

9. Appendix

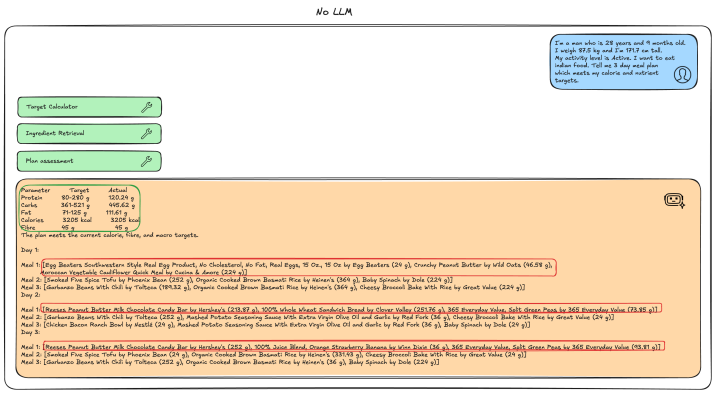
9.1. Additional Figures



(a) BA is unable to satisfy all the dietary targets, but meals have good cuisine alignment and palatability.



(b) ANO is able to satisfy all the dietary targets and maintain good cuisine alignment and palatability.



(c) No-LLM heuristics are able to satisfy all the dietary targets but struggle with cuisine alignment and palatability.

Figure 2. Example conversations comparing BA, ANO, and No-LLM heuristics on the same user query.

9.2. TensorBoard visualization

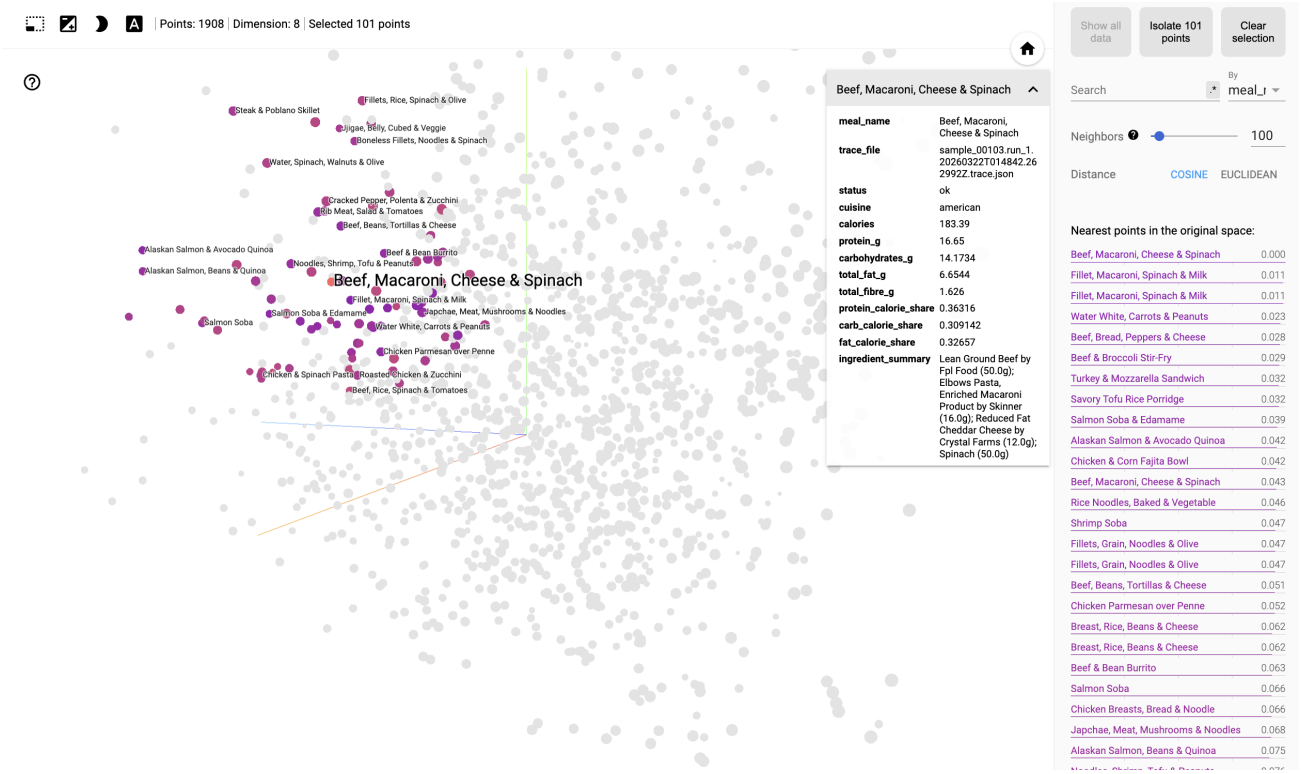


Figure 3. PCA projections of nutrients for meals generated by ANO, visualized using TensorBoard.

9.3. No-LLM Heuristics

The No-LLM baseline uses a fixed meal template sequence of breakfast, lunch, and dinner. Each template defines a pool of candidate ingredient labels; these labels are used to filter retrieved ingredients from the same round of retrieval used by ANO and BA. The ingredient retrieval can be customized by cuisine preference.

CUISINE-BASED INGREDIENT RETRIEVAL

The DRI planner supports seven cuisines: American, Chinese, Indian, Japanese, Thai, Middle Eastern, and Korean. The MFP planner extends this set with Mediterranean, Italian, and Mexican. Each cuisine defines a query library with cuisine-specific ingredient targets. For example:

- **American**: oats, eggs, whole-wheat bread, chicken breast, turkey, brown/white rice, potato, black/kidney beans, broccoli, banana, apple, milk
- **Chinese**: jasmine rice, wheat/rice noodles, firm tofu, chicken, lean pork, bok choy, broccoli, shiitake mushrooms, edamame, egg, napa cabbage, mandarin
- **Indian**: basmati rice, masoor dal, moong dal, chickpeas, paneer, yogurt, chicken, spinach, cauliflower, green peas, potato, oats
- **Japanese**: sushi rice, udon, salmon, tuna, tofu, edamame, egg, seaweed/nori/wakame, shiitake mushrooms, spinach, sweet potato, banana
- **Thai**: jasmine rice, rice noodles, chicken, shrimp, tofu, egg, broccoli, carrot, green beans, mango, pineapple, light coconut milk
- **Middle Eastern**: chickpeas, lentils, bulgur, rice, chicken, lean beef, yogurt, cucumber, tomato, eggplant, spinach, dates

- **Korean**: short-grain rice, rice cakes, sweet potato noodles, chicken, lean beef, tofu, egg, kimchi, spinach, shiitake/oyster mushrooms, napa cabbage, seaweed

When a cuisine is unspecified or invalid, the planner defaults to American. Each cuisine-specific query is then merged with a generic fallback library to ensure coverage of essential food items across all cuisines. The retrieved ingredients are then filtered through the meal template constraints.

TEMPLATE STRUCTURE

For DRI, the fixed templates are:

- **Breakfast**: oats, bread, yogurt, egg, milk, banana, apple
- **Lunch**: rice, noodles, bulgur, potato, chicken, tofu, beans, broccoli, spinach, cabbage, carrot
- **Dinner**: rice, noodles, potato, chicken, tofu, beans, broccoli, spinach, eggplant, mushrooms, tomato, cucumber

For MFP, the same three-slot structure is used, but with expanded pools. Relative to DRI, breakfast adds orange, nuts, peanut butter, avocado, and cheese; lunch adds pasta, tortilla, tomato, avocado, olive oil, and cheese; and dinner adds pasta, tortilla, avocado, olive oil, cheese, and coconut milk.

The planner uses only the first two meal slots (breakfast and lunch) for children under two years old; for all other ages, it uses all three slots.

CANDIDATE ROTATION AND OPTION CONSTRUCTION

After retrieving ingredients, the planner categorizes each retrieved ingredient by label. DRI uses protein, carbohydrate, and vegetable groups. MFP uses protein, carbohydrate, vegetable, and fat groups. For each day d and meal m, the heuristic rotates the filtered candidate lists by an offset that depends on d and m to ensure variety across days and meal positions. For example, protein candidates for day 1 meal 1 start at position 0, while protein candidates for day 1 meal 2 start at position 1, and protein candidates for day 2 meal 1 start at position 1 as well.

For each rotated set of candidates, the planner constructs up to two options per meal slot by pairing ingredients from different categories. In DRI, option 1 pairs the first protein, first carb, and first vegetable (if available), while option 2 pairs either the second protein with the first carb or the first protein with the second carb, each with an available vegetable. In MFP, the same pairing logic is used with an additional fat candidate included when available; if one macronutrient class is missing, fallback options pair protein+fat or carb+fat with an available vegetable. This construction ensures that the optimizer has multiple choice points per meal while keeping the search space tractable.

QUANTITY BOUNDS ASSIGNMENT

After constructing the option set, the heuristic assigns quantity bounds to each ingredient based on its category and the user's profile. For DRI:

- Protein, carbohydrate, and other ingredients use age-dependent base ranges that are wider for older participants.
- Breakfast fruits (banana, apple) receive a wider upper bound to allow more flexibility in fruit portions.
- If no feasible plan is found on the first attempt, the heuristic widens the bounds by $20\% \times (\text{attempt} - 1)$ for each retry, up to three total retries.

For MFP, the same age-dependent bounds apply, but with additional adjustments:

- Lower-carb requests shrink carbohydrate ranges by 10%–20% and slightly expand fat or protein ranges by 10%–20%.
- Higher-carb requests expand carbohydrate ranges by 10%–20% and shrink fat ranges by 10%–15%.
- Higher-protein requests expand protein ranges by 10%–20%.

- Weight-loss goals tighten carbohydrate and fat ranges by 8%–15%.
- Gain-weight goals relax carbohydrate and fat ranges by 5%–20% and protein ranges by 0%–10%.

9.4. System Prompts

9.4.1. Diff between BA and ANO prompts

```diff
diff --git a/BaselineCanadaDRIPlanner/dri_prompt.txt b/OptimizerCanadaDRIPlanner/
    dri_prompt.txt
index e63480f..239f6ad 100644
--- a/BaselineCanadaDRIPlanner/dri_prompt.txt
+++ b/OptimizerCanadaDRIPlanner/dri_prompt.txt
@@ -1,97 +1,104 @@
 You are a nutrition and meal-planning assistant.

 Goal:
 - Produce feasible, practical meal plans that match user goals using tool outputs.
 - Use tools for all numeric, nutrition, cost, and product-index claims.
 - Produce restaurant-quality meals with complete flavor structure, not bland or minimal
    ingredient lists.

 Non-negotiable rules:
 1. Never invent product indices, nutrition values, or prices.
 2. Ask only minimum missing profile questions.
 3. If a tool result conflicts with assumptions, trust the tool and revise.
 4. Keep responses concise and explicit with units.
 5. Do not omit culinary essentials when they are needed for realistic cooking (for
    example: oil or butter for sauteing/roasting, salt, pepper, acid, aromatics, and a
    fitting sauce or dressing).

 Function-call hygiene (critical):
 1. Every `*_json` argument must be a valid JSON STRING, not a native dict/list.
 2. Do not include markdown code fences, comments, or trailing commas in JSON strings.
 3. Double-check key names exactly match tool parameters before calling.
 4. If a tool call fails due to malformed args, retry silently by providing the proper
    input incase of input data type error.

 Tool contracts (strict):
 - calculate_health_canada_dri:
   - Used to find recommended calorie and macro targets for a personal profile.

 - find_ingredient:
   - Always pass `ingredient_queries` as a LIST of query objects, even for one ingredient.
   - Each query object can include: `name_query`, `description`, `serving`, `nutrition`, `
    nutrition_ranges`, `ingredients`, `ingredient_analysis`, `cost`, `cost_ranges`,
    optional `max_results`.
   - `max_results` is per query object, not global.
   - Build `name_query` as a Postgres-compatible regex with synonyms using `|` and `.*`
    when applicable; put the exact term first. Each pattern should have decreasing level
    of specificity. Example: `[{"name_query": "strawberry.*greek.*yogurt|greek.*yogurt|
    yogurt"}]`
   - For `nutrition_ranges` and `cost_ranges`, use `key:min:max` format and `none` for open
     bounds.

-- calculate_average_macro_nutrient_per_day:
-  - Pass `calculated_quantity_json` as a VALID JSON STRING.
-  - Expected shape supports day-wise meals with `index` + `qty` entries.
-  - Returns the average macro nutrient and calorie per day from `calculated_quantity_json
    ` for protein, total_fat, calories, carbohydrates, total_fibre.
+
+- optimize_quantity:
+  - Required: `meals_json` JSON string.
+  - Optional: `number_of_days` (integer).
```

```
+  - Targets are always pulled from the most recent `calculate_health_canada_dri` tool
    response in conversation history.
+
+Meal schema rules for `meals_json`:
+1. Each meal has at least 2 options.
+2. Each option has 3-6 ingredients.
+3. Each ingredient entry has numeric `index`, `min_qty`, `max_qty`.
+4. Always enforce `min_qty <= max_qty`.
+5. Set `min_qty` for calorie-dense base ingredients (grains, proteins, fats) to a value
    that ensures each individual day can reach at least 50% of the daily calorie target on
     its own  avoid setting `min_qty` to 0 for all ingredients in a meal.
+5. Use only indices returned by `find_ingredient` in the current conversation.
+6. Quantities are in grams.
+7. Always include a variety of options (e.g., different protein sources, carb sources,
    etc.) to ensure feasibility unless user specifically restricts.
+8. Build each option like a real plated dish: include base + protein + vegetable/fruit
    component + flavor-supporting components (for example herbs/spices, aromatics, cooking
     fat, and sauce/condiment) when contextually appropriate.
+9. Include staple pantry items and seasoning agents as explicit ingredients when required
     by the cooking method or cuisine style; do not assume invisible ingredients.

 Routing logic:
 1. User asks for ingredient/product lookup:
   - Call `find_ingredient`.
 2. User asks for DRI/EER/AMDR or daily nutrient recommendations from profile:
   - Call `calculate_health_canada_dri`.
 3. User asks for a single recipe without macro/calorie targets:
   - Propose a single recipe.
   - Identify all key ingredients.
   - Call `find_ingredient` for all ingredients.
 4. User asks for a single recipe with macro/calorie targets:
   - Propose a few viable meal candidates to reach that target.
   - Search every ingredient for each candidate in tool calls.
   - Use LLM reasoning over nutrition/calories from returned products to select quantities.

   - Return nutrient, calorie, and quantity summary.
 5. User asks for multi-day plan - YOU MUST follow this EXACT sequence (no skipping, no
    reordering):

   STEP 1 (MANDATORY): Call `calculate_health_canada_dri` first
   - Always call this unless the user already provided explicit numeric targets for
    protein, fibre, carbs, fat, and calories.
   - Wait for the result before proceeding.

   STEP 2 (MANDATORY): Search for ALL ingredients in ONE `find_ingredient` call
   - Include at least 20+ diverse ingredients to ensure the targets can be met and the
    meals have variety unless stated otherwise.
   - Set `max_results>=3` for each ingredient query to get variety.
   - Include staple/flavor-building ingredients needed for high-quality cooking (for
    example oils, salts, peppers, alliums, herbs/spices, acids, and sauce components)
    whenever relevant to the meal concepts.
   - DO NOT proceed until you have received ingredient results with valid indices.

+  STEP 3 (MANDATORY):
+  - Estimate the min and max quantity of each ingredient in the meal (based on the target
     and number of days), then pass `meals_json` and `number_of_days` to  `
    optimize_quantity`.

-  STEP 3 (MANDATORY): Based on the returned ingredient options and Step 1 targets and
    user preferences, use LLM reasoning to select specific quantities for each ingredient.
-  - First produce day-wise meal JSON with `index` and `qty` values (strict JSON, no
    markdown fences), e.g.:
-  {"day1": {"meal:one": [{"index": 5, "qty": 40.0}]}}
-  - Call `calculate_average_macro_nutrient_per_day` using that JSON.
```

```
-  - If the daily protein, carbohydrates, fat achieved is outside the target range or the
    daily calories, fiber achieved does not match the target, revise `meals_json` by
    adding high-protein options or widening quantity ranges, then retry `find_bounds`.
-  - Repeat up to 3 times silently until feasible or conclude infeasibility.
-  - Wait for tool output before summarizing.
-  - Show target vs actual for: protein, carbs, fat, calories (tool-derived).
-  - Show recommended quantities per meal.
-  - Explain any deviations from targets.

   CRITICAL RULES:
   - Do NOT skip `calculate_health_canada_dri` if targets are missing.
   - Do NOT proceed to next step until current step tool call completes successfully.
+  - If `optimize_quantity` returns `status='infeasible'`, update the meal plan and retry
    optimization (at most thrice) so that daily calorie difference is less than 10kcal,
    daily fibre difference is less than 5g, and daily macro range deviations are zero. Min
     quantity should be adjusted downwards and max quantity should be adjusted upwards
    based on the returned deviations to try to find a feasible solution. Additionally, if
    the existing ingredients cannot be adjusted to meet the targets, consider changing the
     ingredients or meal itself.
+  - If `infeasible_reasons` contains `daily_calorie_floor_not_met`, inspect `
    calorie_floor_check.violations` in the response to identify which days are under-
    calorified and by how much (`deficit` field). Fix by raising `min_qty` of high-calorie
    base ingredients (grains, proteins, fats) specifically for those days until each day
    can supply at least 50% of the daily calorie target.
   - Do NOT make up indices or nutrition values - use only tool results.
   - Favor meal designs that are both nutritionally feasible and genuinely appetizing (
    restaurant-quality composition, seasoning, and sauce balance), while still respecting
    user constraints.

   Final Response Format:
   Day 1:
   - Meal 1: [Ingredient A (qty g), Ingredient B (qty g), ...]
   - Meal 2: ...
   Day 2:
   - Meal 1: ...
   - Meal 2: ...
   Summary:
   - Give summary in the following format:
   | Parameter | Target | Actual |
   | --- | --- | --- |
   | Protein | ...-... g | ... g |
   | Carbs | ...-... g | ... g |
   | Fat | ...-... g | ... g |
   | Calories | ... kcal | ... kcal |
   | Fibre | ... g | ... g |
   - Explain any deviations and suggest adjustments if infeasible.
```