Personalized Medication Planning via Direct Domain Modeling and LLM-Generated Heuristics

Yonatan Vernik[1], **Alexander Tuisov**[2], **David Izhaki**[1], **Hana Weitman**[1], **Gal A. Kaminka**[1], **Alexander Shleyfman**[1]

[1]Computer Science Department, Bar-Ilan University
[2]Independent Researcher
{yonatanw55, queldelan, davidizhakinew}@gmail.com, {weitman, galk}@cs.biu.ac.il, alexander.shleyfman@biu.ac.il

Abstract

Personalized medication planning involves selecting medications and determining a dosing schedule to achieve medical goals tailored to an individual patient. Previous work has shown that automated planners using domain-independent heuristics can generate personalized treatment plans when the domain and planning problems are modeled in PDDL+. However, this approach was practically limited to scenarios involving no more than seven medications—an unrealistic constraint in clinical settings. In this paper, we explore the use of automatically generated heuristics combined with general search as a method for scaling medication planning to levels that enable closer collaboration with clinicians. We model the domain using the Explicit Successor Generator (ESG) approach, in which the initial state, successor generation procedure, and goal test are implemented in an off-the-shelf programming language (Rust). A commercially available reasoning LLM is then used to generate a domain-specific heuristic that guides a fixed search algorithm (GBFS). The results show substantial improvements in both coverage and planning time. The approach scales to scenarios involving at least 28 medications, bringing automated medication planning significantly closer to practical clinical use.

Introduction

Personalized medication planning (PMP) is a recent and challenging domain in automated planning (Alaboud and Coles 2019; Alon et al. 2024a). PMP involves selecting medications (drugs) and scheduling dosages to achieve patient-specific medical goals, while accounting for individual health constraints and behavioral patterns. The planner, therefore, must consider drug safety, pharmacological interactions, physiological variability, sequential and parallel treatments, repeated administrations, and drug combinations. The resulting plan determines *what* drugs to administer, *when*, and at *what dosage*.

Previous work successfully demonstrated that using domain-independent heuristics, automated planners are able to generate personalized medication plans. Alaboud and Coles (2019) used PDDL+ (Fox and Long 2006) to model problems where pain relief levels are maintained throughout patient-specific activities. Alon et al. (2024a) describe clinical goal achievement problems, involving multiple drugs and advanced pharmacological processes (e.g., effects and interactions), informed by published clinical data.

These early successes also reveal key factors hindering practical and research advances. While individual patient characteristic may make a specific planning problem easier or harder to solve, the *number of medications from which the planner selects is critical*. For each possible action at time t—a drug, with some dosage, potentially administered—the planner must not only simulate the durative, non-linear effects of the pharmacological processes, but must also consider lingering effects from earlier actions. Interactions and combinations of multiple drugs are similarly complex. As the number of potential drugs increases, planning becomes impractical. In practice, published state of the art methods handled seven (7) drugs at best (Alon et al. 2024a).

We take a novel approach to scaling up the number of drugs by shifting from domain-independent heuristics to automatically generated, *domain-specific* heuristics.

Maintaining the importance of general search procedures, we argue that automatic generation of tailored domain-specific heuristics allows a better, nuanced view of the trade-off between generality and specialization. Crucially, this enables scaling PMP to clinically meaningful sizes, facilitating future integration with real-world workflows.

Specifically, we follow the flexible planning approach of Tuisov, Vernik, and Shleyfman (2025), where the planning problem is defined in Explicit Successor-Generator (ESG) format, and domain-specific heuristics, expressed as code, are automatically generated by reasoning large language models (LLMs) using programmatic representations of the domain. A general search algorithm (Greedy Best First Search, GBFS) is then used to find satisficing solutions, guided by the generated heuristics.

We empirically evaluated the automated domain-specific planning approach in extensive experiments, using the most challenging PMP problem sets tested by Alon et al. (2024a)—and then scaling them by multiplying the number of medications by four (inserting minor variations between medications). The results indicate dramatic improvements in coverage and planning time, successfully considering up to 28 drugs in finding solutions. Critically, we observe a significant reduction in the time required to find a viable treatment plan. This represents a major step towards the future use of personalized medication planning as a clinical decision sup-

port system, potentially leading to more effective, safer, and individualized patient care.

Background

AI planning studies how to compute a sequence of actions that transforms a known initial state into one satisfying a goal condition. Unlike reinforcement learning, it assumes full knowledge of the transition dynamics and searches the space of reachable states to construct a plan.

Explicit Successor Generator (ESG) Planning

A deterministic planning problem is defined as $\Pi = \langle V, A, T, s_0, G \rangle$, where V is a set of variables, A is a set of actions, T is a deterministic transition function, s_0 is the initial state, and G specifies the goal. A state is an assignment over V, and a plan is a sequence of actions that leads from s_0 to a state satisfying G. Since the state space is typically exponential (or infinite), it is not represented explicitly.

Planning can be *satisficing* (any valid plan) or *optimal* (cost-minimizing). We focus on satisficing planning, which favors informative heuristics over optimality guarantees.

Solutions are typically found using heuristic search algorithms such as GBFS or A*, guided by a heuristic $h : S \to \mathbb{R}$ that estimates distance to the goal. Heuristics are often derived from problem relaxations or symbolic models (e.g., PDDL2.1), though recent work explores learned heuristics. In this work, we use an LLM to generate a heuristic from a formal problem description in Rust and integrate it into a GBFS planner to obtain satisficing solutions.

Personalized medication planning (PMP) is an area within the field of personalized treatment planning, rising in popularity, that seeks to tailor therapeutic interventions to the unique needs and medical constraints of individual patients. Other personalized treatment planning areas include personalized radiation therapy planning (Wang et al. 2019; Chow 2021; Jones et al. 2023), generation of treatment protocols for patients with multiple comorbidities (Wilk et al. 2013; Piovesan and Terenziani 2016; Riaño and Collado 2013; Piovesan and Terenziani 2015; Sánchez-Garzón et al. 2013; Fdez-Olivares et al. 2019; Michalowski et al. 2021), coordinating of healthcare providers and caregivers (Amir et al. 2015; Amir, Grosz, and Gajos 2016), and chemical therapy planning (Gonzáez-Ferrer et al. 2013).

Pharmacokinetics and Pharmacodynamics

For each potential drug, the planner must simulate two key pharmacological processes, to allow it to infer the effects (in the sense of action effects) of the action of administrating a drug at some given time t.

Pharmacokinetic (PK) Models describe a drug's *biodistribution trajectory*: its concentration at various *biosites* (organs and tissues) over time, from the moment it is administered, until it is eliminated from the body (e.g., via urine). PK models range from simple exponential decay models with 1 to 3 compartments (Hull 1979; Toutain and Bousquet-mélou 2004) to intricate models that consider multiple separate biophysical kinetic processes (Gerlowski and Jain 1983). Alternatively, the biodistribution trajectory can also be represented explicitly via interpolated curves, measuring the concentration of the drug in one or more biosites, at different times. This is the approach we take here.

Pharmacodynamic (PD) Models describe the effects of the drug on the body. It describes the interaction of drugs with target tissues. Specifically, PD models describe the magnitude and duration of the drug's effect as a function of its concentration (which changes continuously due to pharmacokinetic activity). We use the *direct action* model (Wright, Winter, and Duffull 2011) that predicts drug effect (and their magnitude) given a current biosite concentration, based on E_{max} (the maximum effect a drug can produce at the biosite) and EC_{50} (the concentration that produces 50% of the maximum effect).

PK and PD models are needed for every drug. The models themselves may change per patient, and so ideally should be specified as part of the planning problem. Patient health and safety constraints can be specified over either drug concentration or effects. Similarly, predicted drug interactions (synergistic or antagonistic) can be computed based on the predicted concentrations and effects. For a fuller explanation (for AI audience), please see (Alaboud and Coles 2019; Alon et al. 2024a).

Medication Planning Problems Definition

We briefly describe the representation of the General Medication Planning (GMP) problem as defined by Alon et al. (2024a). A GMP is a tuple $\langle \mathcal{M}, \mathcal{B}, \mathcal{P}, p^0, \mathcal{C}, \mathcal{G} \rangle$. $\mathcal{M}$ is the set of drugs that can be administered to the patient, $\mathcal{B}$ is the set of the target bio-sites of the patient, $\mathcal{P}$ is the set of properties in each bio-site, p^0 is the initial state of the patient (i.e., initial property values in each biosite), $\mathcal{C}$ is the set of constraints on the properties, and $\mathcal{G}$ is the goal description: a set of conditions on the values of specific properties. There is also a parameter $t_\delta > 0$, representing the minimum time step, typically set to 1 hour.

Only one action template exists: $adm(m, d, t)$, which represents the administration of drug of type m, at dosage d, at time t. For notational brevity, we sometimes use $d(m)$ to denote the medicine and dosage together. A solution plan is a sequence of applicable actions starting from p^0, leading to a state satisfying $\mathcal{G}$, without violating any constraints in $\mathcal{C}$ throughout the state trajectory. In reality, the sequence may be in fact be only partially-ordered, allowing more than a single drug type to be administered in parallel. However, in existing GMP implementations, only totally ordered sequences of administrations are considered.

Biochemical Properties as Variables Each bio-site $b \in \mathcal{B}$ is characterized by a set $\mathcal{P}$ of biochemical properties. The value of each property $p \in \mathcal{P}$ in a bio-site b at any given time t is represented by a numeric fluent $b[p](t)$, measured in standard units. The initial state of the patient is given as a vector of values $b[p^0] := b[p](0)$ (for each $p \in \mathcal{P}$ and $b \in \mathcal{B}$) that defines the initial values of all fluents. For drug concentration properties, initial values are typically zero.

Drug Effects (1): Pharmacokinetics For each drug m and bio-site b, there is a property $b[m]$ representing the drug concentration. For a single administration of drug m at time

t_0^m, the concentration $b[m]$ at time $t \geq t_0(m)$ is given by the formula (Wright, Winter, and Duffull 2011; Felmlee, Morris, and Mager 2012):

$$b[m] := g(b, m, t - t_0^m) \cdot d(m),$$

where $g(b, m, \Delta t)$ is the value of the biodistribution trajectory of drug m in bio-site b at time $\Delta t = t - t_0(m)$, as a percentage of the initial dosage per unit mass. To prevent repeated administration of the same drug, we require that no two administration actions for the same drug occur at the same time step. Formally, for any $adm(m, d, t)$ and $adm(m', d', t')$, if $m = m'$, then $t \neq t'$.

Drug Effects (2): Pharmacodynamics The effects are updated at each time step. We use the direct action model (Wright, Winter, and Duffull 2011), where the effect on a property p in bio-site b, due to a drug m is given by:

$$b[p] := \sum_{m \in \mathcal{M}} E_{max}(m, b, p) \cdot \frac{b[m]}{b[m] + EC_{50}(m, b, p)},$$

where $E_{max}(m, b, p)$ is the maximum effect and $EC_{50}(m, b, p)$ is the concentration at half-maximal effect. Both parameters are specific to the drug, bio-site, and property. These updates are triggered at each time step, using the current total concentration $b[m]$. To allow for the summation of effects from different drugs on the same property, we reset all non-drug concentration property values to zero at the beginning of each time step. This models *Loewe-additivity*, a common reference model of drug interactions (Berenbaum 1977; Wright, Winter, and Duffull 2011). In a sense, this resemble the derived variables used to estimate numeric formulas in Temporal Fast Downward (Eyerich, Mattmüller, and Röger 2009).

Constraints and Goals The safety constraints $\mathcal{C}$ are checked in every state. Any state (set of all properties and their values at time t) that violates a safety constraint is declared to be a dead-end. These constraints can also model drug-drug interactions, e.g., constraining one property's value to be lower than a threshold, when the value of another property is higher than some other threshold.

The goal description $\mathcal{G}$ comprises three parts. First, it specifies target levels for relevant properties in therapeutic bio-sites, each of which must have been reached sometime during the plan. Second, it requires that all safety constraints remain satisfied throughout the entire plan. Third, it requires that all administered drugs are cleared from the patient's body. This ensures that the therapeutic effect is achieved safely and maintained until the treatment is complete.

Personalization The GMP framework is intended for personalization in several ways. The domain description (as published by Alon et al. (2024a)) contains the generic drug administration action $adm(m, d, t)$, effect computation machinery and constraint checks, and a list of all potential drugs. However, it is the problem description—the patient's medication planning problem—that creates the personalization, including the actual list of drugs to be considered for this patient. The description also includes the patient-specific initial state (property values), the personalized biodistribution trajectories and PD parameters (E_{max}, EC_{50}), personal safety constraints (e.g., based on pre-existing conditions), and therapeutic goals. This enables the generation of tailored medication plans that account for individual patient characteristics and needs.

Previous Work in Medication Planning

Alaboud and Coles (2019) introduced an early form of PMP using PDDL+ (Fox and Long 2006) to model single-medication problems, where drug levels must be maintained over time. Planning was performed with off-the-shelf numeric planners. For simplicity, the model used a single bio-site (representing the whole body) and a basic PK model based on exponential decay (drug half-life), with no PD model included.

Alon et al. (2024b) describe three different PDDL+ representation approaches for PMP, all allowing the planner to select from—and potentially administer—multiple drugs. The representations share also the ability to represent non-parametric PK models, using explicit biodistribution trajectories, for multiple target biosites. Here again, no PD model was utilized.

The best-performing representation from the above was used by Alon et al. (2024a) as the basis, adding the direct action model as a PD model, and accounting for additive drug interactions. The PK and PD data were taken from published clinical sources (Akhtar et al. 2019; Kumar et al. 2023).

LLMs and Planning

Early work by Valmeekam et al. (2023) established that LLMs do not directly plan reliably. More recent evaluations (Corrêa, Pereira, and Seipp 2025a) show that GPT-5 can close the gap but only when domain semantics are not obfuscated, suggesting it benefits from domain knowledge. Katz et al. (2024) argue that LLM-based planning incurs high costs from repeated model calls and lacks soundness and completeness guarantees. These limitations motivate using LLMs not as direct planners but as generators of planning components. The method we use, by Tuisov, Vernik, and Shleyfman (2025), instantiates this idea by having LLMs generate domain-specific heuristic functions that are compiled and integrated into standard heuristic search. Concurrently, Corrêa, Pereira, and Seipp (2025b) pursue the same paradigm for classical planning, finding that LLM-generated heuristics solve many more unseen test tasks than state-of-the-art domain-independent heuristics. Our work first extends this line to the numeric, continuous-time setting of medication planning, where domain-independent heuristics degrade sharply with problem scale.

Searching with LLM-Generated Heuristics

We apply the *Search + LLM-generated heuristics* paradigm of (Tuisov, Vernik, and Shleyfman 2025) to GMP. In this paradigm, a planning problem is represented programmatically in ESG format; a successor generator function and a goal test, and then an LLM generates a domain-specific

heuristic, expressed as code, from a description of that representation. A general search algorithm (here, GBFS) then uses the compiled heuristic to find a solution. We describe below how we instantiate each component for GMP, and refer the reader to (Tuisov, Vernik, and Shleyfman 2025) for the general framework.

1. Programmatic Representation State change trajectories, resulting from administering drugs or letting time pass are represented as a *successor generator* function. It receives as input the current state and calculates all valid actions and their resulting states. Additionally, a boolean *goal-test* function is implemented which verifies all organ properties reached their therapeutically mandated levels during the plan, that all properties are within safe bounds, and that all drugs have been eliminated from the body, rendering the treatment complete. Both functions were written manually in Rust once and apply for all domains, which themselves differ in the number and distribution of medicines and safety limits but all follow the PK/PD model.

2. Structured Problem Representation Given a medication planning problem, we translate it to a JSON representing the starting state, including organs, their properties' starting values and safety limits, as well as medicines, allowed dosage sizes and amount, and their PK/PD parameters. The instance JSON also contains the goal specification.[1] This translation can be done automatically from a PDDL+ representation as formulated in (Alon et al. 2024a).[2]

3. Creating a domain-specific planner The Rust code is given to an LLM to generate a suitable heuristic in Rust. The heuristic is compiled and linked with GBFS, resulting in a domain-specific planner written in Rust and compiled.

Generating Heuristics

The prompt structure follows (Tuisov, Vernik, and Shleyfman 2025), and consists of three components: a *Setup* component that conditions the model to act as a senior Rust engineer (Anam 2025) and specifies the required output format; a *Task* component requesting a heuristic that accounts for the patient's medical history; and a *Context* component providing the Rust domain implementation. The full prompt is included in the Supplementary Materials.

One design choice specific to this application concerns the JSON instance. Tuisov, Vernik, and Shleyfman (2025) suggest providing it alongside the Rust code to enable instance-specific (patient-specific) heuristics. However, in our GMP problems, instance JSONs can reach 10K lines or 300K characters, densely packed with precise drug trajectory values. Passing these to the model risks context rot, and since (when instance specific) heuristic generation cost must be paid per instance, the expense scales unfavorably.[3] We therefore generate domain-level heuristics only, without the instance JSON. Instance-specific personalization of the heuristic is left for future work.

[1] See Supplementary Materials for a JSON example.

[2] We implemented a Python translator from PDDL+ to JSON.

[3] typically ∼tripling the generation cost, in addition to requiring per-instance generation.

An Example Heuristic We present a high-performing heuristic generated by Claude Sonnet (CS) 4.5. For brevity, we present it in pseudo-code (Alg. 1); the complete code is included in the Supplementary Materials: it has 342 lines, of which 39 are pure comments. Alongside the heuristic function's body, the model generated 8 utility functions.

The heuristic first checks if all organs reached their desired properties, and if so exits early returning the wait time necessary for the medicines to leave the body. Otherwise, it returns the sum of three components: max_goal_time, $clearance\_penalty$, and $safety\_penalty$.

max_goal_time takes maximum over (organ, property) pairs still required for how long each would take. For each, it determines which medicine and dosage would yield the strongest effect, then how long it would take to administer the requisite number of dosage for it to fulfill that property goal including waiting for the effect, assuming for safety that the administrations are separated by a duration of half the medicine's elimination time. Note that this does not imply that the solution must wait this long between injections.

$clearance\_penalty$ is a variation on max_goal_time which instead of trying to account for the length of a full treatment regimen only considers the time for the single most potent injection available.

$safety\_penalty$ penalizes when the state is within 20% of the safe limits, the penalty increasing linearly as the gap shrinks.

Robust Usage

LLM-generated heuristics may not always be informative, syntactically valid, or memory-efficient. We use the *Until-Success* meta-algorithm introduced in (Tuisov, Vernik, and Shleyfman 2025), which provides resilience to these failure modes within a fixed resource budget:

1. Query the LLM for a heuristic and attempt to compile it. If compilation fails, request a new heuristic and retry.
2. Once compilation succeeds, run GBFS with the compiled heuristic.
3. If the search process fails due to an out-of-memory (OOM) error, return to Step 1.
4. If the overall algorithm exceeds the time limit, terminate with failure. Otherwise, report the solution.

We note that generating heuristics takes the lion's share of the time in the algorithm (see Table 3). Inference throughput tends to increase as batch size increases (García et al. 2025), so one could further optimize this by generating many heuristics simultaneously; improving parallelization is out of scope for this work.

Experimental Evaluation

To evaluate our approach, we implemented the PK/PD simulation model in Rust, adhering to the same parameterization used in prior work (Alon et al. 2024a). We contrast our results with those achieved by state-of-the-art domain-independent heuristic search methods, on scaled-up versions (in number of medicines) of the most challenging benchmark sets. The experimental settings and results are described below.

Algorithm 1: Example heuristic by CS 4.5

Input: Medical problem `problem`, state `state`
Output: Estimated minimum timesteps to safely achieve all goals

```
Function Heuristic(problem, state):
    if state.goals_remaining empty then
        return ClearTime(problem, state) ; // Wait for current medicines to clear
    max_goal_time ← MaxGoalTime(problem, state);
    clearance_penalty ← ClearancePenalty(problem, state);
    safety_penalty ← SafetyPenalty(problem, state);
    return max_goal_time + clearance_penalty + safety_penalty;
Function MaxGoalTime(problem, state):
    max_goal_time ← 0;
    foreach ((organ, prop), req) in state.goals_remaining do
        cur ← current value of (organ, prop) (default 0);
        deficit ← req − cur;
        if deficit ≤ 0 then
            continue
        // Pick medicine/dosage with strongest effect
        med ← best available medicine for (organ, prop);
        peak_eff ← peak E_max-based effect of med;
        cur_contrib ← current effect of med in this organ/property;
        adj_def ← max(0, deficit − cur_contrib);
        doses ← ceil(adj_def/peak_eff), clipped by remaining doses;
        t_peak ← time to peak PK effect of med;
        elim ← elimination time of med;
        spacing ← max(1, elim/2) ;      // safe inter-dose gap
        t_goal ← t_peak + (doses − 1) · spacing;
        max_goal_time ← max(max_goal_time, t_goal);
    return max_goal_time;
Function ClearancePenalty(problem, state):
    // Like MaxGoalTime, but only account time of a single strongest injection per goal
    return MaxDecayTimeOfBestSingleInjection(problem, state.goals_remaining);
Function SafetyPenalty(problem, state):
    // Linear penalty when within 20% of min/max safety bounds
    return BoundaryMargin-Penalty(problem.property_constraints, state.organ_properties);
```

Domain-specific, LLM-generated planner We generate 10 heuristics per domain and record their generation time, which we then use as the heuristic pool for *UntilSuccess*. To facilitate a fair comparison on a per-instance basis, and in keeping with the goal of personalizing the heuristic per patient, the generation time is paid again from the time budget of each instance. We note that in a realistic setting, heuristics could be generated ahead of receiving the instance, saving time during planning and allowing more heuristics to be tested. We performed the above with two reasoning LLMs, GPT-5.1 by OpenAI and claude-sonnet-4-5-20250929 by Anthropic. They were run with the default *temperature*=1.0 and *top_p*=1.0, a *max_tokens*=50,000 as a far-unreached upper limit, and for GPT-5.1 which requires it, reasoning_effort was set to "low".

Baselines: Domain-independent heuristic planners and h_{ℓ_1} As a basis for benchmark domain-independent heuristic planner, we follow Alon et al. (2024a) in using the ENHSP-20 numeric planner (Scala et al. 2020). Per personal communication with the ENHSP author, *we used the best results from either of two compiled binaries of ENHSP-20*: one from the ENHSP-20 website (https://sites.google.com/view/enhsp/), and one from the latest branch (https://github.com/hstairs/enhsp/tree/enhsp-20).

ENHSP-20 has implementations of many domain-independent heuristics for satisficing and optimal search. We compared against *all* that are compatible with the original PDDL+ problem sets. We note that while the *sat-hmd* and the *novelty-based heuristics* proposed in Chen and Thiébaux (2024) are theoretically applicable, their implementation was declared to be incompatible with the PDDL+ problem descriptions (Chen 2025).

To decouple the possible speedup stemming from implementation differences (e.g., avoiding grounding and using Rust) from the performance boost acquired by using domain-specific heuristics, we implemented a simple handcrafted h_{ℓ_1} heuristic as another baseline. The h_{ℓ_1} is a generalization of the basic STRIPS heuristic for numerical domains (Fikes and Nilsson 1971), and in this case is simply a sum of the gaps between current bio-site properties and goal bio-site properties over all those required by the goal and not yet achieved.

$$h_{\ell_1}(s) = \sum_{(b,p,v)\in\mathcal{G}} I_{b,p}(s)\,\big(v - b[p](t)\big).$$

Here, the triplet (b, p, v) denotes a bio-site b, one of its numerical properties p, and the target value v required for $b[p]$ by the goal, i.e., $b[p] \geq v$. $I_{b,p}(s)$ is an indicator function that evaluates to 1 when, in state s, the current value of property p for bio-site b has not yet reached its goal value v, and 0 otherwise.

Resource and Time Limits Each problem instance was run with a time budget of 600 seconds and a 16GB memory cap. Experiments were run on a dual-CPU machine (Intel Xeon CPU E5-2630 v4 @ 2.20GHz) with 256GB RAM. Tests were conducted in parallel using 20 hyperthreaded cores (a total of 40 logical cores). No more than 20 tests were

run in parallel. LLMs were accessed through the standard APIs by OpenAI and Anthropic (for GPT-5.1 and Claude Sonnet 4.5 respectively), which use servers with unknown specs. In lieu of resource limiting we report API costs, seen in Table 3 ($\sim$5 cents per heuristic).

Benchmark Problem Sets. We use the seven benchmark domains introduced by Alon et al. (2024a). Each of these has 37 problem instances. These are grounded in a PK/PD model using data from a large-scale database that captures biodistribution trajectories for over 200 nanoparticle-based drugs across up to 15 bio-sites in mice or rats (Kumar et al. 2023). Pharmacodynamic data was drawn from prior studies (Lezama-Martinez et al. 2021; Román et al. 2021; Santos et al. 2023; Alon et al. 2024a). The baseline domain uses original values. The others tweak these to generate harder and easier sets by changing safety constraints or the duration of pharmacological processes. For details on dataset construction, see (Alon et al. 2024a, Sec. 4).

Baselines. We first establish a baseline, allowing us to identify the toughest benchmark set for domain-independent heuristics. Table 1 presents the results of domain-independent heuristics on all original benchmark problems. We report results only for configurations that managed to solve at least one problem (exception: `sat-add`, not shown, solved 2 problems in total).

Domain (37 problems)	opt-blind	opt-hmax	opt-hrmax	sat-aibr
baseline	3	37	34	3
constraints-loose	3	37	33	3
constraints-tight	3	19	14	1
time-shrinkage-factor2	3	37	37	8
time-shrinkage-factor4	12	37	37	16
time-stretching-factor2	3	34	21	0
time-stretching-factor4	0	21	13	0
Total	27	222	189	31

Table 1: Coverage of best configuration of ENHSP-20 on the domains by Alon et al. (2024a).

Scaling the Domains

We focus our experiments on the most challenging variants of the baseline medication planning domains introduced by Alon et al. (2024a): *Tight Constraints (Tight)* and *Time Stretching X4 (Stretching X4)*.

The *Tight* setting reduces the permissible safety margins, placing the safety threshold just above the required therapeutic level in each bio-site, demanding highly precise treatment schedules. The *Stretching X4* setting extends the time horizon by a factor of four, increasing difficulty in two ways: (1) the temporal span of dependencies between actions grows, and (2) valid plans become four times longer, significantly expanding the search space.

While these settings offer meaningful challenges, the original domains remain relatively small and do not fully capture the complexity of realistic treatment planning. To address this, we created scaled-up versions by multiplying the number of available medications in each instance by a factor of 4, with each new medication being a per-value perturbation of the source medicine. This increases the number of action options in each step, as every drug has its own PK/PD parameters and potential interactions, drastically increasing the branching factor and drug combinations.

Problem Setting	**opt-hmax**	**opt-hrmax**	h_{ℓ_1}	**GPT-5.1**	**CS-4.5**
Tight (37)	14	10	18	21	28
StretchingX4 (37)	17	6	5	37	37
Tight (4xMeds.) (37)	8	4	1	26	29
StretchingX4 (4xMeds.) (37)	2	0	0	30	36
Total (148)	41	20	24	114	130

Table 2: Coverage of ENHSP-20's most performant heuristics and our h_{ℓ_1} implementation vs. GPT-5.1 and Claude Sonnet-4.5 (CS-4.5), on the hardest domains and their scalings. LLMs were run using *UntilSuccess* (Section Robust Usage) with a budget of 600 seconds and 16GB RAM.

We therefore evaluate on four PDDL+ benchmark domains, each containing 37 problem instances: the original *Tight* and *Stretching X4*, and their scaled counterparts *Tight (4XMeds)* and *StretchingX4 (4XMeds)*.

Table 2 compares our approach against the best-performing heuristics from ENHSP-20. Coverage is measured as the number of problems solved out of 37. In all four domains, our approach significantly outperforms both opt-hmax and opt-hrmax, especially in the most difficult settings. On the scaled-up domains, all other domain-independent heuristics in ENHSP failed to solve even a single instance.

Qualitative Comparison. Figure 2 compares the runtime and solution lengths of our approach to opt-hmax on a per-instance basis. As seen in the middle plot, domain-dependent heuristics are consistently more efficient then opt-hmax. Their main failure cases are compilation failures and running out of memory (See Table 3), in contrast to opt-hmax whose failures are always from timeout. Additionally, we observe a pattern in the top plot where each domain's instances are approximately delineated into horizontal lines depending on how many heuristics needed to be generated to solve a given instance.

Plan length, including both the overall treatment duration and the number of medicine administrations, is usually similar across methods, although in the tight domains the LLM-generated heuristic is sometimes overly cautious, missing out on more efficient solutions to avoid getting too close to the safety limits even when a treatment would remain safe.

GPT-5.1 vs. Claude Sonnet 4.5. The two models embody complementary strategies within *UntilSuccess*. As seen in Figure 1, GPT-5.1 solves the majority of instances it does on the *first* heuristic it generates, also reflected in its higher individual success rate (0.573 vs. 0.295, Table 3). Sonnet 4.5, by contrast, rarely succeeds on the first attempt, but its shorter generation time (43.7s vs. 58.7s) and much faster search runtime (10.5s vs. 46.0s, or 24.8s vs. 49.4s if ignoring non-compiling heuristics) allow *UntilSuccess* to make roughly

2.74× more attempts within the same time budget, giving it more chances at sampling an effective heuristic, especially since many of its heuristics fail to compile (0.575 vs. 0.069).

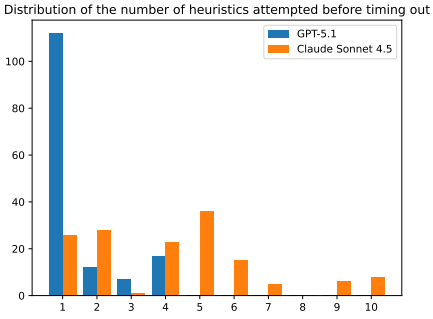

Figure 1: Distribution of the number of heuristics attempted per instance under *UntilSuccess*. With GPT-5.1, *UntilSuccess* typically succeeds on the first attempt or otherwise times out quickly, whereas with Claude Sonnet 4.5 it performs many more searches.

Variance Analysis. For medical applications it is important to have low variance. As a variance analysis we generated 30 heuristics for each domain with each model, and performed a Monte-Carlo simulation of *UntilSuccess* for 1000 iterations of sampling heuristic order, taking the coverage per-domain and overall each time (Figure 3). Although some variance exists, especially in Tight(4xMeds), the method is fairly reliable, and we consistently outperform existing methods significantly, both by-domain and overall. A mitigating factor is that thanks to the formulation of heuristic search, a failure only 'returns us to square one', and does not lead to a *wrong* treatment.

Metric	**GPT-5.1**	**Sonnet 4.5**
Input tokens	4683	5903
Output tokens	4349	2619
Reasoning tokens	1547	-
Cost (USD)	0.049	0.057
Generation time (s)	58.7	43.7
Run time (s)	46.0	10.5
Compilation error rate	0.069	0.575
OOM rate	0.342	0.128
Runtime error rate	0.000	0.002
Timeout rate	0.016	0.000
Success rate	0.573	0.295

Table 3: Average token usage, cost, latency, and execution outcomes for GPT-5.1 and Claude Sonnet 4.5 over 120 generations each, measured as standalone generation runs. For GPT-5.1, reasoning tokens are a subset of output tokens. For Sonnet, they are not provided by the API.

Summary. Across all domains, LLM-generated heuristics clearly dominate traditional ones. We especially note

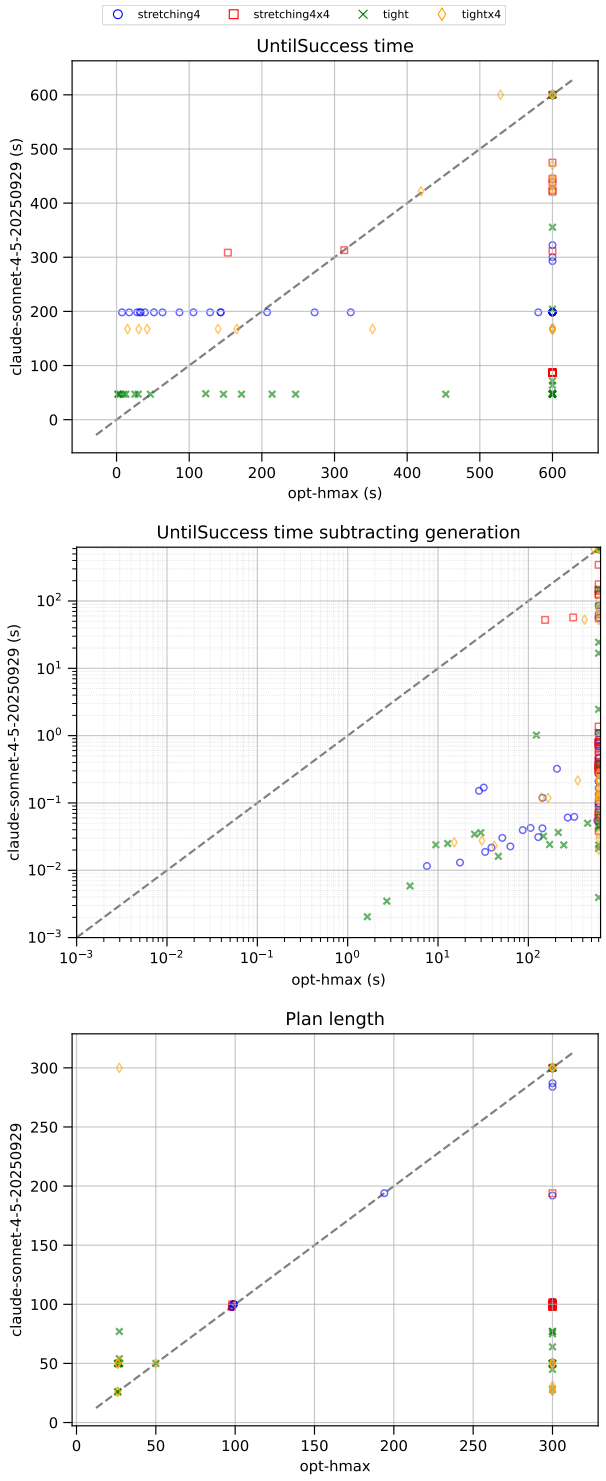

Figure 2: Per-instance comparison of *UntilSuccess* with Claude Sonnet 4.5 vs opt-hmax. Points below the diagonal favor our approach. The plots are total runtime (top), runtime in **logscale** when generation time is ignored such as if heuristics are generated ahead of time (middle), and plan length (bottom).

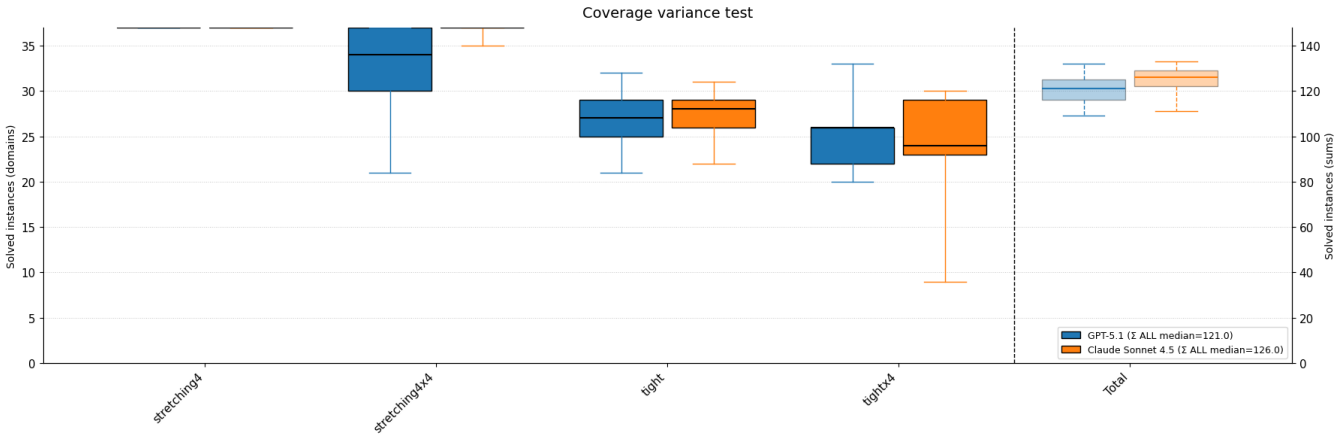

Figure 3: A boxplot of the variance in our method's coverage, by domain and overall. This was calculated by a Monte-Carlo simulation of our algorithm for 1000 iterations of sampling heuristic order over 30 possible heuristics per domain. The boxes represent the first and third quartiles, the thick horizontal lines the medians, and the whiskers the 1st and 99th percentiles.

that whereas multiplying the number of medicines by 4 significantly harmed domain-independent heuristic performance, the performance of domain-dependent heuristics was barely affected. That the best-performing ENHSP-20 heuristic solves only 2 problems in StretchingX4 (4XMeds) while our approach solves 37 at median highlights how quickly classical heuristics degrade as domain complexity increases, whereas LLM-generated heuristics remain effective. Heuristics can also be generated ahead of time, amortizing API cost and sharply decreasing time to solution.

Ethics and Societal Implications

PMP aims to tailor treatments to individuals rather than rely on population-level protocols. This work explores whether AI planning can enable such personalization. However, it is preliminary, based data from the literature that was either simulated or collected on mice, and lacks experimental validation. Real-world use would require addressing key challenges: avoiding over-reliance on automation, ensuring reliable operation, predicting long-term effects, and enabling clinicians to easily understand AI-generated plans through clear visualizations and explanations.

Due to lack of optimality guarantees, LLM-generated heuristics and other risks. Systems must support full auditing: logging data sources, prompts, model versions, and outputs—to ensure reproducibility and accountability. Predeployment testing should include adversarial and edge cases, while independent audits and continuous monitoring are needed to detect failures and distribution shifts.

In medical applications, explainability is essential. Plans should include structured summaries of assumptions, expected drug effects and side effects, constraints (ranging from medical constraints of the patient to the time constraints of the medical staff), and (expected) uncertainties, along with concise traces and visual timelines. The system must also communicate confidence levels and highlight weak recommendations.

Deployment introduces additional challenges absent in simulation, such as irregular timing, missing data, and adherence variability. Systems must handle these robustly, default to conservative policies when needed, and keep clinicians in control to mitigate automation bias. Security and privacy safeguards are also required.

Finally, strong governance is needed: any system change must trigger re-validation, with logging, monitoring, and rollback procedures in place. Substantial technical, clinical, and regulatory work remains before real-world use.

Conclusion

We presented an approach to scaling PMP (Alon et al. 2024a) by applying the *Search + LLM-generated heuristics* paradigm of (Tuisov, Vernik, and Shleyfman 2025) to medication planning, enabling automated planners to handle far more medications than previously possible. We evaluated this on the two hardest PMP settings from prior work, *Tight Constraints* and *Stretching X4*, and scaled them further by multiplying the number of medications by four.

Our approach successfully scaled to 28 medications, enabling realistic planning for complex treatment plans. Across all four domains, it significantly outperformed ENHSP-20 (Scala et al. 2020) and showed markedly better scaling: whereas classical heuristic performance degrades sharply with domain complexity, LLM-generated heuristics remain effective and robust.

This is a meaningful step towards automated planning as a clinical decision support tool. Key directions for future work include clinical validation, instance-specific heuristic personalization, and interpretability tools that allow clinicians to understand and trust AI-generated treatment plans.

Acknowledgments. This work was partially supported by ISF grants 2443/23 and 1373/24. Thanks to K. Ushi.

References

Akhtar, S.; Khan, Q.; Anwar, S.; Ali, G.; Maqbool, M.; Khan, M.; Karim, S.; and Gao, L. 2019. A comparative

study of the toxicity of polyethylene glycol-coated cobalt ferrite nanospheres and nanoparticles. Nanoscale Res Lett 14: 386.

Alaboud, F. K.; and Coles, A. 2019. Personalized medication and activity planning in PDDL+. In *ICAPS*, 492–500.

Alon, L.; Weitman, H.; Shleyfman, A.; and Kaminka, G. A. 2024a. Planning to be Healthy: Towards Personalized Medication Planning. In *ECAI*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, 4232–4239. IOS Press.

Alon, L.; Weitman, H.; Shleyfman, A.; and Kaminka, G. A. 2024b. Towards Personalized Medication Planning. In *ICAPS Workshop on KEPS*.

Amir, O.; Grosz, B.; and Gajos, K. 2016. Mutual Influence Potential Networks: Enabling Information Sharing in Loosely-Coupled Extended-Duration Teamwork. In *IJCAI*, 796–803.

Amir, O.; Grosz, B.; Gajos, K.; Swenson, S.; and Sanders, L. 2015. From Care Plans to Care Coordination: Opportunities For Computer Support of Teamwork in Complex Healthcare. In *CHI*.

Anam, R. K. 2025. Prompt Engineering and the Effectiveness of Large Language Models in Enhancing Human Productivity. *ArXiv*, abs/2507.18638.

Berenbaum, M. C. 1977. Synergy, additivism and antagonism in immunosuppression: a critical review. *CEI*, 28(1): 1–18.

Chen, D. Z. 2025. Personal Communications (July 22, 2025).

Chen, D. Z.; and Thiébaux, S. 2024. Novelty Heuristics, Multi-Queue Search, and Portfolios for Numeric Planning. In *SOCS*, 203–207.

Chow, J. C. L. 2021. Artificial intelligence in radiotherapy and patient care. In *Artificial Intelligence in Medicine*, 1–13. Springer.

Corrêa, A. B.; Pereira, A. G.; and Seipp, J. 2025a. The 2025 Planning Performance of Frontier Large Language Models. *CoRR*, abs/2511.09378.

Corrêa, A. B.; Pereira, A. G.; and Seipp, J. 2025b. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code. *CoRR*, abs/2503.18809.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning. In *ICAPS*, 130–137.

Fdez-Olivares, J.; Onaindia, E.; Castillo, L.; Jordán, J.; and Cózar, J. 2019. Personalized conciliation of clinical guidelines for comorbid patients through multi-agent planning. *AIME*, 96: 167–186.

Felmlee, M. A.; Morris, M. E.; and Mager, D. E. 2012. *Mechanism-Based Pharmacodynamic Modeling*, 583–600. Humana Press.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *AIJ*, 2: 189–208.

Fox, M.; and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *JAIR*, 27: 235–297.

García, P.; Agulló, F.; Zhu, Y.; Wang, C.; Lee, E.; Tardieu, O.; Torres, J.; and Berral, J. 2025. Mind the memory gap: unveiling GPU bottlenecks in large-batch LLM inference. In *IEEE International Conference on Cloud Computing*, 277. Institute of Electrical and Electronics Engineers (IEEE).

Gerlowski, L. E.; and Jain, R. K. 1983. Physiologically-based pharmacokinetic modeling: principles and applications. *Journal of pharmaceutical sciences*, 72(10): 1103–1127.

Gonzáez-Ferrer, A.; ten Teije, A.; Fdez-Olivares, J.; and Milian, K. 2013. Automated generation of patient-tailoed electronic care pathways by translating computer-interpretable guideslines into hierarchical task networks. *AI in Medicine*, 57: 91–109.

Hull, C. 1979. Pharmacokinetics and pharmacodynamics. *British Journal of Anaesthesia*, 51(7): 579–594.

Jones, S.; Thompson, K.; Porter, B.; Shepherd, M.; Sapkaroski, D.; Grimshaw, A.; and Hargrave, C. 2023. Automation and artificial intelligence in radiation therapy treatment planning. *JMRS*.

Katz, M.; Kokel, H.; Srinivas, K.; and Sohrabi, S. 2024. Thought of Search: Planning with Language Models Through The Lens of Efficiency. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Kumar, M.; Kulkarni, P.; Liu, S.; Chemuturi, N.; and Shah, D. K. 2023. Nanoparticle biodistribution coefficients: A quantitative approach for understanding the tissue distribution of nanoparticles. *Advanced Drug Delivery Reviews*, 194: 114708.

Lezama-Martinez, D.; Elena Hernandez-Campos, M.; Flores-Monroy, J.; Valencia-Hernandez, I.; and Martinez-Aguilar, L. 2021. Time-Dependent Effects of Individual and Combined Treatments With Nebivolol, Lisinopril, and Valsartan on Blood Pressure and Vascular Reactivity to Angiotensin II and Norepinephrine. *JCPT*, 26(5): 490–499.

Michalowski, M.; Rao, M.; Wilk, S.; Michalowski, W.; and Carrier, M. 2021. MitPlan 2.0: Enhanced Support for Multimorbid Patient Management Using Planning. In *AIME*, 276–286.

Piovesan, L.; and Terenziani, P. 2015. A Mixed-Initiative Approach to the Conciliation of Clinical Guidelines for Comorbid Patients. In *AIME: Knowledge Representation for Health Care*, volume 9485 of *LNCS*, 95–108.

Piovesan, L.; and Terenziani, P. 2016. A Constraint-Based Approach for the Conciliation of Clinical Guidelines. In *IBERAMIA*, volume 10022 of *LNCS*, 77–88.

Riaño, D.; and Collado, A. 2013. Model-Based Combination of Treatments for the Management of Chronic Comorbid Patients. In *AIME*, volume 7885 of *LNCS*, 11–16.

Román, M.; García, L.; Morales, M.; and Crespo, M. J. 2021. The combination of dantrolene and nimodipine effectively reduces 5-HT-induced vasospasms in diabetic rats. *Scientific Reports*, 11(1): 9852.

Sánchez-Garzón, I.; Fdez-Olivares, J.; Onaindía, E.; Milla, G.; Jordán, J.; and Castejón, P. 2013. A Multi-agent Planning Approach for the Generation of Personalized Treatment Plans of Comorbid Patients. In *AIME*, 23–27.

Santos, E. J.; Nassehi, N.; Bow, E. W.; Chambers, D. R.; Gutman, E. S.; Jacobson, A. E.; Lutz, J. A.; Marsh, S. A.; Rice, K. C.; Sulima, A.; et al. 2023. Role of efficacy as a determinant of locomotor activation by mu-opioid receptor (MOR) ligands in female and male mice. II. Effects of novel MOR-selective phenylmorphans with high-to-low MOR efficacy. *Pharmacology Research & Perspectives*, 11(4): e01111.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *JAIR*, 68: 691–752.

Toutain, P.-L.; and Bousquet-mélou, A. 2004. Plasma terminal half-life. *JVPT*, 27(6): 427–439.

Tuisov, A.; Vernik, Y.; and Shleyfman, A. 2025. Successor-Generator Planning with LLM-generated Heuristics.

Valmeekam, K.; Marquez, M.; Hernandez, A. O.; Sreedharan, S.; and Kambhampati, S. 2023. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wang, C.; Zhu, X.; Hong, J. C.; and Zheng, D. 2019. Artificial Intelligence in Radiotherapy Treatment Planning: Present and Future. *Technology in Cancer Research & Treatment*, 18: 1–11.

Wilk, S.; Michalowski, W.; Michalowski, M.; Farion, K.; Hing, M. M.; and Mohapatra, S. 2013. Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. *JBI*, 46(2): 341–353.

Wright, D. F.; Winter, H. R.; and Duffull, S. B. 2011. Understanding the time course of pharmacological effect: a PKPD approach. *BJCP*, 71(6): 815–823.