Personalized Pricing of Customized Car Features: A POMDP Benchmark with Consumer Simulation and Pricing Agents

Yingjie Lei

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Aberdeen.

Department of Computing Science

April 2026

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: April 2026

Abstract

This thesis studies personalized pricing for customized car features. The problem is difficult because buyers are heterogeneous, strategic, and only partially observable. Existing dynamic pricing work often assumes single-round quoting or simplified demand responses, which makes it difficult to study sequential pricing decisions with latent consumer traits and negotiation behaviour. For this reason, personalized pricing over customized car features is better viewed as a sequential partially observable decision problem.

To address this setting, the thesis formulates the task as a partially observable Markov decision process (POMDP) and develops a consumer-simulation-based benchmark. This benchmark provides a controlled setting in which heterogeneous buyers, multi-round negotiation, and different classes of pricing agents can be studied under one shared evaluation protocol.

The experiments compare these agents under the same benchmark protocol. The results show that the benchmark is non-trivial: profit and agreement do not coincide, different methods exhibit clearly different negotiation profiles, and learning-based agents do not automatically outperform stronger heuristic strategies. In particular, the findings suggest that representation quality and reward design materially affect learned-agent behaviour within the benchmark. Overall, the thesis contributes a structured and reproducible benchmark for studying personalized negotiation-based pricing under partial observability.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Binod Bhattarai and Dr. Faizan Khan, for their guidance, patience, and constructive feedback throughout this thesis. Their support has been invaluable to the completion of this work.

I am also grateful to the SCNU-UoA Joint Institute, where I spent four years of study and growth. I would like to thank my classmates and friends for their encouragement, discussions, and support during this period.

Finally, I would like to thank my family for their constant encouragement and support throughout my studies. Their advice and understanding have meant a great deal to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research background

Customized car feature pricing is a practical decision problem in which a seller adjusts the prices of optional vehicle features, such as colour, transmission type, or interior upgrades, to improve profit while maintaining consumer acceptance. Unlike pricing a fixed product, this setting requires the seller to reason about how consumers value different combinations of features, how much those features cost to implement, and how pricing decisions interact with purchase intent. In real customization scenarios, the same feature bundle may be perceived very differently by different buyers, making uniform pricing an insufficient approximation.

This problem is especially relevant when pricing decisions are personalized rather than purely catalogue-based. Some consumers care more about aesthetics, some about safety or technology, and others are constrained primarily by price or household needs. A useful pricing system therefore cannot rely only on a single estimate of market demand. It must instead reason about consumer heterogeneity at the level of individual or group-specific preferences, while still preserving a coherent pricing policy from the seller's perspective.

1.2 Motivation

These observations also help explain why the thesis is not framed only as an application study of car pricing. They point to a broader methodological gap concerning how personalized, negotiation-based pricing problems can be formalized and evaluated in a controlled way.

Existing work on dynamic pricing has provided valuable models for demand response, revenue management, and learning-based price adjustment [1–3]. However, much of that literature focuses on single-round offers, aggregate demand models, or relatively homogeneous consumer assumptions. These formulations are useful in many business settings, but they provide limited support for studying personalized feature-level pricing with multi-round negotiation and hidden consumer states in a unified and reproducible

way. In particular, prior work rarely standardizes this combination of partial observability, consumer heterogeneity, and bargaining interaction into a benchmark that can support controlled comparison across pricing agents, even though negotiation frameworks and evaluation environments already exist in adjacent lines of work [4, 5].

1.3 Problem statement

To address this gap, this thesis formulates customized car feature pricing as a partially observable Markov decision process (POMDP) and develops a simulator-based benchmark for controlled evaluation. The proposed framework combines a public-data-grounded persona generation process, a fixed persona bank, a multi-round negotiation environment, and a unified evaluation protocol for heuristic, model-free, and model-based pricing agents. At a practical level, the benchmark brings together explicit persona sampling, a bargaining process, and several comparable agent families under one task definition.

The main thesis claim is not that this work fully reconstructs a real vehicle market, nor that it proposes a single final pricing algorithm for deployment. Instead, the aim of the thesis is to provide a structured, reproducible, and extensible research setting for studying personalized negotiation-based pricing under partial observability. More specifically, its contribution lies in formalizing the task, instantiating it as a simulator-based benchmark, and using that benchmark to compare different classes of pricing agents. This also leaves room for later extensions, including richer semantic signals and adaptive inference mechanisms, without turning the thesis into a pure algorithm-novelty study.

1.4 Research challenges

The task also becomes more difficult when pricing is treated as an interactive process rather than a one-shot quotation. In many realistic buying settings, the seller does not observe the consumer's full willingness to pay at the start of the interaction. Observable information, such as age band, income band, household stage, vehicle ownership stage, or primary use case, can offer partial clues, but important decision variables remain hidden. These hidden factors include feature priorities, negotiation style, price sensitivity, patience, and walk-away behaviour. As a result, the pricing problem is better understood as sequential decision making under uncertainty than as static optimization over a fully known demand curve.

This sequential nature matters because the seller must update decisions based on limited feedback over multiple rounds. A quoted price may be accepted, rejected, countered, or followed by disengagement. Each response reveals only a small amount of information about the consumer's latent state, while delaying agreement can reduce the value of the interaction. Hence, the seller faces a combined inference-and-control problem: infer hidden consumer traits from behavioural signals and choose subsequent actions that improve long-term profit. This structure motivates a formal treatment beyond conventional

single-step pricing formulations.

1.5 Contributions

The main contributions of this thesis are as follows:

- We formulate personalized feature-level pricing for customized car options as a POMDP with hidden consumer traits, multi-round interaction, and profit-oriented sequential decision making, thereby turning an informal pricing scenario into a well-defined research task.
- We develop a consumer simulation framework that combines public-data-grounded persona generation, explicit separation between observable profiles and hidden decision traits, and a fixed persona bank for reproducible training, validation, and testing.
- We implement a multi-round negotiation environment for customized feature pricing, using a structured interaction protocol in which seller offers, consumer responses, counter-offers, and termination outcomes can be logged and evaluated consistently across episodes.
- We implement different types of pricing agents, including heuristic baselines and learning-based agents, for profit-oriented decision making in the proposed benchmark.
- We conduct comparative evaluation of these pricing agents under a shared benchmark protocol, and further examine the effects of reward design and selected extension modules through ablation studies.

1.6 Thesis organization

The remainder of this thesis is organized as follows.

Chapter 2 reviews related work on dynamic pricing, consumer simulation, negotiation environments, and learning-based sequential decision making.

Chapter 3 introduces the formal POMDP task definition.

Chapter 4 describes the persona generation pipeline, the negotiation environment, and the benchmark design choices.

Chapter 5 summarizes the pricing agents considered in this benchmark, together with selected extensions.

Chapter 6 presents the evaluation protocol, baseline methods, and experimental results.

Chapter 7 concludes the thesis by summarizing the main findings, offering a critical appraisal of the project, and outlining future work.

Chapter 2

Related work

This chapter reviews the main literature threads that are most relevant to the thesis. Rather than treating the problem only as a pricing task or only as a reinforcement-learning task, it considers related work on pricing and personalization, consumer heterogeneity and simulation-based modeling, negotiation under hidden information, and learning-based agent evaluation. The aim is to show that these components are individually well studied, while their combination into a reproducible benchmark for personalized feature-level pricing remains relatively limited.

2.1 Dynamic and personalized pricing

Feature pricing and product configuration pricing. Feature pricing and product configuration pricing have been studied in several adjacent forms. In the automotive domain, Thomassen [6] analyzes pricing across automobile engine variants and shows that markup patterns can vary systematically with product quality within a model line. This treats product variants as differentiated pricing objects rather than as a single homogeneous product. Related ideas also appear in configurable-product research, where the seller must jointly reason about component choices, assortments, and prices. For example, Wang et al. [7] study assortment planning and pricing for configurable products under a sequential choice process, showing that customer choice over required and optional components can materially affect optimal pricing decisions. Taken together, these studies support the view that option-level and attribute-level pricing is a meaningful problem.

Sequential dynamic pricing under uncertain demand. Dynamic pricing has also been extensively studied as a sequential decision problem under uncertain demand. Common themes in this literature include contextual price adjustment, strategic buyer behaviour, and learning-based control. Choi et al. [1], for example, develop a semi-parametric contextual pricing algorithm in which the seller adapts prices under censored valuation information. Liu et al. [8] further study contextual dynamic pricing with strategic buyers, showing that buyer-side strategic behaviour can materially complicate the pricing problem. Deep reinforcement learning has also been applied to adjacent pricing settings.

Pandey et al. [2] formulate dynamic toll pricing as a POMDP and use deep reinforcement learning under partial observability. Liu et al. [3] similarly apply deep reinforcement learning to dynamic pricing in e-commerce, while Villarrubia-Martin et al. [9] propose a multi-agent reinforcement-learning framework for high-speed rail pricing. Taken together, these studies show that dynamic pricing is already a rich sequential literature, and that reinforcement learning is a credible methodological choice when the seller must adapt prices over time under uncertainty.

Personalized pricing, welfare, and fairness. A smaller but still relevant thread considers how personalized pricing is interpreted once sellers begin to exploit consumer differences. Biggs et al. [10] emphasize interpretability in personalized pricing, while Dubé and Misra [11] and Priester et al. [12] examine welfare and fairness concerns from the consumer side. These issues are not the main focus of the present thesis, but they show that personalized pricing is discussed in the literature not only as an optimization problem, but also as a practice with broader consumer-facing implications.

Taken together, the pricing literature already establishes that product attributes can matter for pricing and that dynamic pricing is inherently sequential under uncertain demand. However, these strands are often studied separately. Existing work usually focuses on configurable products without multi-round bargaining, on sequential pricing without a fixed customized bundle, or on personalized pricing without a reproducible benchmarked negotiation setting. The present thesis builds on these strands by bringing feature-level pricing, hidden consumer heterogeneity, and sequential negotiation into one structured evaluation framework.

2.2 Consumer heterogeneity and simulation-based modeling

Consumer heterogeneity and willingness to pay have long been central concerns in economics and marketing. A recurring message in this literature is that buyers do not respond to prices or product attributes in the same way, and that pricing models become more informative once this variation is represented explicitly. In the automotive domain, for instance, Shin et al. [13] show that consumers differ substantially in their preferences for vehicle technology options and fuel types. This line of work makes clear why pricing over customized features should be grounded in heterogeneous buyer responses rather than in a single representative demand curve.

Related work has also studied how such heterogeneity can be represented in computational systems. Schlechtinger et al. [14] build a market-simulation framework that supports different demand models and market conditions, and use it to analyze the behaviour of learning-based pricing agents. Xia et al. [15] similarly propose a multi-stage model for generating synthetic retail transactions with heterogeneous customer behaviour, including

price sensitivity and experience-dependent effects. These studies are useful here because they show that consumer-side variation can be embedded into simulation environments in a structured and research-oriented way rather than treated as unmodeled noise.

Another closely related direction concerns explicit simulated populations and persona-style agents. Salem et al. [16], for example, present TinyTroupe as an LLM-powered toolkit for defining and simulating detailed personas with persistent attributes and behavioural tendencies. Although its goals are broader than pricing, it helps justify the use of structured persona records as simulation assets. At the level of benchmark infrastructure, Liu et al. [17] show in an adjacent domain that reusable environments, fixed evaluation protocols, and shared assets can themselves be meaningful research contributions. Together, these strands support the simulation choices made in this thesis. At the same time, relatively few works organize heterogeneous consumers as a fixed persona bank with explicit observable and hidden traits under one shared pricing protocol.

2.3 Negotiation and partially observable sequential decision making

Automated negotiation has a long research history, and several frameworks already support the design and comparison of bargaining agents. A representative example is Genius [4], which was developed as a general environment for automated negotiators across different domains. Competition-style evaluation has also become standard in this literature. The ANAC competitions, for example, were explicitly designed to encourage the development and comparison of automated negotiation agents under shared protocols [18]. More recent platforms such as NegMAS [5] and NEGOTIATOR [19] further broaden the available tooling for negotiation research and human–agent interaction.

Taken together, these frameworks show that multi-round bargaining is already a recognized research setting rather than a niche implementation choice. They model negotiation as an interactive process in which offers, counter-offers, acceptance, and breakdown unfold over time rather than being compressed into a single posted-price decision. More recent work, including LLM-oriented negotiation benchmarks [20, 21], shows that negotiation evaluation remains active, although such studies are not primarily concerned with pricing benchmarks.

This literature is also closely connected to partially observable sequential decision making. In many negotiation settings, the agent must act without direct access to the other party's full preferences or intentions, which makes hidden-state reasoning central. More generally, POMDPs remain a standard framework for modeling sequential decision problems under incomplete information [22]. Even in adjacent pricing applications, POMDP formulations have been used when the seller only receives partial behavioural signals rather than direct access to buyer valuations [2]. Together, these strands help justify why

the present thesis treats pricing over customized bundles as a negotiation problem under partial observability, rather than as a static supervised prediction task.

2.4 Reinforcement learning methods

Model-free reinforcement learning. Model-free reinforcement learning has been widely used in sequential decision problems because it learns a policy directly from interaction rather than from an explicit model of environment dynamics. Among these methods, Proximal Policy Optimization (PPO) [23] has become one of the most commonly used baselines because of its relative simplicity and robustness across domains. In pricing-related applications, reinforcement learning has already been explored in settings such as consumer credit, toll pricing, and e-commerce [2, 3, 24]. These works differ in domain and task structure, but together they show that model-free reinforcement learning is a plausible baseline family whenever pricing decisions must adapt over time under uncertainty.

World models and latent-state methods. A complementary direction is model-based or latent-state reinforcement learning, in which the agent learns an internal representation of environment dynamics and uses it to support control. Early work on world models [25] showed how compact latent dynamics can be learned from experience and then used for downstream decision making. More recently, DreamerV3 [26] has demonstrated that latent world-model learning can provide a strong and general-purpose control framework across many domains. This line of work is especially relevant when the environment is partially observable, because the agent must infer useful latent structure from incomplete observations rather than rely on full state access at each step. It therefore provides a clear methodological contrast to PPO rather than merely a second learning baseline of the same kind.

Together, these reinforcement-learning methods provide the main methodological context for the learning-based pricing agents considered in this thesis. The point of including them here is not to claim a new RL algorithm, but to show that both model-free and model-based agent families are well-motivated reference methods for a sequential pricing benchmark under hidden information.

Chapter Synthesis. Taken together, the literature reviewed in this chapter establishes four ingredients that shape the thesis: pricing over configurable products, buyer-side heterogeneity and simulation, negotiation under hidden information, and learning-based agent comparison. Each strand already exists on its own, but they remain only loosely connected in prior work. The next chapter therefore turns from literature context to formal task definition by specifying how these ingredients are combined into one POMDP benchmark for personalized negotiation-based pricing.

Chapter 3

Problem formulation

This chapter defines the pricing task at the problem level. Its purpose is to state what one episode represents, why the setting is partially observable and sequential, and how the task can be formalized as a POMDP. The focus is on the decision structure of the benchmark: the hidden consumer state, the information available to the seller, the action semantics, the reward definition, and the finite negotiation horizon. This chapter remains at the level of task formulation rather than simulator implementation. Detailed persona construction, willingness-to-pay modeling, negotiation mechanics, and benchmark instantiation are deferred to Chapter 4.

3.1 Task setting and episode definition

This section defines the benchmark task before introducing the formal POMDP tuple. The focus here is on what one episode represents, who interacts in the environment, and what information and objectives characterize the task at a high level.

3.1.1 Episode definition

One episode is defined as one negotiation between the pricing agent, acting as the seller, and one sampled consumer, acting as the buyer, over one selected bundle of customized car features. The configuration remains fixed within the episode, while the quoted price can change across rounds as the interaction unfolds. Hence, the benchmark does not model product redesign during negotiation; it models dynamic pricing over a fixed customized package.

At the episode level, the interaction starts from an initial observation that contains the selected bundle and the observable part of the consumer profile. The agent then takes negotiation actions, and the consumer responds according to its internal state. The episode ends when an agreement is reached, one side leaves the negotiation, or the interaction horizon is exhausted. In task terms, each episode is a single bargaining trajectory defined over one buyer–configuration pair.

Figure 3.1 summarizes this structure at a high level. One sampled consumer and one fixed configuration bundle are paired at initialization, the seller then acts through a

bounded multi-round bargaining loop under partial observability, and the episode terminates either with realized profit from agreement or with zero profit when no agreement is reached.

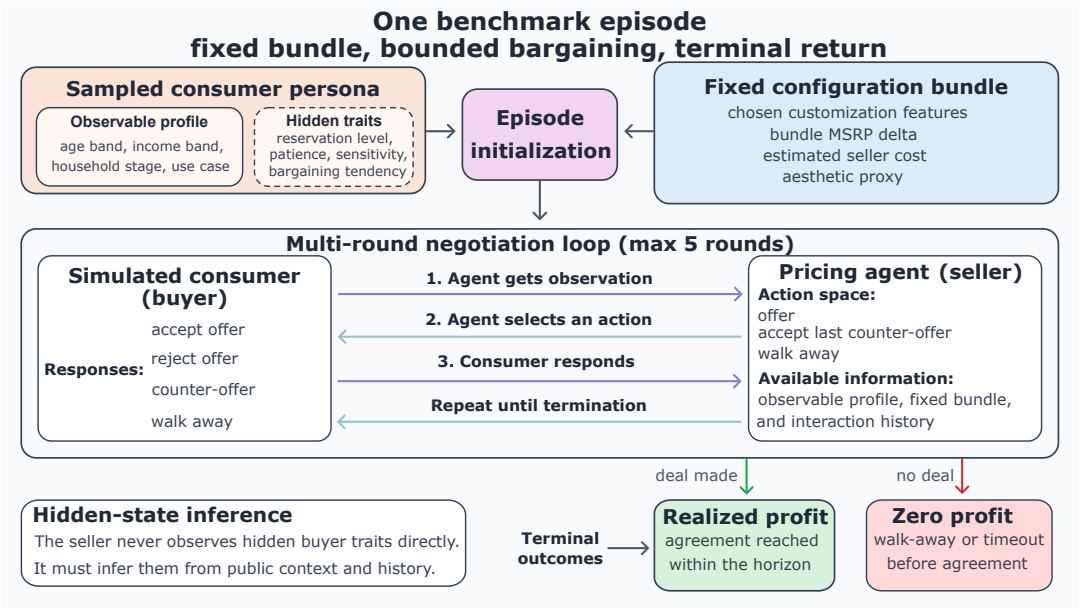

Figure 3.1: High-level structure of one benchmark episode. Each episode begins with one sampled consumer and one fixed configuration bundle, proceeds through a bounded bargaining loop of up to five rounds, and ends either with realized profit from agreement or with zero profit when no agreement is reached.

3.1.2 Seller objective

The goal of the pricing agent is to maximize expected profit over the course of the interaction. This objective depends not only on the quoted transaction price, but also on the cost of delivering the selected features and on whether the negotiation successfully reaches agreement. A pricing policy that asks for a very high price may improve margin in successful cases, but it may also increase the chance of rejection or walk-away.

As a result, the seller faces a trade-off between deal profit and negotiation failure, with the cost of prolonged bargaining arising mainly through delay-induced failure risk and the finite interaction budget rather than through many stepwise heuristics. This makes the task different from static preference estimation. The agent is required to choose actions that improve long-run return rather than merely predict consumer preferences.

3.1.3 Observable and hidden consumer information

Consumer heterogeneity is an essential assumption of the benchmark. Different consumers may respond differently to the same feature bundle and the same price trajectory because they differ in budget tolerance, feature priorities, negotiation style, and willingness to continue bargaining. As a result, the same offer may lead to acceptance in one episode, a counter-offer in another, and immediate disengagement in a third.

To reflect this structure, the benchmark distinguishes between observable consumer descriptors and hidden behavioural traits. Observable information includes coarse profile fields such as age band, income band, household stage, ownership stage, and primary use case. Hidden information includes factors such as reservation-price level, price sensitivity, patience, counter-offer tendency, walk-away tendency, and latent feature preferences. The detailed generation of these traits is deferred to Chapter 4; here they are introduced only as task-level components.

3.1.4 Interaction protocol

The interaction protocol is multi-round. In each round, the seller may issue a new offer, accept the latest consumer counter-offer, or walk away. The consumer may then accept, reject, counter, or disengage depending on its hidden state and the current interaction context. This repeated exchange forms the basic interaction structure of the benchmark and is more suitable for this task than a single-shot quotation.

Importantly, the history of the interaction carries information. Previous offers, responses, and counters affect what the agent knows about the consumer and what actions remain reasonable in later rounds. This motivates a history-dependent decision process rather than a single-step pricing problem.

3.2 Why the task is a POMDP

This section explains why the task defined above is more appropriately modeled as a POMDP than as a fully observable or single-step decision problem. The key reason is that the agent must act over multiple rounds while observing only a partial view of the consumer state.

3.2.1 Partial observability

The pricing agent never observes the consumer's full internal decision state. What the agent sees is limited to the observable profile, the selected configuration, and the negotiation history revealed so far. Important decision variables, such as latent feature preferences, reservation-price drivers, price sensitivity, patience, and walk-away tendency, remain hidden throughout the interaction.

This matters because observable behaviour does not uniquely identify the underlying consumer state. The same observed rejection or counter-offer may arise from different internal reasons, and those reasons may lead to different future responses. The agent must therefore infer hidden information indirectly from behaviour rather than rely on direct state access.

3.2.2 Sequential decision making

The task is sequential because current actions affect both future observations and future returns. A high offer may produce a profitable immediate deal if accepted, but it may also

trigger rejection, increase walk-away risk, or lead to a tougher counter later. Likewise, an early concession may preserve the interaction while reducing short-term margin.

This means that decisions cannot be evaluated independently round by round. The agent must reason over the trajectory of the negotiation, since each action changes the information available at the next step and influences the set of future outcomes that remain possible.

3.2.3 Control under uncertainty

The problem is not only to infer consumer preferences, but to choose actions that improve expected return under uncertainty. Even if the agent can partially infer that a consumer is price sensitive or impatient, the main challenge is still to decide what to do next: continue negotiating, lower the offer, accept a counter, or terminate the interaction.

For this reason, the benchmark is more naturally understood as a control problem under uncertainty than as a pure estimation problem. The POMDP formulation captures this combination of hidden state, sequential interaction, and profit-oriented decision making in a unified way.

3.3 POMDP definition

We define the benchmark task as a finite-horizon POMDP

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, R, \rho_0, H, \gamma),$$

where $\mathcal{S}$ is the hidden state space, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, T is the transition function, Ω is the observation function, R is the reward function, ρ_0 is the initial-state distribution, H is the finite horizon, and γ is the discount factor. In our benchmark, the horizon is $H = 5$. The task is therefore best interpreted as finite-horizon return maximization under partial observability.

Figure 3.2 summarizes the same formulation visually by separating the environment-side latent state from the seller-visible observation and showing how action, reward, transition, and the finite horizon connect across this information boundary.

3.3.1 Hidden state space $\mathcal{S}$

The hidden state represents the full information required to determine the future course of the negotiation. At a structural level, it can be written as

$$s_t = (s_t^{\text{consumer}}, s_t^{\text{config}}, s_t^{\text{neg}}),$$

where s_t^{consumer} is the consumer state, s_t^{config} is the selected configuration state, and s_t^{neg} is the current negotiation status.

The consumer component s_t^{consumer} contains both observable profile descriptors and

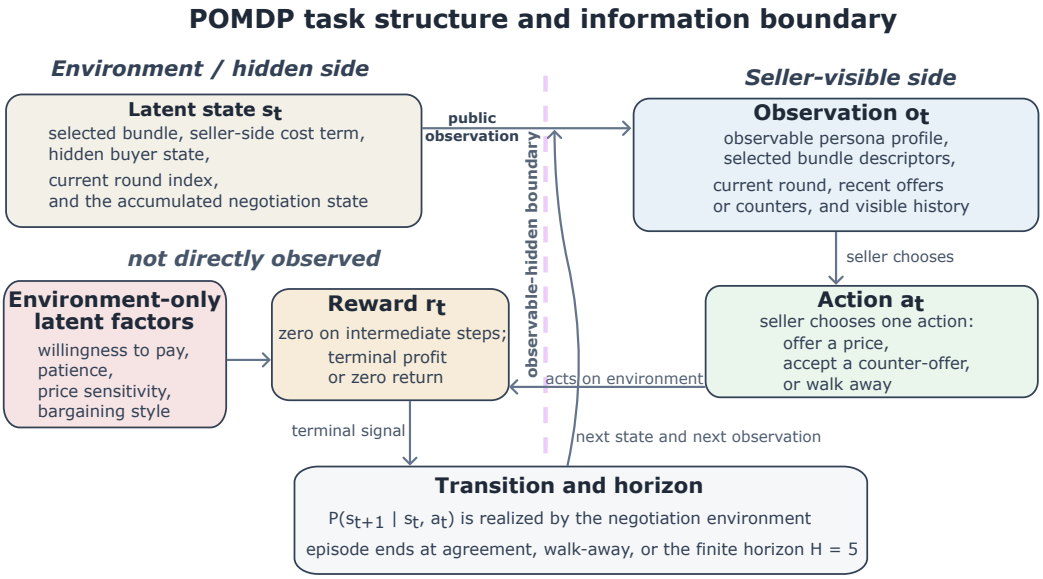

Figure 3.2: POMDP task structure and information boundary: the environment maintains the latent negotiation state, while the seller receives only a partial observation and acts through the restricted action interface. Reward is realized through episode outcome, and transition proceeds under the finite horizon $H = 5$.

latent behavioural variables. The observable part includes coarse persona descriptors, while the hidden part contains the valuation-related and bargaining-related factors that shape behaviour. Together, these hidden variables determine how the consumer values the current bundle and how it reacts to seller actions.

The configuration component s_t^{config} contains the selected bundle of customized car features under negotiation together with the bundle-level descriptors that influence both buyer-side valuation and seller-side outcomes. These include economic descriptors of the bundle and an appearance-related signal, although not all of this information is directly exposed to the agent.

The negotiation component s_t^{neg} records the current stage of the interaction. At a minimum, it includes the *round index*, the latest seller offer, the latest consumer response, the latest consumer counter-offer when available, and whether the episode has already terminated. Although parts of this information are reflected in the observation, the full negotiation state is not directly revealed to the agent.

Finally, the hidden state also contains a round-dependent latent valuation term, denoted here as the consumer's current willingness to pay, WTP_t. This quantity evolves with hidden persona variables, bundle-level signals, round progression, and stochastic variation. It is not directly observable to the agent, but it strongly influences whether a given offer is accepted, rejected, or countered. Its detailed construction is deferred to Chapter 4.

3.3.2 Action space $\mathcal{A}$

The seller action space is defined over three move types:

$$\mathcal{A} = \{\texttt{offer}, \texttt{accept}, \texttt{walkaway}\} \times \mathcal{P},$$

where the price component in $\mathcal{P}$ is relevant only when the move is `offer`. The three moves have straightforward task-level semantics. The `offer` action proposes a new seller price for the current configuration. The `accept` action commits to the latest consumer counter-offer when such a counter is available. The `walkaway` action ends the negotiation immediately from the seller side. The concrete agent-facing realizations of these decisions are described later, but the task itself is defined only by these three move types together with the offered price on the `offer` branch.

3.3.3 Observation space $\mathcal{O}$

The observation space contains only seller-visible information. At round t, the agent receives a partial observation

$$o_t = \Omega(s_t),$$

which includes the current negotiation status, selected bundle descriptors, and the observable part of the consumer profile, but excludes the latent behavioural variables that actually generate consumer responses.

In practical terms, this means that the seller observes coarse profile information, bundle-level descriptors, and the interaction history accumulated so far, including recent offers and responses. These signals are sufficient to support sequential decision making, but they do not expose the hidden variables that actually govern buyer valuation and behaviour.

The key point is the information boundary itself: the seller receives only seller-visible information and must act without direct access to the buyer's full internal state.

3.3.4 Transition function T

The transition function specifies how the hidden state evolves after each seller action. At a high level, the next state depends on the current consumer state, the fixed configuration, the current negotiation status, and the chosen action. The resulting transition may produce an agreement, a rejection, a counter-offer, a buyer walk-away event, or a timeout-like progression.

Because consumer response depends on latent valuation and bargaining traits, the same observed outcome need not correspond to the same underlying state. The transition therefore changes both what remains possible later in the episode and what can be inferred about the buyer. It also advances the negotiation toward termination under a finite interaction budget, so delaying agreement can affect both future observations and future

return.

3.3.5 Observation function Ω

The observation function maps the hidden state into the limited feedback available to the seller. In this benchmark, Ω does not reveal the consumer's internal willingness-to-pay decomposition or latent behavioural coefficients directly. Instead, it exposes only what the seller could plausibly observe from the interaction: coarse profile descriptors, bundle-level descriptors, and the latest interaction outcome.

This design is important for preserving the POMDP structure. The seller may observe the current stage of the negotiation and the most recent interaction outcome, but not the hidden reasons that generated that outcome. The observation function therefore acts as a controlled information bottleneck between the full simulator state and the agent.

3.3.6 Reward function R

At the task-definition level, the reward function is profit-oriented and sparse. The seller receives positive return only when the negotiation terminates in a deal, with magnitude determined by realized profit. Unsuccessful episodes yield zero terminal return. The POMDP reward can therefore be written as

$$R_t = \begin{cases} \mathit{profit}_t, & \text{if the episode terminates with a deal,} \\ 0, & \text{otherwise,} \end{cases}$$

where profit_t is the realized seller profit at deal completion,

$$\mathit{profit}_t = p_{\text{deal}} - c_{\text{impl}},$$

where p_{deal} is the final agreed price and c_{impl} is the seller-side cost term associated with the selected configuration. This formulation captures the central task objective: use a limited interaction budget to convert buyer–seller negotiation into profitable agreements. At the benchmark-interface level, the environment may still emit auxiliary event-level signals for particular interaction outcomes, but those do not change the underlying task objective defined here. Later chapters describe how this objective is realized under the concrete benchmark protocol and learning interfaces.

3.3.7 Initial-state distribution ρ_0 and discount factor γ

At the task-definition level, the initial-state distribution ρ_0 is a controlled distribution over buyer–configuration pairs. Each episode begins with one consumer and one fixed configuration bundle before any interaction history has been revealed. Conceptually, the role of ρ_0 is to define how negotiation episodes are initialized.

The discount factor γ is included in the POMDP tuple for completeness. At the

task-definition level, the benchmark is formulated as *undiscounted episodic profit maximization*, so the problem-level interpretation is $\gamma = 1$. Different learning algorithms may use their own optimization settings during training, but those are implementation choices rather than part of the task definition itself. The practical decision pressure in the benchmark comes primarily from the finite negotiation horizon and the risk of reaching a no-deal termination.

3.3.8 Termination and finite horizon

Episodes terminate under several clearly defined conditions. Agreement is terminal when the consumer accepts the seller's offer or when the seller accepts the latest consumer counter-offer. Seller walk-away is also terminal. In addition, the episode may end because the consumer leaves the negotiation or because the interaction horizon is exhausted.

The benchmark is explicitly finite horizon. Delaying too long can itself turn a potentially profitable interaction into failure. Because consumers may differ in how they tolerate continued bargaining, time pressure interacts naturally with hidden buyer state. The finite-horizon structure is therefore a core part of the POMDP formulation.

Chapter 4

Consumer simulator and benchmark design

This chapter presents the consumer simulator and benchmark design. It describes how vehicle inputs, public-data-grounded consumer assumptions, persona construction, latent value modeling, and multi-round negotiation are assembled into a benchmark environment for pricing-agent evaluation. The chapter also explains the benchmark choices that support controlled comparison, including a fixed persona population, reproducible episode generation, and shared evaluation conditions across methods. Throughout the chapter, the emphasis is on a structured and transparent simulator design.

4.1 Customization scope

This section defines the input scope of the benchmark. It first introduces the fixed vehicle customization space over which pricing decisions are made, and then explains how public consumer priors and explicit abstraction choices are used to ground the simulator. The focus is on a controlled and transparent benchmark setup with explicit scope boundaries

4.1.1 Vehicle customization catalog

The configuration space of the benchmark is built from the customization options of a real vehicle model. In our benchmark, we use the *Mercedes-Benz E350 Sedan* [27] as the anchor vehicle and organize its customization choices into a fixed catalog of feature options. The option-level MSRP deltas are compiled from the official Mercedes-Benz pricing materials for this vehicle configuration and then standardized into a benchmark catalog. Each option is associated with an MSRP delta, which is later used both in buyer-side value modeling and in the seller-side cost proxy.

The catalog contains *20 canonical options across 11 customization dimensions*, including paint color, wheels, exterior styling, upholstery, trim, comfort features, audio system, technology package, safety features, performance upgrades, and lighting options. In each episode, the benchmark samples *one canonical option per dimension* to form the configuration bundle under negotiation. In our benchmark, the catalog is intentionally

Table 4.1: Overview of the E350 customization catalog used in the benchmark. MSRP deltas are compiled into the benchmark catalog from official vehicle pricing materials.

Dimension	**Representative options**	**MSRP delta range**
Paint color	Standard paint, metallic paint, MANUFAKTUR paint	\$0–\$1,750
Wheels	18-inch standard, 19-inch upgrade, AMG high-end wheels	\$0–\$1,950
Exterior styling	Styling upgrade package	\$400
Upholstery	MB-Tex, leather, nappa leather	\$0–\$2,990
Trim	Standard trim, premium trim	\$0–\$150
Comfort	Seat comfort upgrade, soft-close doors, multi-contour package	\$500–\$2,950
Audio	Burmester 4D audio	\$1,030
Technology	MBUX Superscreen	\$1,500
Safety	Driver Assistance Package	\$1,950
Performance	AIRMATIC package	\$3,200
Lighting	Digital Light	\$990

kept free of richer package-dependency logic or cross-option compatibility constraints, so that the configuration space remains transparent and comparable across methods. Table 4.1 provides an overview of the representative options in each dimension along with their associated MSRP delta ranges. A complete option-level reference, including the full list of benchmark option identifiers, MSRP deltas, and aesthetic prior weights, is provided in Appendix A. This catalog serves as the fixed product space for the benchmark, ensuring that all compared agents face the same configuration choices and allowing the analysis to focus on pricing and negotiation behaviour rather than on cross-product variation.

The same catalog also defines the benchmark's economic quantities. The customer-facing price anchor of a bundle is the sum of option-level MSRP deltas, denoted as the total MSRP delta of the selected configuration. On the seller side, implementation cost is modeled through a fixed-ratio proxy:

$$c_{\text{impl}}(b) = 0.5 \sum_{i \in b} \Delta\text{MSRP}_i,$$

where b is the selected configuration bundle and ΔMSRP_i is the MSRP delta of option i. This assumption treats option cost as a benchmark-level approximation, allowing profit to be defined transparently and reproducibly.

In addition to price information, each catalog option is assigned a scalar appearance-oriented prior weight, which provides the hand-crafted signal that drives appearance-related value in the base simulator. The optional semantic extension then adds offline CLIP text features on the observation side for agent learning, including a semantic vector and a projected scalar score, while leaving both the underlying catalog space and the base

simulator's aesthetic signal unchanged.

Using one fixed vehicle catalog serves an important benchmark purpose. It keeps the product space stable across compared agents, ensures that all methods face the same configuration choices, and keeps the analysis focused on pricing and negotiation behaviour. Within this benchmark, the E350 catalog is used as a realistic and controlled substrate.

4.1.2 Consumer priors and abstraction choices

The **observable** part of the consumer population is shaped by public statistics from the United States. In our benchmark, *age band* and *household stage* are anchored in U.S. census-style population summaries [28], *income band* is further calibrated against recent U.S. income reporting [29], and *primary use case* is mapped from household travel-purpose statistics [30]. By contrast, *ownership stage* is treated as an explicit benchmark-level modeling assumption rather than as a field read directly from one public table.

This design gives the synthetic population a transparent empirical anchor while avoiding private or individual-level consumer records that would be harder to reproduce. The simulator is designed for *population-level realism*

Not all variables required by the simulator can be obtained directly from public data. In particular, latent preference factors such as appearance-oriented taste, feature-level preference weighting, and some behavioural tendencies must be represented through simulator-defined proxies or latent variables. For example, the simulator includes an *aesthetic proxy* to capture appearance-related preference variation. This quantity serves as a structured proxy for hidden preference.

These abstraction choices are deliberate. In this benchmark, proxy variables provide a transparent and reproducible way to connect public population-level inputs with the latent preference structure required by a POMDP-based negotiation environment.

4.2 Consumer simulator

Upon fixing the customization scope, the benchmark instantiates a consumer simulator to represent heterogeneous buyers in a structured way. The simulator is built around a layered persona schema, which separates agent-visible profile descriptors from environment-only behavioural parameters. On top of this schema, the benchmark defines configuration-level value signals and a round-level willingness-to-pay process that together drive negotiation behaviour during each episode.

4.2.1 Persona schema

The consumer simulator represents each buyer through a structured persona record rather than through a free-form profile description. Each persona is defined by two complementary layers: *observable fields*, which are visible to the pricing agent, and *hidden behavioural fields*, which are used internally by the environment. This separation is not only

a modeling convenience, but also a core requirement of the benchmark's POMDP formulation. Table 4.2 summarizes the main fields in each layer along with their generation basis and visibility status.

The **observable layer** provides the coarse profile information that the agent is allowed to condition on during negotiation. In our persona schema, these fields are *age band* (*18–25, 26–35, 36–50, 50+*), *income band* (*<60k, 60–100k, 100–180k, 180k+*), *household stage* (*single, couple, family*), *ownership stage* (*first-time, replacement, additional*), and *primary use case* (*commute, family, luxury, performance, mixed*). The underlying priors are grounded in public U.S. demographic and travel data, with age and household composition anchored in census-style summaries [28], income calibration tied to recent U.S. income reporting [29], and use-case proportions mapped from household travel-purpose statistics [30]. The ownership-stage field is kept visible because it is behaviourally important for negotiation, but its distribution is treated as a benchmark-level modeling assumption rather than as a directly observed public-statistics variable. Importantly, these fields are not sampled independently. Instead, the simulator uses lightweight conditional tables, for example sampling income, household stage, and ownership stage conditioned on age band, and sampling primary use case conditioned on household stage. This conditional structure keeps the overall persona distribution closer to plausible market patterns and avoids obviously inconsistent combinations that would arise from naive independent sampling. The full observable priors and conditional tables are reported in Appendix B.

The **hidden layer** captures the latent factors that govern how a consumer interprets the same bundle and the same price trajectory. In our simulator, these variables include categorical traits such as *decision style* (*analytic, balanced, expressive*), *tech-affinity band* (*low, medium, high*), and the consumer's *top two stated priorities* drawn from *price, safety, comfort, performance, aesthetics,* and *technology*. It also includes numeric parameters such as *reservation-price base, price sensitivity, aesthetic sensitivity, patience, counter strength, walk-away threshold, belief obscurity, brand loyalty,* and *impulsivity*, together with a normalized latent *feature-preference weight vector* over *safety, comfort, performance, technology,* and *aesthetics*. These hidden terms are not directly observed by the agent. Instead, they are generated from the observable profile through explicit conditional mappings, numeric mixtures, and coupling rules. For example, decision style and top priorities are conditioned on primary use case, tech affinity is conditioned on age band, reservation-price base is conditioned on income band, and several numeric traits are further adjusted by ownership stage, use case, and latent preference interactions. These hidden traits are modeled as a structured latent layer that is behaviorally coherent and reproducible. The hidden conditional tables and linked numeric rules are also summarized in Appendix B.

This layered schema preserves partial observability while keeping the joint consumer

Table 4.2: Summary of the persona schema used in the consumer simulator. Observable fields are grounded in public U.S. buyer priors, while hidden fields are derived through conditional mapping and coupling rules.

Layer	Fields	Generation basis	Visible
Observable profile	Age band, income band, household stage, ownership stage, primary use case	Public-data-grounded U.S. buyer priors with lightweight conditional tables to preserve plausible cross-field dependencies	Yes
Hidden categories	Decision style, tech-affinity band, top two stated priorities	Conditional mappings from observable profile, especially age band and primary use case	No
Hidden monetary and sensitivity terms	Reservation-price base, price sensitivity, aesthetic sensitivity	Income-conditioned sampling plus numeric mixtures and bounded adjustments	No
Hidden bargaining traits	Patience, counter strength, walk-away threshold, belief obscurity, impulsivity	Numeric mixtures with additional conditional shifts and coupling rules	No
Hidden preference structure	Brand loyalty and feature-preference weight vector over safety, comfort, performance, technology, and aesthetics	Template-based initialization with conditional refinement from latent profile structure	No

distribution plausible enough for benchmark use. Observable fields provide the agent with a coarse but interpretable view of the consumer, while hidden fields drive the bargaining and valuation dynamics that must be inferred from interaction history. It also separates stable persona attributes from session-level variables such as round-specific willingness to pay or negotiation fatigue.

4.2.2 Persona generation and offline persona bank

In our benchmark, persona generation follows a two-stage pipeline. Figure 4.1 summarizes this pipeline and the resulting offline-bank workflow. First, the simulator samples the *observable profile* from the public-data-grounded U.S. buyer priors described above. This stage uses the conditional sampling order encoded in the observable distribution file: age band is sampled first, after which income band, household stage, and ownership stage are sampled conditionally on age band, and primary use case is sampled conditionally on household stage. As a result, the generated personas follow a structured joint distribution rather than a collection of independently drawn fields.

Second, the simulator generates the *hidden profile* conditioned on the sampled observable fields. Categorical hidden traits are drawn from explicit conditional mappings, for example decision style and top-priority pairs given primary use case, and tech-affinity band given age band. Numeric traits are then sampled from predefined mixtures and refined through additional conditional shifts and coupling rules. In practice, this means that income band influences reservation-price base, ownership stage can shift price sensitivity,

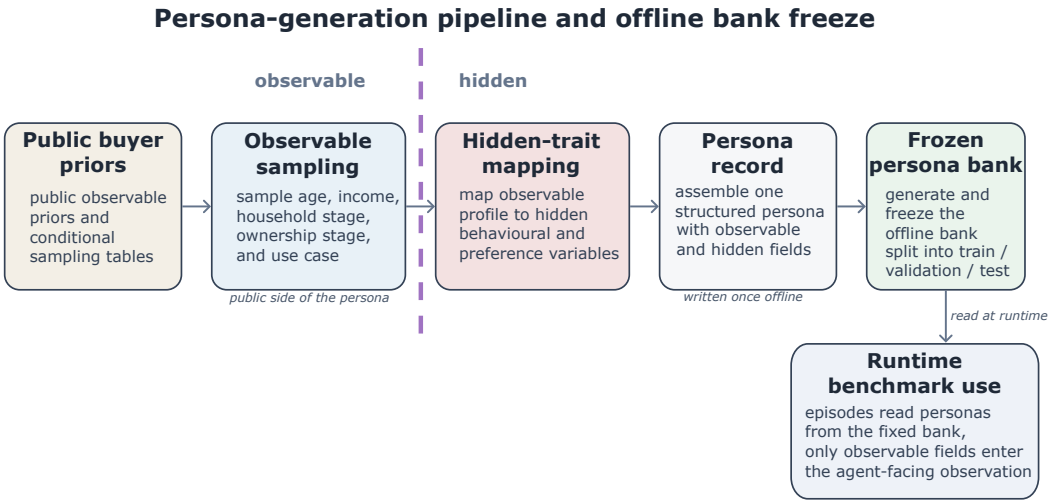

Figure 4.1: Persona-generation pipeline from public observable priors to the frozen offline persona bank. Observable fields are sampled first, hidden behavioural variables are then mapped from them, and the resulting structured persona records are written once and reused across benchmark splits at runtime.

patience, and walk-away tendency, and latent interactions further tie together quantities such as price sensitivity, reservation value, counter strength, and walk-away threshold. The feature-preference weight vector is built from use-case templates and then adjusted by decision style and stated priorities before being normalized. This sequence produces personas with behaviorally linked hidden parameters.

The resulting persona record is written in a structured schema with a stable identifier, split tag, provenance tag, and explicit observable and hidden fields. A completed negotiation can therefore be linked back to a concrete persona instance for later analysis by profile segment, hidden-trait regime, or split membership.

For the benchmark itself, these personas are materialized into a fixed *offline persona bank* before any training or evaluation begins. In our benchmark, the main bank contains *50,000 personas* generated with a fixed seed and partitioned into train, validation, and test subsets using a *70/15/15* split. The environment samples only from the selected split, while each episode records the corresponding *persona ID*, *persona source*, and *persona split*.

Freezing the persona bank decouples persona generation from downstream agent training, keeps the evaluation population fixed across methods, and improves reproducibility under stable split boundaries and persona identities. It also avoids repeated large-scale online persona generation during training and evaluation, which keeps the benchmark workflow lighter and more predictable.

Figure 4.2 gives a compact overview of the observable composition of the frozen persona bank. The distribution is visibly structured, with greater concentration in mid-to-higher income bands and replacement buyers, while the remaining observable fields also

retain broad coverage across the benchmark.

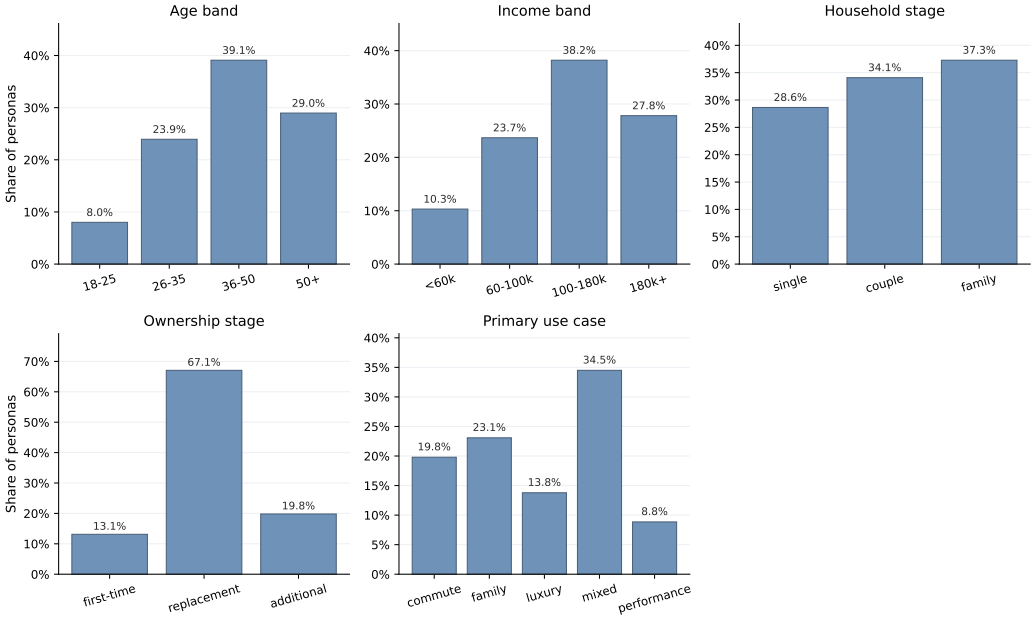

Figure 4.2: Observable profile distributions in the frozen persona bank.

4.2.3 Configuration-level value signals

This subsection explains how a fixed customization bundle becomes inputs to the buyer-side value model. The selected options are first converted into deterministic *bundle-level value signals*. Some of these signals are visible to the pricing agent as bundle descriptors, while others remain inside the simulator as structural summaries or persona-conditioned quantities. The bundle itself stays fixed during an episode; what changes later is how the current buyer values that bundle as the negotiation unfolds.

Agent-visible bundle descriptors. The first layer describes the selected bundle in terms the pricing agent can use directly. For any bundle b, the simulator computes the *total MSRP delta*,

$$\Delta_{\mathrm{MSRP}}(b) = \sum_{i \in b} \Delta\mathrm{MSRP}_i,$$

which serves as the consumer-facing price anchor of the selected customization set. The raw observation also includes the selected option keys, the benchmark-level implementation-cost proxy introduced earlier in this chapter, and the base aesthetic proxy derived from catalog-level appearance weights. These agent-visible descriptors can be summarized as

$$d_{\mathrm{obs}}(b) = \big(\textit{selected option keys},\ \Delta_{\mathrm{MSRP}}(b),\ c_{\mathrm{impl}}(b),\ a_{\mathrm{proxy}}(b)\big),$$

where $c_{\mathrm{impl}}(b)$ is the seller-side implementation-cost proxy and $a_{\mathrm{proxy}}(b)$ is the hand-crafted appearance-oriented proxy score of the bundle. These descriptors tell the agent

what was selected and summarize the bundle's price and appearance signals, but they do not reveal the buyer's hidden feature preferences.

Simulator-internal bundle summaries. The second layer summarizes the selected options as an internal functional profile. Instead of treating the bundle only as a list of option keys, the simulator maps each selected option into one of five feature channels: *safety, comfort, performance, technology,* and *aesthetics*. Each selected option is first assigned an internal weight,

$$w_i = \max(300,\ \Delta\text{MSRP}_i),$$

so that zero-cost baseline options still contribute to the bundle representation. The simulator then computes the share of the weighted bundle that belongs to each feature channel k:

$$g_k(b) = \frac{\sum_{i \in b:\ c(i)=k} w_i}{\sum_{j \in b} w_j},$$

where $c(i)$ denotes the feature category associated with option i. The base simulator also computes an *aesthetic proxy* by averaging the catalog-level appearance weights of the selected options. The internal bundle-summary vector is then formed by concatenating this aesthetic proxy with the five normalized channel signals:

$$g(b) = \big(a_{\text{proxy}}(b),\ g_{\text{safety}}(b),\ g_{\text{comfort}}(b),\ g_{\text{performance}}(b),\ g_{\text{tech}}(b),\ g_{\text{aesthetics}}(b)\big).$$

This vector is a simulator-internal summary of the bundle's feature composition. It supports buyer-side value construction, but it is not exposed to the agent as a separate observation field.

Persona-conditioned internal signals. The final internal step combines the bundle summary with hidden persona preferences. The simulator compares the bundle's feature-channel profile with the persona's hidden feature-preference weights to form a *feature-match signal*. It also retains the technology channel as a dedicated *technology signal*. These signals explain why the same bundle can carry different value for different personas. They are used inside the consumer value model and are not exposed to the agent.

Overall, the staged decomposition separates *what the configuration contains* from *how a particular persona reacts to it*. The agent-visible descriptors summarize selected options, price scale, cost proxy, and base appearance signal. The internal feature vector summarizes bundle composition, while the persona-conditioned signals connect that composition to hidden preferences. This keeps the value model interpretable while preserving the POMDP information boundary.

The benchmark also includes an optional **semantic extension** based on offline CLIP text semantics. This extension augments the agent-facing observation with a

configuration-level semantic vector and a projected scalar score derived from the selected options. These CLIP-based signals enrich the agent-visible bundle representation while the base simulator continues to use the same catalog, bundle construction logic, and hand-crafted aesthetic proxy.

Figure 4.3 summarizes this staged conversion from one fixed bundle to the value-relevant signals used later in the simulator. It distinguishes the descriptors that can enter the agent-facing observation from the internal summaries and persona-conditioned signals that remain inside the value model, while keeping the optional CLIP branch clearly secondary to the base pipeline.

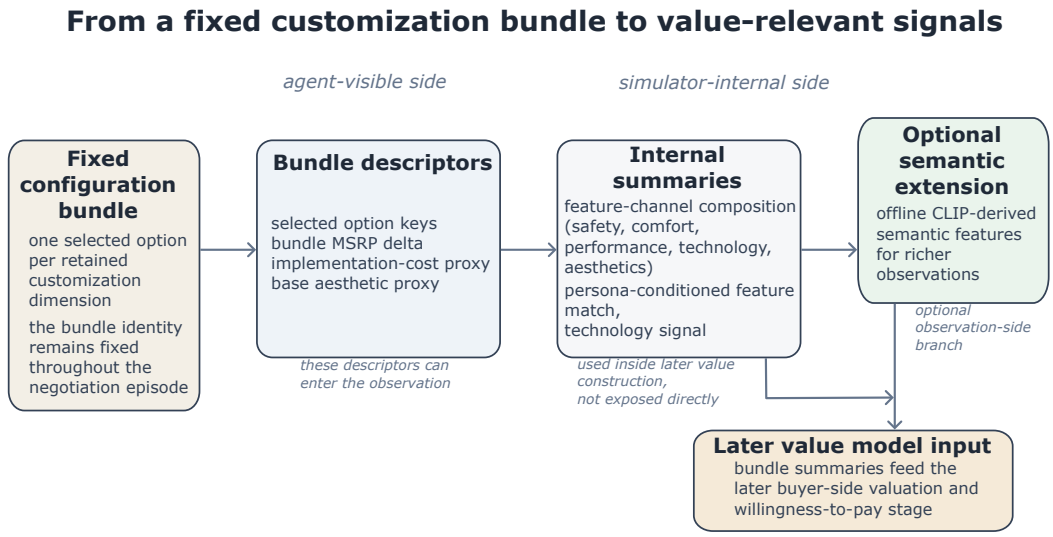

Figure 4.3: Staged conversion from one fixed customization bundle to the signals used later in buyer-side valuation. The figure separates agent-visible bundle descriptors from simulator-internal summaries and shows the optional semantic extension only as a secondary observation-side branch.

4.2.4 Dynamic willingness to pay and buyer utility

After computing the bundle-level signals, the consumer simulator uses them together with the current persona and round context to compute willingness to pay. This value, denoted by WTP_t, is not a fixed persona attribute: the same buyer and bundle can produce different values as bargaining progresses. The model is decomposed as follows:

$$\mathrm{WTP}_t = R_{\text{base}} + V_{\text{custom}} + V_{\text{aesthetic}} + V_{\text{brand-tech}} - V_{\text{fatigue}} + \varepsilon_t.$$

This decomposition is intentionally interpretable. Each term corresponds to a conceptually different part of buyer-side valuation, and together they turn the static persona record into a dynamic valuation process that evolves over the course of bargaining.

Baseline and customization value (R_{base} and V_{custom}). The first two terms capture

the buyer's general spending capacity and the perceived functional value of the current bundle. In our simulator, the baseline term R_{base} is anchored by the persona's hidden reservation-price base and rescaled by price sensitivity, so that more price-sensitive personas start from a lower effective valuation level. The customization term V_{custom} represents how much additional value the buyer places on the selected customization features. It is computed from the bundle's total MSRP anchor and a fixed value-markup factor, then adjusted by the persona-conditioned feature-match signal introduced in the previous subsection. In other words, V_{custom} is higher when the bundle's feature mix matches the buyer's hidden preferences.

Aesthetic and brand–technology value ($V_{\text{aesthetic}}$ and $V_{\text{brand-tech}}$). The next two terms capture value that is not purely functional. The aesthetic term $V_{\text{aesthetic}}$ combines the bundle's handcrafted appearance proxy with the persona's hidden aesthetic sensitivity, so visually expressive bundles matter more for buyers who are more sensitive to appearance. The brand–technology term $V_{\text{brand-tech}}$ combines hidden brand loyalty with the bundle's technology signal and the persona's tech-affinity band, allowing technology-heavy bundles to matter more for technology-oriented buyers.

Fatigue and stochasticity (V_{fatigue} and ε_t). The final two terms make the model dynamic across rounds. The fatigue term V_{fatigue} increases with round progression and is modulated by the persona's patience and impulsivity, so prolonged bargaining gradually reduces effective willingness to pay for some consumers more than for others. The noise term ε_t introduces round-level uncertainty through zero-mean Gaussian variation, with magnitude controlled by the persona's belief obscurity. This keeps buyer responses from becoming a fully deterministic threshold rule, while still preserving the same underlying persona structure.

Once round-level willingness to pay has been computed, the simulator derives *buyer utility* for a proposed transaction price p_t as

$$U_t^{\text{buyer}}(p_t) = \text{WTP}_t - p_t.$$

This quantity provides the immediate buyer-side interpretation of the current offer. Positive utility means that the offered price is at or below the buyer's current latent valuation, whereas negative utility indicates that the offer exceeds what the buyer is willing to pay in that round.

The negotiation backend uses this buyer utility as the main behavioral reference for response generation. When buyer utility is non-negative, acceptance becomes more likely. When utility is negative, the buyer may reject, walk away, or issue a counter-offer depending on the utility gap and additional hidden bargaining traits such as walk-away threshold and counter strength. The simulator does not use willingness to pay as a direct response rule; it first converts willingness to pay into buyer utility, and then uses that utility to guide

the observable buyer response.

This links the consumer-side valuation model to the interaction environment: bundle signals become round-level valuation, valuation becomes buyer utility, and buyer utility shapes the response observed by the agent. The next section embeds these utility-driven responses into the multi-round negotiation loop.

4.3 Negotiation environment

The negotiation environment is where the buyer persona and the pricing agent are brought into direct interaction. It integrates the consumer-side components developed above into a multi-round bargaining process, so that bundle descriptors, hidden buyer traits, and dynamic willingness to pay are translated into observable negotiation behavior. This section explains how one concrete negotiation episode is instantiated and evolves inside the benchmark, including the bargaining flow, the persistent backend, and the realization of terminal outcomes and traces.

4.3.1 Episode structure and bargaining flow

Operationally, each episode instantiates the task definition from Chapter 3 by resetting the environment with one consumer persona and one customization bundle. The persona is sampled from the selected split of the fixed offline persona bank, while the bundle is sampled from the fixed vehicle catalog by selecting one option from each retained customization dimension. After reset, the bundle remains fixed and only the bargaining over price evolves. This keeps product variation controlled within the episode, while buyer-side heterogeneity enters through hidden preferences, price sensitivity, patience, and bargaining tendencies.

After persona and bundle selection, the environment creates one persistent bargaining session and initializes the episode state. The round index is set to the first negotiation round, the last-offer fields are empty, and the buyer-side latent variables remain internal to the simulator. At this stage, the agent receives only the initial observation, which includes the selected configuration, observable persona descriptors, and an empty negotiation history.

The within-episode interaction then proceeds round by round. At each step, the environment first computes the current buyer-side willingness to pay from the persona, the fixed bundle, and the current round context. The pricing agent then chooses one of the valid negotiation moves: issue a new offer, accept the latest buyer counter-offer, or walk away. When the move is an offer, the environment submits the quoted price into the bargaining session and obtains a buyer response.

If the episode remains active after the current interaction, the negotiation state is updated and the next round begins. The offer history, latest response, and any buyer counter-offer are carried forward, so later actions are conditioned on what has already

happened. In this way, one episode is realized as a persistent bargaining trajectory rather than as a sequence of independent quotation decisions. The detailed terminal outcomes and their realization are described in Section 4.3.3.

4.3.2 Persistent NegMAS bargaining backend

NegMAS [5] is used to maintain a persistent buyer–seller bargaining session throughout each episode. This keeps offers, counter-offers, and responses inside one evolving interaction, so later rounds remain connected to the earlier bargaining history.

Within this session, the seller acts through explicit offer actions, while the buyer side is driven by the utility model defined in the previous subsection. At each round, the environment updates the buyer's current willingness to pay and hidden bargaining parameters, then submits the seller's offer into the continuing session. The simulated buyer may accept, reject, end the negotiation, or propose a counter-offer. The backend therefore turns the utility-driven consumer model into observable negotiation behaviour.

The backend also keeps the negotiation space focused and interpretable. Price is the only issue under negotiation, and the admissible price range is bounded at episode initialization according to the selected bundle and the sampled buyer's baseline reservation context. This matches the benchmark scope: dynamic pricing over a fixed customized package.

The persistent backend provides two practical benefits for the benchmark. It preserves continuity across rounds, and it gives the interaction clear negotiation semantics: the seller proposes prices explicitly, while the buyer responds through a structured utility-driven bargaining mechanism.

4.3.3 Termination, reward realization, and trace logging

The environment distinguishes several concrete outcome types. A deal is reached when the buyer accepts the seller's current offer inside the bargaining session or when the seller accepts the buyer's latest counter-offer. No-deal outcomes arise when the seller walks away, when the buyer ends the negotiation, or when the effective horizon is exhausted. An *invalid accept* is treated as a recorded negotiation event rather than as a terminal outcome, so the episode may continue.

The environment then turns these outcomes into recorded episode results. When agreement is reached, it records the final transaction price and the realized deal profit, computed as the agreed price minus the benchmark-level implementation-cost proxy of the selected bundle. When no agreement is reached, the episode ends without realized deal profit. The environment may also emit event-level signals for specific outcomes such as invalid accept, rejection, counter, walk-away, or timeout. For cross-method comparison, however, the benchmark follows the common task-level reward objective introduced in Chapter 3: detailed negotiation events are resolved by the environment, while reported comparison follows a consistent profit-oriented task definition.

Termination is also governed by the benchmark horizon. The environment maintains a global maximum number of rounds, but the effective episode length is further bounded by the sampled buyer's patience. As a result, timeout is not only a generic cutoff mechanism; it is also one way in which buyer-specific tolerance for extended bargaining affects the interaction outcome.

In addition to outcome realization, the environment records detailed traces for later analysis. At each step, it stores a compact event record containing the round index, the seller move, the offered price when applicable, the buyer response, any counter-offer, and the termination cause when the episode closes. At the end of the episode, these step-level events are aggregated into an episode trace and accompanied by summary metrics such as whether a deal was reached, the final profit, the number of rounds used, and whether the interaction ended in walk-away. These records are useful because they make the negotiation process inspectable without changing the observation boundary seen by the agent.

4.4 Benchmark protocol

Having defined the customization scope, the consumer simulator, and the negotiation environment, we now turn to the protocol used to evaluate pricing policies on top of this environment. This section explains how the benchmark supports cross-method comparison, reproducible evaluation, and interpretable reporting. It clarifies how shared episode streams are constructed, how the evaluation setting is controlled across methods, and how economic performance and negotiation dynamics are reported.

4.4.1 Shared episode contract

Our benchmark uses a shared episode contract so that different methods are evaluated on the same underlying population and the same episode stream. The persona bank is divided into training, validation, and test splits, and these splits play distinct roles in the evaluation protocol. Learning-based policies are trained on the training split, the validation split can be used for model or setting selection where needed, and the test split is reserved for final benchmark reporting. This follows the normal logic of experimental separation while remaining compatible with both heuristic and learning-based policies.

Within one benchmark report, *the same episodes* means more than simply using the same split. It means that compared policies are evaluated under the same persona-bank split, the same episode-seed list, and the same environment setting. In consequence, they face the same sampled buyer identities, and under the shared reset-seed protocol they also face the same sampled customization bundles in the same order. The benchmark explicitly verifies shared consumer identity through stable *persona IDs*, so that fairness is not left as an informal assumption.

This shared-episode design is important for reliable comparison. Without it, differences in performance could partly reflect differences in sampled consumers rather than genuine differences in pricing quality. By enforcing a common episode stream, the benchmark makes cross-method comparison easier to interpret.

Figure 4.4 summarizes this protocol logic. Frozen benchmark assets, shared episode generation, and a common environment interface ensure that heuristic and learned policies are compared under the same external contract, and that their episode-level outcomes are aggregated through the same metrics.

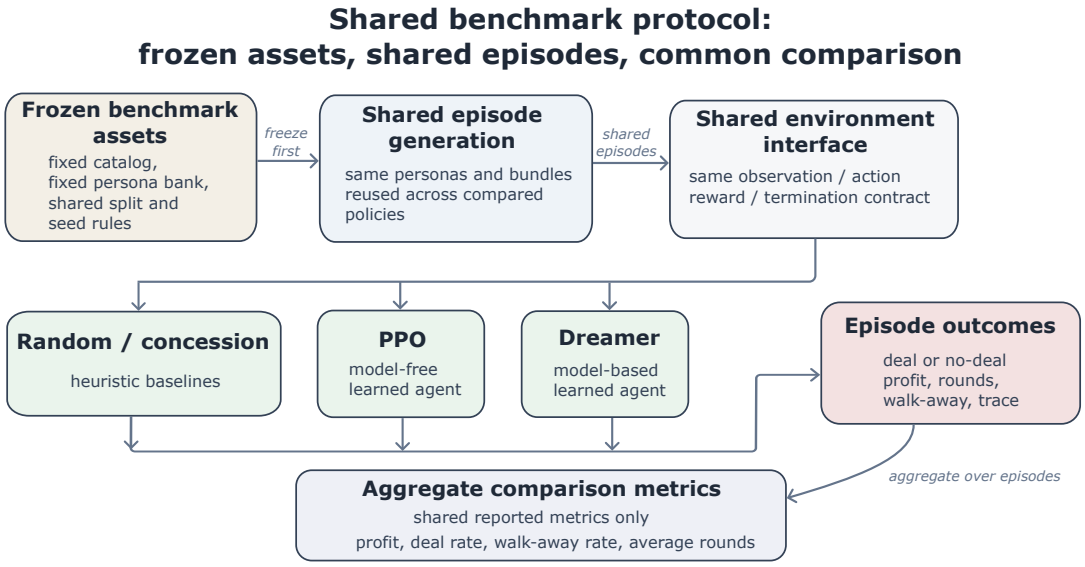

Figure 4.4: Shared benchmark protocol linking frozen benchmark assets, shared episode generation, a common environment interface, episode-level outcomes, and the aggregate metrics used for cross-method comparison.

4.4.2 Reproducibility and evaluation control

Reproducibility is a core principle of the benchmark design. We freeze the persona bank rather than relying on online resampling, so that the evaluated population remains stable across repeated experiments and across different methods. The benchmark also fixes the evaluation split, the global seed, and the episode count, allowing the same sequence of negotiated episodes to be reconstructed when needed.

This control matters because the compared policies differ internally. Heuristic baselines, PPO-style agents, and Dreamer-style agents do not share the same training procedure, but they must still be evaluated under the same external protocol. What matters at the benchmark level is that they face the same episode stream, the same observation boundary, and the same task objective. Evaluation control is therefore part of the comparison design.

Benchmark records also preserve enough metadata for later inspection. Alongside

aggregate results, the benchmark records the evaluation split, the seed, the compared policies, and whether the shared-consumer condition was verified across methods. This makes the benchmark suitable not only for one-time reporting, but also for repeated analysis and later extension.

4.4.3 Metrics and reporting

The benchmark uses *average profit in USD* as the primary metric because it aligns directly with the seller's objective in the task. Under this protocol, only successful deal episodes contribute realized profit, while no-deal outcomes contribute zero profit. This makes profit a natural summary of both pricing quality and the ability to close agreements under negotiation.

Alongside profit, the benchmark reports several negotiation-dynamics metrics, including *deal rate*, *walk-away rate*, and *average rounds used*. These secondary diagnostics are useful because two policies with similar profit may still behave quite differently. One policy may achieve profit through frequent agreements and moderate prices, while another may reach similar profit with fewer deals, longer bargaining, or a higher walk-away rate. These additional indicators make the benchmark more interpretable.

The reporting logic is intentionally two-layered: profit is treated as the primary task-level outcome, while deal dynamics are treated as supporting diagnostics. This balance matches the benchmark objective: to assess whether a pricing policy performs well economically while remaining understandable through a small set of behaviorally meaningful indicators.

Chapter 5

Pricing agents

This chapter turns to the pricing-agent side of the benchmark. Whereas the previous chapter explained how the consumer simulator and negotiation environment are instantiated, this chapter describes how different decision-making agents are developed and connected to that same benchmark. The chapter considers three broad agent families: heuristic baselines, PPO-based agents, and DreamerV3-based agents.

5.1 Agent taxonomy and design goals

This section introduces the agent families evaluated in the benchmark and clarifies the main design principles behind their comparison. It explains why the thesis considers multiple types of pricing agents and why all of them must be studied against the same fixed benchmark.

5.1.1 Why compare multiple agent families

Because the main contribution is a benchmark rather than a single final pricing policy, multiple agent families need to be evaluated on the same task. A benchmark is only informative when it can support comparison across methods with clearly different levels of inductive bias and learning capacity. For this reason, the chapter includes non-learning heuristic baselines, a model-free reinforcement-learning agent, and a model-based world-model agent. Together, these methods provide a useful spread from simple rule-based behaviour to learned history-dependent control.

This comparison is organized along two main dimensions. The first is *heuristic versus learning-based* decision making: heuristic agents rely on fixed negotiation rules, whereas learning-based agents improve their behaviour from interaction data. The second is *model-free versus model-based* learning: PPO learns a pricing policy directly from experience without an explicit dynamics model, while DreamerV3 learns a latent model of the negotiation process and optimizes behaviour through that learned representation. Despite these differences, all agents are evaluated against the same seller-side objective under partial observability, namely profit-oriented sequential decision making in the POMDP task defined in Chapter 3.

The spread of agent families makes the benchmark more informative. Random and concession-style policies provide interpretable reference points and sanity checks, while PPO and DreamerV3 test whether stronger learning methods can make better use of interaction history and latent structure. The benchmark is therefore not tied to one preferred algorithmic paradigm; instead, it is designed to compare distinct agent classes under one shared evaluation setting.

5.1.2 Separation between benchmark and agent

An important design principle is the separation between the benchmark environment and the pricing agent. The consumer simulator, negotiation dynamics, action semantics, and task-level objective are fixed independently of any particular pricing method. Heuristic, PPO-based, and DreamerV3-based agents therefore operate on the same benchmark.

This separation keeps the role of the agent clear: different methods vary only in how they make pricing decisions from the available observations and interaction history. As discussed in Chapter 4, shared evaluation conditions are essential for fair comparison. Here, the same principle is applied at the agent interface level: different agent families are connected to one common task definition.

To support this shared comparison, the benchmark provides a standardized *Gymnasium* [31] agent-facing interface. The wrapper keeps the negotiation logic inside the underlying environment, but converts observations into fixed-size normalized vectors, maps learning-agent actions back into benchmark move and price semantics, and exposes the standard reset/step interaction pattern. This allows PPO- and Dreamer-based agents to use the same negotiation environment without changing the benchmark task definition.

5.2 Heuristic baselines

Heuristic baselines are agents that follow fixed decision rules rather than learning from interaction data. They are training-free and provide interpretable reference points for evaluating stronger learning-based methods. Our benchmark includes two heuristic baselines: a random policy and a concession-style policy. Together, they serve different purposes. The random policy acts as a minimal sanity baseline, while the concession policy provides a more meaningful non-learning bargaining strategy against which learned agents can be compared.

5.2.1 Random policy

The random policy behaves in accordance with its name: it samples actions directly from simple stochastic rules over the benchmark action space. In most steps, it generates a random offer within the admissible offer range; when a buyer counter-offer is available, it occasionally accepts that counter-offer, and it also retains a small probability of walking away. It therefore does not attempt to model buyer behaviour, negotiation history, or future consequences in any structured way.

The random policy serves as a sanity-check baseline and lower-bound reference point. Stronger agents are expected to outperform it if they are using useful structure from buyer responses, negotiation history, or future consequences.

5.2.2 Concession policy

The concession policy is a rule-based bargaining agent that starts from a relatively high target price and gradually moves toward a lower floor as the episode progresses. This makes later-round offers more accommodating than early-round offers. When a buyer counter-offer is present, the policy also uses that counter-offer to revise its next proposal instead of treating each round as an isolated quotation step.

This baseline is more informative than the random policy because it captures a recognizable bargaining heuristic. It reflects the idea that sellers may begin with higher margins and then move toward agreement as the deadline approaches. The concession policy also contains a simple late-round acceptance rule when the buyer's counter-offer remains above the policy's floor. It therefore serves as the strongest non-learning baseline in the benchmark: it remains hand-crafted, but already uses the multi-round structure of the task.

5.3 PPO pricing agent

PPO (Proximal Policy Optimization) [23] is the model-free reinforcement-learning baseline used in the benchmark. The PPO agent learns negotiation behaviour directly from interaction experience without building an explicit model of environment dynamics.

5.3.1 Learning interface

The PPO agent does not interact with the raw simulator directly. Instead, it uses a PPO-facing environment adapter built on top of the common negotiation wrapper. The adapter keeps the underlying benchmark dynamics unchanged, but presents the task in a stable learning format. PPO therefore receives the same observable persona descriptors, bundle-level signals, and negotiation-history information defined by the benchmark observation contract, while the hidden buyer state remains unobserved.

On the action side, PPO uses a MultiDiscrete action space with shape $[3, 301]$. The first action component selects the negotiation move, namely *offer*, *accept*, or *walkaway*. The second component selects a price-delta bin. In the benchmark setting considered here, the price-bin count is 301, with a step size of 20 USD between adjacent bins. The center bin corresponds to zero adjustment, while bins on either side represent positive or negative changes relative to a rolling reference price.

This reference price is not fixed for the whole episode. At reset, it is initialized from a bundle-scale price anchor, and during negotiation it is updated according to the latest available bargaining information, in particular the buyer's latest counter-offer or the

agent's latest quoted offer. The discrete action therefore remains tied to the current negotiation context rather than to a static global price grid. Although only *offer* requires an explicit price choice, the benchmark uses one unified fixed-size action interface for all three move types. The price-bin component is ignored when the selected move is *accept* or *walkaway*, which keeps the PPO interface fixed-size while preserving the pricing semantics of the benchmark.

5.3.2 Recurrent policy and PPO optimization

In this benchmark, the PPO baseline is implemented with *RecurrentPPO* using an *MlpLstmPolicy*. The recurrent architecture is important because the negotiation process is both partially observable and multi-round. At any given step, the agent observes only the current benchmark observation, not the buyer's hidden state or full future response model, so the LSTM provides a compact memory mechanism for integrating previous offers, responses, and counters when making the next decision.

PPO learns a policy over the discrete negotiation actions while also learning a value estimate for expected return. It is therefore a history-aware but model-free baseline: it can use temporal interaction structure through recurrence, but it does not learn an explicit transition model of the negotiation environment. This makes it a useful reference point for testing how far experience-driven policy optimization can go under the benchmark observation boundary.

Across PPO variants, the optimization setup is kept shared for reproducibility. PPO is optimized to maximize expected seller-side return under the profit-oriented reward, making it the principal model-free learning baseline for negotiation under partial observability.

5.3.3 Reward setting

For the main benchmark comparison, PPO uses a pure-profit reward setting. Additional penalties for no-deal outcomes, low-profit deals, and invalid accepts are set to zero, so the PPO learning reward reduces to a sparse terminal signal:

$$r_t^{\mathrm{PPO}} = \begin{cases} \dfrac{\mathrm{profit}_t}{1000}, & \text{if the episode terminates with a deal,} \\ 0, & \text{otherwise.} \end{cases}$$

Here, profit_t is the realized deal profit in USD, and the divisor 1000 is a fixed normalization constant used for numerical scaling. This reward aligns PPO training with the seller-side objective defined in Chapter 3: profitable agreement is rewarded, while all non-deal or non-terminal steps receive zero reward.

This keeps PPO as a model-free baseline under the benchmark's sparse profit-oriented objective. Richer auxiliary shaping is treated later as an ablation rather than as the default PPO setting.

5.4 DreamerV3 pricing agent

DreamerV3 [26] provides the model-based reinforcement-learning baseline in the benchmark. It learns a latent dynamics model of the negotiation process and uses that model to optimize behaviour through imagined rollouts. This makes it the main model-based counterpart to the heuristic baselines and the PPO-based model-free agent.

5.4.1 Learning interface

DreamerV3 is connected to the benchmark through a Dreamer-specific adapter built on top of the same Gymnasium-compatible interface family used by PPO. The adapter keeps the benchmark observation contract intact: Dreamer receives the same agent-visible persona descriptors, bundle-level signals, and negotiation-history information as the other learning-based agents. The difference is in the interface format required by the world-model pipeline, not in the underlying negotiation task.

On the action side, the Dreamer adapter maps the same move-plus-price-bin structure used by PPO into one flat discrete action space. Concretely, the PPO tuple action $(\textit{move}, \textit{delta bin})$ is flattened through a deterministic codec into a single *Discrete(903)* action index, corresponding to 3×301 action combinations. This format matches the Dreamer implementation while preserving the benchmark's negotiation moves and price-bin granularity.

5.4.2 Latent dynamics and policy optimization

Partial observability and multi-round bargaining make latent dynamics modeling a natural direction to evaluate in this benchmark. Unlike PPO, which optimizes a recurrent policy directly from interaction histories, Dreamer learns an internal latent representation of the negotiation dynamics and improves behaviour through imagined rollouts in that latent space. This makes it a suitable model-based baseline for studying whether world-model-style learning can make better use of the benchmark's sequential structure.

The learned representation should be read as an agent-side control state. It combines latent dynamics, actor, and value components that together support decision making under partial observability. The model is not designed to expose symbolic estimates of quantities such as the buyer's exact reservation price or true sensitivity parameters. Instead, its latent state compresses the observable negotiation history into a representation useful for predicting consequences and selecting actions.

This role is closely aligned with the benchmark setting. The negotiation task is sequential, partially observable, and strongly history-dependent: the agent must act under latent buyer state, while each response reveals only incomplete information about the underlying bargaining situation. A world-model baseline gives the agent a structured way to organize this partial history before action selection. It also tests whether explicit latent-dynamics learning can exploit the benchmark's hidden-state structure beyond what

a recurrent model-free policy can achieve.

From the benchmark perspective, DreamerV3 plays a complementary role to PPO. PPO tests how far a recurrent model-free learner can go using history-dependent policy optimization, whereas Dreamer tests whether a learned latent model of negotiation dynamics offers additional value under the same observation boundary and action semantics. It is used here as the benchmark's principal model-based learning baseline.

5.4.3 Reward setting

DreamerV3 uses the same sparse profit-oriented reward as PPO in the main benchmark comparison:

$$r_t^{\text{Dreamer}} = \begin{cases} \dfrac{\text{profit}_t}{1000}, & \text{if the episode terminates with a deal,} \\ 0, & \text{otherwise.} \end{cases}$$

This reward gives Dreamer the same seller-side objective used by the other learning baseline: profitable agreement produces positive terminal reward, while no-deal and non-terminal steps receive zero reward. Within DreamerV3, this signal is used to train the world model, value function, and actor through the standard latent rollout objective.

Using the same reward definition keeps the PPO–Dreamer comparison focused on the learning mechanism rather than reward design. PPO optimizes a recurrent model-free policy, while Dreamer optimizes behaviour through a learned latent dynamics model. Additional Dreamer reward variants are treated later as ablations rather than as the default benchmark setting.

5.5 Modular extensions

Beyond the base comparison among heuristic, PPO-based, and DreamerV3-based agents, the benchmark also includes two optional extensions. These extensions keep the same consumer simulator, negotiation mechanics, and task objective, but add extra agent-side mechanisms for studying richer observation and decision-time adaptation. The two extension directions considered here are CLIP-based semantic observation augmentation and inference-time adaptation.

5.5.1 Extension philosophy

The extensions are designed to be modular. The base benchmark remains the primary comparison setting, and each extension can be enabled or removed without changing the task definition, the consumer simulator, or the negotiation protocol. This makes it possible to study the effect of each added mechanism while keeping the underlying pricing problem fixed.

This separation also helps interpretation. If an extension improves performance, the improvement can be attributed to the added semantic signal or decision-time adaptation

layer, rather than to a different environment or reward definition. The extensions are therefore treated as controlled add-ons to the base benchmark, not as separate benchmark tasks.

5.5.2 CLIP semantic observation extension

Because *aesthetic preference* is one of the latent buyer dimensions in the simulator, the benchmark also tests whether richer semantic descriptions of customization options can help pricing agents. The optional CLIP [32] extension is built offline from text-only prompts rather than image inputs. Each catalog option receives a CLIP-derived semantic embedding, and the embeddings of the selected options are aggregated into configuration-level semantic features.

These semantic features are added only to the agent-visible observation space. They provide an additional representation of the selected bundle, while the consumer value model, negotiation mechanics, and hand-crafted aesthetic prior remain unchanged. CLIP is therefore used as an observation-side augmentation module for the pricing agents, not as a replacement for the benchmark's base appearance proxy.

5.5.3 Inference-time adaptation

In addition to CLIP, we include an inference-time adaptation module inspired by the broader idea of test-time adaptation [33]. The module is implemented as a training-free decision-time layer on top of an already trained Dreamer policy. At each step, the Dreamer backbone first proposes a raw action. The adaptation layer then reads the current observable negotiation context, including recent offer and counter-offer history, round progression, and response signals. These observations are used to maintain a compact per-episode belief state over quantities such as an estimated acceptable price ceiling, counter aggressiveness, and walk-away risk.

The adaptation layer uses bounded candidate reranking. Starting from the raw Dreamer action, it constructs a small local candidate set, mainly consisting of nearby offer prices and, when supported by the current negotiation state, possible *accept* or *walkaway* actions. These candidates are then re-ranked using proximity to the raw policy choice, seller-side margin preference, observable deal feasibility, and rejection risk. In the final TTA v2 setting used for the benchmark, this mechanism acts primarily as a selective offer-side correction layer. The final selected action is executed in the same fixed environment. Policy parameters, hidden buyer variables, and environment dynamics remain unchanged, so the module tests whether lightweight agent-side inference can improve decision quality under the same benchmark task.

Chapter 6

Experiments

This chapter presents the experimental evaluation of the benchmark. It compares different pricing agents under one shared protocol, reports both core benchmark results and extension analysis, and then interprets what these results suggest about the benchmark setting.

6.1 Experimental setting and protocol

The experiments reported in this chapter follow the benchmark protocol defined in Chapter 4 and the agent implementations described in Chapter 5. All methods are evaluated on the same fixed benchmark setting, including the shared test split of the frozen persona bank, the same episode stream under controlled seeds, and the same profit-oriented task objective. The compared methods include the two heuristic baselines, the PPO-based variants, and the DreamerV3-based variants together with selected extensions. This keeps the comparison aligned across agent families while avoiding a full restatement of the benchmark and method design.

Fairness is enforced through shared evaluation controls rather than through method-specific adjustment. All benchmark comparisons in this chapter use the same test split, the same episode count, and the same seed-driven episode stream. Each reported benchmark table is based on 5000 test episodes, and the benchmark uses the same-consumer protocol described earlier in the thesis so that compared policies are evaluated on the same underlying population rather than on independently drifting samples.

The learning-based agents are reported under one fixed training budget per family. PPO is trained for 600,000 interaction steps, while DreamerV3 is trained for 2,000,000 steps under one fixed world-model preset. These settings define the retained benchmark comparison rather than an exhaustive search for the strongest possible checkpoint in each family. PPO and Dreamer do not share the same optimization dynamics, so identical training budgets would not by themselves guarantee a more meaningful comparison. What matters here is that each family is evaluated under one consistent and explicitly reported

training regime, using the latest checkpoint produced by that setup. Table 6.1 summarizes these settings for reference rather than as the outcome of a separate hyperparameter search.

Table 6.1: Common training settings for the reported learning-based agents.

Component	PPO family	DreamerV3 family
Training split	train	train
Validation split	val	val
Benchmark split	test	test
Total training budget	600,000 steps	2,000,000 steps
Action discretization	301 bins, 20 USD step	301 bins, 20 USD step
Price markup at reset	1.4	1.4
Parallel environments	4	1
Batch setting	batch size 512, rollout 256	batch size 8, sequence length 32
Learning rate / train ratio	2×10^{-4}	train ratio 48
Discount setting	$\gamma = 0.99$, $\lambda = 0.95$	world-model preset size4m
Checkpoint/report basis	fixed ablation config	fixed ablation config

Some entries in Table 6.1 require brief clarification. The *price markup at reset* denotes the multiplier applied to the bundle-scale price anchor when the initial offer reference is constructed at the start of an episode, so that the initial bargaining scale is aligned across methods. The number of *parallel environments* refers to rollout-collection concurrency during training. For PPO, the *discount setting* reports the standard return-discount and advantage-estimation parameters used during policy optimization. For DreamerV3, *train ratio* indicates the amount of world-model and policy updating performed relative to environment interaction, while *size4m* refers to the fixed model-capacity preset used in the reported runs.

6.2 Evaluation metrics

As defined in Chapter 4, the benchmark is evaluated primarily by average profit in USD, because this metric aligns directly with the seller-side objective of the task. The experiments also report deal rate, walk-away rate, and average rounds used. Together, these metrics help explain whether a method reaches its profit level through frequent agreement, stronger margin preservation, or different bargaining length under the same negotiation protocol.

6.3 Main benchmark results

Table 6.2 summarizes the main cross-method comparison under the shared benchmark protocol. To keep the core comparison focused, this table includes the two heuristic references together with one representative PPO variant and one representative Dreamer variant. The learned variants shown here are *PPO + CLIP* and *Dreamer + CLIP*, which serve as the retained base learned variants under the shared pure-profit setting. They are compared under the same benchmark protocol, the same evaluation split and episode stream, and one fixed training setup for each agent family. Inference-time adaptation is treated separately as an optional extension, and later sections also examine reward-setting and module-level extensions in a more detailed ablation analysis.

Table 6.2: Core benchmark comparison on the test split under the shared thesis protocol (5000 episodes, same-consumer evaluation).

Method	Avg Profit (USD) ↑	Deal Rate ↑	Walkaway Rate ↓	Avg Rounds
Random	6058.8570	0.5796	0.4204	1.3208
Concession	**14725.4262**	0.7244	0.2756	1.7196
PPO + CLIP [23]	11972.5010	0.8212	0.1788	2.3572
Dreamer + CLIP [26]	12286.3688	**0.8664**	**0.1336**	**1.6010**

At a high level, the table shows that the benchmark is neither trivial nor dominated by a single weak baseline. The random policy remains clearly inferior to all stronger methods, while the concession baseline still achieves the highest average profit. The benchmark therefore remains challenging for the learned agents and still leaves room for stronger pricing policies.

From a training perspective, PPO was comparatively easier to stabilize than Dreamer, while the Dreamer side remained more sensitive to training budget and configuration details. The table should therefore be read as the benchmark outcome under one retained comparison setting, not as evidence that training convergence is fully resolved across agent families.

6.4 Quantitative analysis

6.4.1 Profit and deal rate

The most immediate pattern in Table 6.2 is that profit and deal rate do not move together in a simple way in this benchmark. The concession baseline achieves the highest average profit, at 14725.4262 USD, while Dreamer + CLIP achieves the highest deal rate, at 0.8664. PPO + CLIP lies between these two extremes: it closes substantially more deals than the heuristics, but it does not match the strongest profit result.

This separation clarifies what the benchmark is actually testing. Figure 6.1 shows

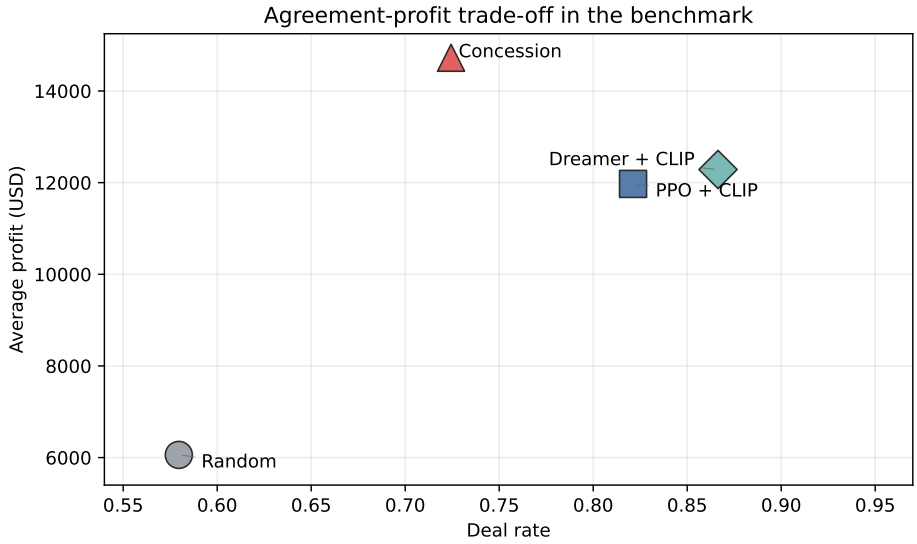

Figure 6.1: Agreement-profit trade-off in the main bank50k benchmark comparison across the four core methods. The figure shows that concession achieves the strongest average profit without the highest deal rate, while Dreamer + CLIP reaches more agreements without matching concession on profit. Agreement frequency and seller-side return therefore do not move together in a simple one-to-one way in this benchmark.

the same pattern visually: higher agreement does not automatically imply higher seller-side return. The methods appear to trade agreement and margin in different ways. In other words, the benchmark distinguishes between *closing deals* and *closing profitable deals*, rather than collapsing both behaviours into a single success signal. The behavioural meaning of this trade-off is examined next.

6.4.2 Negotiation styles

The main methods also differ in how they appear to use the negotiation process itself. The random policy terminates very quickly, with only 1.3208 rounds on average, and also has by far the highest walk-away rate, at 0.4204. This is consistent with its role as a sanity baseline: it does not exploit interaction history in any structured way, so many episodes end early without productive bargaining. The concession policy shows a different pattern. It achieves the highest average profit, but not the highest deal rate, which suggests a bargaining style that preserves seller-side margin while still maintaining a credible path to agreement.

The two learned agents occupy a different part of the trade-off space. PPO + CLIP reaches a higher deal rate than the heuristic baselines, but it also uses the largest number of rounds, at 2.3572, suggesting that it remains in negotiation longer before resolution. Dreamer + CLIP, by contrast, combines the highest deal rate, at 0.8664, with the lowest walk-away rate, at 0.1336, while still terminating in fewer rounds than PPO. This points to a more agreement-oriented style: Dreamer converts more negotiations into deals, but it does not preserve the same seller-side margin as the concession policy. The benchmark

horizon is only five rounds, yet all methods terminate well before exhausting it on average. The challenge is therefore not to sustain long bargaining, but to use a small number of rounds to infer buyer behaviour, decide how aggressively to price, and balance agreement against margin.

6.4.3 What the main results imply

Taken together, the main results show that the benchmark remains competitive for both heuristic and learned methods. The random baseline stays clearly inferior, which is the expected sanity outcome, but the stronger heuristic baseline remains highly competitive. In particular, the concession policy still achieves the highest average profit even when compared against the strongest retained base learned variants. This is an important benchmark-level finding. It shows that the task is not solved merely by introducing a recurrent policy or a world-model agent. Meaningful performance still depends on how well a method uses limited interaction to preserve margin while maintaining a credible path to agreement.

The results also suggest that the hardest part of the benchmark is not simply deciding whether to continue bargaining, but using a small number of rounds to infer hidden buyer characteristics and adapt price accordingly. A stronger seller policy would use early interaction to estimate reservation-price level, price sensitivity, and bargaining tendency, and then translate that information into more targeted pricing later in the episode. The current learned agents do not yet exhibit this behaviour in a fully convincing way. Dreamer + CLIP closes more deals, but it does not overtake concession on profit; PPO + CLIP improves over the weaker baselines, but it still uses relatively many rounds without turning that additional interaction into the best return. This gap between agreement-seeking behaviour and profit-maximizing behaviour is precisely what makes the benchmark informative. It indicates that the task is sensitive to hidden-state inference, finite-horizon reasoning, and sparse seller-side reward, rather than rewarding simple agreement frequency alone.

6.5 Extension analysis

6.5.1 CLIP effect

Table 6.3 isolates the CLIP comparison within each learning-based family by comparing the base variant against its CLIP-augmented counterpart. In this benchmark, CLIP acts as an observation-side extension: it augments the agent-visible bundle description with offline semantic features derived from the selected options, without changing the task definition or the consumer simulator. A fuller description of the module is given in Section 5.5.2.

Table 6.3: CLIP ablation within the two learning-based agent families under the shared benchmark protocol.

Variant	Avg Profit (USD) ↑	Deal Rate ↑	Walkaway Rate ↓	Avg Rounds
PPO Base	11601.1730	0.8004	0.1996	2.5442
PPO + CLIP	11972.5010 (+371.3280)	0.8212 (+0.0208)	0.1788 (-0.0208)	2.3572 (-0.1870)
Dreamer Base	8318.7230	0.5730	0.4270	3.7310
Dreamer + CLIP	12286.3688 (+3967.6458)	0.8664 (+0.2934)	0.1336 (-0.2934)	1.6010 (-2.1300)

The CLIP extension has different effects across the two learning-based agent families. For PPO, adding CLIP produces a modest improvement in profit, increasing average profit by 371.3280 USD together with a higher deal rate and a lower walk-away rate. For Dreamer, however, the improvement is substantially larger: the move from the base Dreamer variant to Dreamer + CLIP increases average profit by 3967.6458 USD and also produces a much higher deal rate and a lower walk-away rate. This pattern suggests that richer semantic signals can benefit both learning-based agents in the benchmark, but that the effect is much stronger for the model-based agent than for the recurrent model-free baseline.

The contrast is informative at the benchmark level. PPO [23] already has access to the same interaction history and improves slightly when the observation is enriched, but the gain remains limited. Dreamer [26], by contrast, starts from a much weaker base result and then improves sharply once the same semantic augmentation is added. One reasonable interpretation is that the base Dreamer observation is still too coarse for the world model to organize bundle-level value differences effectively, so the learned latent dynamics do not yet translate into strong pricing behaviour. When CLIP features are added, the bundle representation becomes more informative, and the world-model agent appears better able to connect interaction history with the semantic structure of the configuration under negotiation. In this sense, the CLIP result is not only an extension result; it also shows that the benchmark is sensitive to observation quality and representation richness.

Taken together, the CLIP result is best read as an extension-side representation check. It shows that the benchmark responds to the quality of agent-visible bundle representation, especially for the model-based agent, without changing the underlying task itself.

6.5.2 Inference-time adaptation effect

We also examine whether inference-time adaptation (TTA) can further improve Dreamer's performance once the policy already uses CLIP-augmented observations. Here, TTA is a training-free bounded candidate-reranking layer applied at decision time, using only observable negotiation signals from the current episode. A fuller description of the module is given in Section 5.5.3. Table 6.4 reports a paired comparison under the strongest retained Dreamer + CLIP reward profile, namely the *soft-shortfall* setting. This reward

profile is defined in detail later in the Dreamer reward-design analysis. Both rows use the same CLIP-enabled observation extension, the same benchmark protocol, the same backbone checkpoint, and the same soft-shortfall reward profile. The only intended difference is whether the inference-time reranking module is enabled.

Table 6.4: Dreamer-side inference-time adaptation comparison under the CLIP + soft-shortfall setting.

Case	Avg Profit (USD) ↑	Deal Rate ↑	Walkaway Rate ↓	Avg Rounds
(1)	12616.0280	0.9106	0.0894	1.2864
(2)	**12906.3440** (+290.3160)	**0.9404** (+0.0298)	**0.0596** (-0.0298)	**1.1500** (-0.1364)

Case (1): Dreamer + CLIP + soft shortfall. Case (2): Dreamer + CLIP + soft shortfall + TTA.

Under this paired comparison, TTA improves all four reported metrics. Average profit increases by 290.3160 USD, deal rate rises by 0.0298, walk-away rate falls by 0.0298, and average rounds decrease by 0.1364.

This pattern matches the role of the module. The TTA mechanism is a training-free inference-time reranking layer over a small local action set, using only observable negotiation signals rather than hidden persona state or online parameter updates. In the benchmark report, only about 3% of prediction steps are actually changed, and the recorded overrides are offer-side corrections rather than broad policy replacement. The gain therefore appears to come from a small number of locally improved pricing decisions that matter for agreement and margin.

Overall, the TTA result is best read as an extension-side inference refinement. It shows that a lightweight agent-side adaptation layer can improve decision quality under the same benchmark setting while leaving the underlying benchmark definition unchanged.

6.6 Reward-design analysis

6.6.1 PPO reward-design ablation

We also conduct a reward-design ablation for the PPO family under the CLIP-enabled setting. In this study, the PPO optimization setup is kept fixed while only the reward profile is changed, so the resulting differences can be interpreted as sensitivity to reward design rather than to broader training changes. The baseline in this comparison is the profit-only PPO setting, while *v1–v3* denote three alternative shaped reward profiles built on the same CLIP-enabled PPO setup.

Across these PPO variants, the same shaped reward template is used, while only the

parameter values change:

$$r_t^{\mathrm{PPO}} = \begin{cases} \dfrac{\mathrm{profit}_t}{1000} + \mathbb{I}[\mathrm{profit}_t < \tau] \cdot \lambda_{\mathrm{low}}, & \text{if the episode terminates with a deal,} \\ \lambda_{\mathrm{nodeal}}, & \text{if the episode terminates without a deal and the no-deal condition is active,} \\ \lambda_{\mathrm{invalid}}, & \text{if an invalid accept occurs,} \\ \lambda_{\mathrm{step}}, & \text{if a step no-deal penalty is active,} \\ 0, & \text{otherwise,} \end{cases}$$

where τ is the profit target, λ_{low} is the low-profit penalty, $\lambda_{\mathrm{nodeal}}$ is the terminal no-deal penalty, $\lambda_{\mathrm{invalid}}$ is the invalid-accept penalty, and λ_{step} is the optional step no-deal penalty applied on intermediate non-terminal steps. Table 6.5 then specifies how these parameters are instantiated in the profit-only baseline and in the three shaped variants.

Table 6.5: Reward settings for the PPO reward-design ablation. The baseline is the profit-only PPO setting, and *v1–v3* are shaped reward profiles under the same CLIP-enabled PPO setup.

Variant	No-deal pen.	Target (USD)	Low-profit pen.	Pos. margin req.	Invalid accept	Step no-deal pen.	Start round
Profit-Only	0.00	0	0.00	true	0.00	0.00	–
v1	-0.30	12000	-0.20	false	0.00	-0.30	3
v2	-0.35	12000	-0.15	false	0.00	-0.35	3
v3	-0.25	12000	-0.08	false	0.00	-0.25	3

Table 6.6: Results of the PPO reward-design ablation on the shared bank50k test protocol. The baseline corresponds to the profit-only PPO setting, while *v1–v3* denote alternative shaped reward profiles.

Variant	Avg Profit (USD) ↑	Deal Rate ↑	Walkaway ↓	Rounds
Profit-Only	11972.501	0.8212	0.1788	2.3572
v1	**12244.415** (+271.914)	**0.8428** (+0.0216)	**0.1572** (-0.0216)	2.1574 (-0.1998)
v2	12197.840 (+225.339)	0.8406 (+0.0194)	0.1594 (-0.0194)	**2.1254** (-0.2318)
v3	12002.194 (+29.693)	0.8238 (+0.0026)	0.1762 (-0.0026)	2.3286 (-0.0286)

The reward fields in Table 6.5 shape PPO's learning signal in different ways. The *no-deal penalty* discourages unsuccessful terminal outcomes. The *profit target* and *low-profit penalty* determine whether a completed deal is rewarded as sufficiently profitable or

penalized as too weak. The *positive-margin requirement* controls whether no-deal penalties are applied only when a positive-margin agreement was still observably feasible. The *invalid-accept penalty* discourages accepting without a valid buyer counter-offer, while the *step no-deal penalty* and its *start round* add delay-related pressure once bargaining continues into later rounds.

Against this background, the *profit-only* baseline is the cleanest task-aligned variant: it removes all additional shaping and leaves PPO to learn only from terminal profit. By contrast, *v1* and *v2* both add explicit no-deal and low-profit penalties together with round-dependent delay pressure, making the learning signal more strongly agreement-oriented than the baseline. Between them, *v2* applies the strongest no-deal penalty, whereas *v1* combines slightly milder no-deal pressure with a stronger low-profit penalty. *v3* is the mildest shaped profile, retaining the same overall structure but reducing the penalty magnitudes.

The result table shows that PPO is sensitive to reward design even when the rest of the training setup is held constant. Among the tested profiles, reward v1 achieves the highest average profit, with reward v2 remaining close behind and reward v3 still improving slightly over the profit-only baseline. The gains are therefore real, but they are not monotonic in penalty size: the most aggressive profile is not the best-performing one, and the mildest profile yields only a small improvement. This is consistent with PPO's role as a model-free learner whose behavior depends directly on the reward signal used during training.

The pattern suggests that some forms of shaping are better aligned with the task than others. The pure-profit baseline remains the cleanest task-level definition, but moderate shaping helps PPO by discouraging clearly unproductive no-deal outcomes and low-quality agreements. At the same time, the benchmark does not reward arbitrary extra penalties. In this sweep, shaping works best when it stays close to the underlying seller objective while adding just enough structure to help the policy distinguish profitable resolution from unproductive bargaining.

6.6.2 Dreamer reward-design ablation

We also examine Dreamer reward sensitivity under the CLIP-enabled setting. Here the base comparison uses the same pure-profit objective described earlier, while the ablation introduces two alternative Dreamer-side reward variants: one with a late step no-deal penalty and one with a soft shortfall penalty on low-profit deals.

Across these Dreamer variants, the same profit-oriented reward template is retained,

while the shaping terms are changed:

$$r_t^{\text{Dreamer}} = \begin{cases} \dfrac{\text{profit}_t}{1000} + \mathbb{I}[\text{profit}_t < \tau] \cdot \lambda_{\text{low}} - \beta \cdot \dfrac{\max(0, \tau - \text{profit}_t)}{1000}, & \text{(i)}, \\ \lambda_{\text{nodeal}}, & \text{(ii)}, \\ \lambda_{\text{step}} + \lambda_{\text{invalid}}, & \text{(iii)}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, case (i) applies when the episode ends with a deal, case (ii) when the episode ends without a deal and the no-deal penalty condition is active, and case (iii) on non-terminal steps where a late step penalty or an invalid-accept penalty is active. The remaining case corresponds to zero reward. In the notation above, τ is the profit target, λ_{low} is a discrete penalty for deals that fall below that target, and β is the coefficient of the continuous soft-shortfall penalty, which grows as realized profit falls further below the target. The remaining terms are λ_{nodeal} for unsuccessful terminal outcomes, λ_{step} for negotiations that continue into later rounds, and λ_{invalid} for invalid accept actions. Table 6.7 then shows how these terms are instantiated in the pure-profit baseline and in the two shaped variants.

Table 6.7: Reward settings for the Dreamer reward-design ablation. All variants use CLIP-enabled observations and no TTA.

Variant	No-deal pen.	Target (USD)	Low-profit pen.	Soft shortfall coeff.	Step no-deal pen.	Start round	Pos. margin req.	Invalid accept
Pure profit	0.00	0	0.00	0.00	0.00	–	true	0.00
Late penalty	-0.10	0	0.00	0.00	-0.02	4	false	0.00
Soft shortfall	-0.10	12000	0.00	0.10	0.00	4	false	0.00

Table 6.8: Results of the Dreamer reward-design ablation on the shared bank50k test protocol. The baseline corresponds to the pure-profit Dreamer + CLIP setting.

Variant	Avg Profit (USD) ↑	Deal Rate ↑	Walkaway ↓	Rounds
Pure profit	12286.369	0.8664	0.1336	1.6010
Late penalty	9705.080 (-2581.289)	0.6690 (-0.1974)	0.3310 (+0.1974)	2.5980 (+0.9970)
Soft shortfall	**12616.028** (+329.659)	**0.9106** (+0.0442)	**0.0894** (-0.0442)	**1.2864** (-0.3146)

The contrast between these two variants is informative. The late-penalty variant underperforms the pure-profit baseline by a large margin, with a lower deal rate, a higher walk-away rate, and substantially longer negotiations. By contrast, the soft-shortfall variant improves on the pure-profit baseline across all reported benchmark metrics.

The three profiles encourage materially different behaviour. Under the pure-profit setting, Dreamer is rewarded only through realized deal profit, so unsuccessful episodes

and still-open negotiations contribute little training structure beyond eventual outcome. This remains closely aligned with the task objective, but it does not necessarily produce the strongest overall benchmark result, because the agent may still fail to convert enough potentially profitable negotiations into agreements.

The late-penalty variant performs much worse. One reasonable interpretation is that repeated penalties on still-open negotiations make the imagined continuation value of bargaining too pessimistic, so the learned model becomes less effective at exploiting recoverable trajectories. By contrast, the soft-shortfall variant preserves the profit-oriented objective while adding a smoother preference against low-profit deals. This appears to help Dreamer distinguish more useful trajectories without heavily distorting the episode dynamics, leading to higher deal rate, lower walk-away rate, fewer rounds, and the best average profit among the tested variants.

These results suggest that Dreamer is sensitive not only to the presence of shaping, but also to the form in which that shaping is introduced. For this benchmark, a smooth terminal preference appears more compatible with Dreamer's latent-dynamics learning than repeated intermediate penalties on ongoing negotiations. More broadly, the reward-design study shows that the benchmark is rich enough to expose meaningful differences between reward formulations even when the backbone architecture is fixed. It also suggests that pure-profit alignment alone does not guarantee the best learning outcome for a world-model agent; the way profit-oriented preferences are expressed in the learning signal also matters.

6.7 Limitations

The current benchmark and experiments come with clear limitations. First, the study uses fixed thesis budgets and retained settings for PPO and Dreamer, rather than a family-wise search for the strongest possible configuration of each method. The reward ablations should be read in the same way: they show that the benchmark is sensitive to learning-signal design, but they do not exhaust the full space of reasonable reward formulations. The value of these experiments is that they demonstrate a benchmark setting rich enough to support further method development and comparison.

The scope of the evidence is also bounded by the simulator itself. The benchmark is built from one fixed catalog, one frozen persona-bank construction, and one simulator-based negotiation environment grounded in accessible public data, so the results should be interpreted within this benchmark setting rather than as broad market generalization. Likewise, the extension studies are indicative rather than exhaustive: CLIP, TTA, and reward shaping are included to show that the benchmark responds to representation quality, inference-time refinement, and reward design, not to cover every promising agent improvement. The simulator is also not real transaction data. Its role is to turn personalized negotiation-based pricing into a structured and controllable research problem that can be

studied under reproducible conditions. These limitations point naturally to future work on stronger agent design, broader simulator scope, and more systematic evaluation settings.

Chapter 7

Conclusion

Summary of the thesis. This thesis formulates personalized pricing for customized car features as a structured POMDP benchmark with multi-round negotiation. It develops a consumer simulator grounded in public U.S. demographic and travel data, instantiates a fixed vehicle-customization setting based on a real car catalog, and connects these components to a shared negotiation environment in which heuristic and learning-based pricing agents can be evaluated under the same protocol. In this way, the thesis turns a broad and loosely defined pricing problem into a reproducible research setting for studying consumer heterogeneity, bargaining, and pricing-agent behaviour together.

Main findings. The experiments show that this benchmark is meaningful and non-trivial. Profit and agreement do not move together in a simple way in this setting, and the agents do not yet make the most of that trade-off. The concession heuristic, despite its simplicity, remains the strongest method on average profit, which indicates that the benchmark is still challenging rather than already solved by current learned agents. At the same time, the learned-agent studies show that representation quality and reward design matter materially: CLIP-based semantic augmentation changes results substantially, especially for Dreamer, and reward shaping can alter learned behaviour in both the PPO and Dreamer families. Taken together, these findings suggest that the benchmark is strong enough to distinguish not only between agent classes, but also between different forms of observation design, inference-time refinement, and learning-signal construction.

Critical appraisal. This thesis has both clear strengths and clear weaknesses. On the positive side, the benchmark setting works well as a research framework: it is reproducible, structured, and challenging enough to produce meaningful differences between heuristic and learning-based methods. The consumer simulator also succeeds in one of its main intended roles, namely to introduce heterogeneous buyers with different observable profiles, latent preferences, willingness-to-pay structure, and bargaining behaviour. The multi-round negotiation environment is another successful part of the project, because it turns pricing into a sequential interaction problem rather than a one-shot quotation problem. In addition, the wrapper-based design makes the benchmark easier to reuse and

extend in later work.

At the same time, several important weaknesses remain. Compared with an ideal personalized-pricing benchmark, the current simulator is still a deliberately narrowed research setting. The consumer side is far from a full representation of a real vehicle market, and the current willingness-to-pay and utility design remains a simplified behavioural model rather than a comprehensive economic one. The benchmark also studies price adjustment over a fixed bundle rather than full personalized bundle construction, which makes the task more controlled but also less expressive than the broader pricing problem that originally motivated the thesis. The current pricing agents are similarly limited in how effectively they use the interaction horizon. Although the task allows up to five rounds, the learned agents do not yet consistently show the stronger behaviour one would hope for, such as using the early rounds to probe the buyer and later rounds to exploit the information gained. The empirical study is also limited in agent breadth, since it does not yet cover broader policy families or dialogue-based agents such as current LLM systems.

If this project were started again, the first change would be to invest earlier in stronger evaluation design and broader agent coverage. In particular, it would have been valuable to build richer diagnostic tools, more systematic training-dynamics analysis, and a wider range of benchmarked agent families earlier in the project, rather than adding them only after the main environment and simulator were already established. A second improvement would be to design the consumer side with a clearer path from public-data grounding to richer behavioural realism, so that later extensions to bundle construction, negotiation style, and buyer-side utility could be added more naturally. More generally, the project would have benefited from making these broader evaluation and simulator-expansion priorities explicit earlier, rather than reaching them only after the core benchmark had already been implemented.

Future work. Future work follows directly from these limitations. On the method side, stronger agents should make better use of the interaction horizon, latent-state inference, and richer representations, including broader comparisons across model-free, model-based, dialogue-based, and later LLM-based policies. On the simulator side, the consumer model could move toward richer behavioural realism, broader catalog scope, and eventually joint bundle-construction and pricing decisions. On the evaluation side, later work should add broader robustness checks, richer customer-segment analysis, and more systematic studies of training dynamics, reward sensitivity, and extension behaviour. The main outcome of this thesis is therefore a benchmark foundation for further study.

Bibliography

[1] Young-Geun Choi, Gi-Soo Kim, Yunseo Choi, Wooseong Cho, Myunghee Cho Paik, and Min-Hwan Oh. Semi-Parametric Contextual Pricing Algorithm using Cox Proportional Hazards Model. In *Proceedings of the 40th International Conference on Machine Learning*, pages 5771–5786. PMLR, July 2023.

[2] Venktesh Pandey, Evana Wang, and Stephen D. Boyles. Deep Reinforcement Learning Algorithm for Dynamic Pricing of Express Lanes with Multiple Access Locations. *Transportation Research Part C: Emerging Technologies*, 119:102715, October 2020. ISSN 0968090X. doi: 10.1016/j.trc.2020.102715.

[3] Jiaxi Liu, Yidong Zhang, Xiaoqing Wang, Yuming Deng, and Xingyu Wu. Dynamic Pricing on E-commerce Platform with Deep Reinforcement Learning: A Field Experiment. Technical Report arXiv:1912.02572, arXiv, August 2021.

[4] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An Integrated Environment for Supporting the Design of Generic Automated Negotiators. *Computational Intelligence*, 30(1):48–70, 2014. ISSN 1467-8640. doi: 10.1111/j.1467-8640.2012.00463.x.

[5] Yasser Mohammad, Shinji Nakadai, and Amy Greenwald. NegMAS: A Platform for Automated Negotiations. In Takahiro Uchiya, Quan Bai, and Iván Marsá Maestre, editors, *PRIMA 2020: Principles and Practice of Multi-Agent Systems*, volume 12568, pages 343–351. Springer International Publishing, Cham, 2021. ISBN 978-3-030-69321-3 978-3-030-69322-0. doi: 10.1007/978-3-030-69322-0_23.

[6] Øyvind Thomassen. An Empirical Model of Automobile Engine Variant Pricing. *International Journal of the Economics of Business*, 24(3):275–293, September 2017. ISSN 1357-1516. doi: 10.1080/13571516.2017.1333733.

[7] Yana Wang, Zhen-Song Chen, and Xian-Jia Wang. Assortment planning and pricing for configurable product under sequential choice process. *Management System Engineering*, 1(1):6, October 2022. ISSN 2731-5843. doi: 10.1007/s44176-022-00002-3.

[8] Pangpang Liu, Zhuoran Yang, Zhaoran Wang, and Will Wei Sun. Contextual dynamic pricing with strategic buyers. *Journal of the American Statistical Association*, 120(550):896–908, 2025.

[9] Enrique Adrian Villarrubia-Martin, Luis Rodriguez-Benitez, David Muñoz-Valero,

Giovanni Montana, and Luis Jimenez-Linares. Dynamic Pricing in High-Speed Railways Using Multi-Agent Reinforcement Learning. Technical Report arXiv:2501.08234, arXiv, September 2025.

[10] Max Biggs, Wei Sun, and Markus Ettl. Model distillation for revenue optimization: Interpretable personalized pricing. In *International Conference on Machine Learning*, pages 946–956. PMLR, 2021.

[11] Jean-Pierre Dubé and Sanjog Misra. Personalized pricing and consumer welfare. *Journal of Political Economy*, 131(1):131–189, 2023.

[12] Anna Priester, Thomas Robbert, and Stefan Roth. A special price just for you: Effects of personalized dynamic pricing on consumer fairness perceptions. *Journal of Revenue and Pricing Management*, 19(2):99–112, April 2020. ISSN 1477-657X. doi: 10.1057/s41272-019-00224-3.

[13] Jungwoo Shin, Chandra R. Bhat, Daehyun You, Venu M. Garikapati, and Ram M. Pendyala. Consumer preferences and willingness to pay for advanced vehicle technology options and fuel types. *Transportation Research Part C: Emerging Technologies*, 60:511–524, November 2015. ISSN 0968090X. doi: 10.1016/j.trc.2015.10.003.

[14] Michael Schlechtinger, Damaris Kosack, Franz Krause, and Heiko Paulheim. By Fair Means or Foul: Quantifying Collusion in a Market Simulation with Deep Reinforcement Learning. Technical Report arXiv:2406.02650, arXiv, June 2024.

[15] Yu Xia, Ali Arian, Sriram Narayanamoorthy, and Joshua Mabry. RetailSynth: Synthetic Data Generation for Retail AI Systems Evaluation. Technical Report arXiv:2312.14095, arXiv, December 2023.

[16] Paulo Salem, Robert Sim, Christopher Olsen, Prerit Saxena, Rafael Barcelos, and Yi Ding. TinyTroupe: An LLM-powered Multiagent Persona Simulation Toolkit. Technical Report arXiv:2507.09788, arXiv, July 2025.

[17] Xiao-Yang Liu. FinRL-Meta: Market Environments and Benchmarks for Data-Driven Financial Reinforcement Learning. *SSRN Electronic Journal*, 2022. ISSN 1556-5068. doi: 10.2139/ssrn.4253139.

[18] Tim Baarslag, Koen Hindriks, Catholijn Jonker, Sarit Kraus, and Raz Lin. The First Automated Negotiating Agents Competition (ANAC 2010). In Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo, editors, *New Trends in Agent-Based Complex Automated Negotiations*, pages 113–135. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-24696-8. doi: 10.1007/978-3-642-24696-8_7.

[19] Mehmet Onur Keskin, Berk Buzcu, Berkecan Koçyiğit, Umut Çakan, Anıl Doğru, and Reyhan Aydoğan. Negotiator: A comprehensive framework for human-agent negotiation integrating preferences, interaction, and emotion. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 8700–8703, 2024.

[20] Chunkit Chan, Jiayang Cheng, Yauwai Yim, Zheye Deng, Wei Fan, Haoran Li, Xin Liu, Hongming Zhang, Weiqi Wang, and Yangqiu Song. Negotiationtom: A benchmark for stress-testing machine theory of mind on negotiation surrounding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4211–4241, 2024.

[21] Tian Xia, Zhiwei He, Tong Ren, Yibo Miao, Zhuosheng Zhang, Yang Yang, and Rui Wang. Measuring bargaining abilities of llms: A benchmark and a buyer-enhancement method. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3579–3602, 2024.

[22] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially Observable Markov Decision Processes in Robotics: A Survey. *IEEE Transactions on Robotics*, 39(1):21–40, February 2023. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2022.3200138.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. Technical Report arXiv:1707.06347, arXiv, August 2017.

[24] Raad Khraishi and Ramin Okhrati. Offline deep reinforcement learning for dynamic pricing of consumer credit. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 325–333, 2022.

[25] David Ha and Jürgen Schmidhuber. World Models. March 2018. doi: 10.5281/zenodo.1207631.

[26] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. Technical Report arXiv:2301.04104, arXiv, April 2024.

[27] Build Your Own 2026 E 350 Sedan. https://www.mbusa.com/en/vehicles/build/e-class/sedan/e350w.

[28] Census profile: United States. http://censusreporter.org/profiles/01000US-united-states/.

[29] US Census Bureau. Income in the United States: 2023. https://www.census.gov/library/publications/2024/demo/p60-282.html.

[30] Stacey Bricka, Timothy Reuscher, Paul Schroeder, Mitchell Fisher, Justina Beard, and Xiaoyuan Layla Sun. Summary of travel trends: 2022 national household travel survey. 2024.

[31] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. Technical Report arXiv:2407.17032, arXiv, November 2025.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al.

Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[33] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uXl3bZLkr3c`.

Appendix A

Full E350 customization catalog

This appendix provides the complete option-level catalog used by the benchmark. It lists the benchmark option identifiers, representative option descriptions, consumer-facing MSRP deltas, and the scalar aesthetic prior weights assigned to each option in the base simulator.

Table A.1: Full E350 customization catalog used in the benchmark. MSRP deltas are compiled from official Mercedes-Benz pricing materials, and the aesthetic prior is the scalar appearance-oriented weight used by the base simulator.

Dimension	Option ID	Representative option(s)	MSRP (USD)	Aesthetic prior
paint color	paint_standard	Standard black or Polar White paint	0	0.20
paint color	paint_metallic	Metallic paint finish	750	0.45
paint color	paint_manufaktur	MANUFAKTUR paint finish	1750	0.80
wheels	wheel_18_standard	18-inch standard wheel set	0	0.20
wheels	wheel_19_upgrade	19-inch wheel upgrade	600	0.50
wheels	wheel_amg_high	AMG 20/21-inch wheel upgrade	1950	0.85
exterior style	styling_upgrade	Night Package or illuminated grille styling upgrade	400	0.55
upholstery	mb_tex	MB-Tex upholstery	0	0.25
upholstery	leather	Leather upholstery	1620	0.65
upholstery	nappa_leather	Nappa leather upholstery	2990	0.90
trim	standard_trim	Standard wood or piano-black trim	0	0.25
trim	premium_trim	Premium wood, metallic, or star-pattern trim	150	0.55
comfort	multicontour_package	Multicontour Seating Package	2950	0.85
comfort	seat_comfort_upgrade	Ventilated front seats or heated rear seats	500	0.45
comfort	soft_close_doors	Soft-close doors	550	0.40
audio	burmester_4d	Burmester 4D surround-sound system	1030	0.70
technology	mbux_superscreen	MBUX Superscreen Package	1500	0.90
safety	driver_assistance_package	Driver Assistance Package	1950	0.60
performance	airmatic_package	AIRMATIC Package	3200	0.65
lighting	digital_light	DIGITAL LIGHT headlamps	990	0.60

Appendix B

Persona priors and conditional distributions

This appendix documents the probability tables and generation rules used by the persona schema. The observable profile is grounded in public U.S. buyer priors and lightweight conditional approximations, while the hidden layer is derived through conditional mappings, numeric mixtures, and bounded coupling rules. These tables are provided to make the simulator assumptions transparent and auditable without overloading the main text.

B.1 Observable priors

Table B.1: Observable profile priors used by the consumer simulator.

Field	Values	Probability vector
Age band	18–25, 26–35, 36–50, 50+	[0.08, 0.24, 0.39, 0.29]
Income band	<60k, 60–100k, 100–180k, 180k+	[0.08, 0.22, 0.40, 0.30]
Household stage	single, couple, family	[0.27, 0.31, 0.42]
Ownership stage	first-time, replacement, additional	[0.15, 0.68, 0.17]
Primary use case	commute, family, luxury, performance, mixed	[0.19, 0.22, 0.14, 0.10, 0.35]

B.2 Observable conditional approximations

Table B.2: $P(\text{income band} \mid \text{age band})$ used in observable-profile sampling.

Age band	<60k	60–100k	100–180k	180k+
18–25	0.42	0.36	0.17	0.05
26–35	0.14	0.33	0.34	0.19
36–50	0.05	0.20	0.43	0.32
50+	0.06	0.18	0.40	0.36

Table B.3: $P(\text{household stage} \mid \text{age band})$ used in observable-profile sampling.

Age band	single	couple	family
18–25	0.56	0.25	0.19
26–35	0.32	0.31	0.37
36–50	0.17	0.29	0.54
50+	0.34	0.46	0.20

Table B.4: $P(\text{ownership stage} \mid \text{age band})$ used in observable-profile sampling.

Age band	first-time	replacement	additional
18–25	0.52	0.42	0.06
26–35	0.21	0.65	0.14
36–50	0.08	0.70	0.22
50+	0.03	0.72	0.25

Table B.5: $P(\text{primary use case} \mid \text{household stage})$ used in observable-profile sampling.

Household stage	commute	family	luxury	performance	mixed
single	0.30	0.08	0.17	0.10	0.35
couple	0.20	0.14	0.18	0.11	0.37
family	0.12	0.43	0.08	0.06	0.31

B.3 Hidden conditional mappings

Table B.6: $P(\text{decision style} \mid \text{primary use case})$ used in hidden-profile generation.

Primary use case	analytic	balanced	expressive
commute	0.46	0.44	0.10
family	0.40	0.50	0.10
luxury	0.24	0.46	0.30
performance	0.26	0.34	0.40
mixed	0.31	0.44	0.25

Table B.7: $P(\text{tech-affinity band} \mid \text{age band})$ used in hidden-profile generation.

Age band	low	medium	high
18–25	0.10	0.36	0.54
26–35	0.12	0.43	0.45
36–50	0.19	0.51	0.30
50+	0.33	0.50	0.17

Table B.8: $P(\text{top two priorities} \mid \text{primary use case})$ used in hidden-profile generation.

Primary use case	Priority pair	Probability
commute	(price, comfort)	0.30
commute	(comfort, safety)	0.24
commute	(tech, comfort)	0.18
commute	(safety, tech)	0.18
commute	(aesthetics, comfort)	0.10
family	(comfort, safety)	0.35
family	(safety, tech)	0.32
family	(price, comfort)	0.23
family	(tech, comfort)	0.10
luxury	(aesthetics, comfort)	0.34
luxury	(tech, comfort)	0.30
luxury	(performance, aesthetics)	0.24
luxury	(comfort, safety)	0.12
performance	(performance, aesthetics)	0.50
performance	(tech, comfort)	0.20
performance	(aesthetics, comfort)	0.18
performance	(price, comfort)	0.12
mixed	(price, comfort)	0.20
mixed	(comfort, safety)	0.22
mixed	(performance, aesthetics)	0.13
mixed	(tech, comfort)	0.17
mixed	(safety, tech)	0.16
mixed	(aesthetics, comfort)	0.12

B.4 Hidden numeric mixtures and linked rules

Table B.9: Numeric mixture distributions used in hidden-profile generation.

Variable	Support	Probability vector
Price sensitivity	[0.70, 1.00, 1.35]	[0.28, 0.50, 0.22]
Aesthetic sensitivity	[0.45, 0.75, 1.05]	[0.24, 0.52, 0.24]
Patience	[3, 4, 5, 6]	[0.20, 0.34, 0.30, 0.16]
Counter strength	[0.30, 0.55, 0.80]	[0.30, 0.48, 0.22]
Walk-away threshold	[0.05, 0.10, 0.18]	[0.42, 0.40, 0.18]
Belief obscurity	[0.20, 0.45, 0.70]	[0.30, 0.50, 0.20]
Brand loyalty	[0.30, 0.55, 0.80]	[0.24, 0.52, 0.24]
Impulsivity	[0.20, 0.45, 0.75]	[0.30, 0.48, 0.22]

Table B.10: Income-conditioned reservation-price base distribution.

Income band	Mean (USD)	Std (USD)
<60k	6800	850
60–100k	9200	1100
100–180k	12800	1400
180k+	17200	1700

Table B.11: Conditional shift rules applied after initial hidden-variable sampling.

Condition source	Value	Shift applied
Primary use case	luxury	brand loyalty +0.08
Primary use case	performance	price sensitivity -0.08
Ownership stage	first-time	price sensitivity +0.12; brand loyalty -0.10; walk-away threshold +0.08; patience -1
Ownership stage	replacement	brand loyalty +0.06; walk-away threshold -0.03; patience +1
Ownership stage	additional	price sensitivity -0.05; aesthetic sensitivity +0.08; brand loyalty +0.04
Tech-affinity band	high	brand loyalty +0.04
Priority set contains	price	price sensitivity +0.12

Table B.12: Coupling rules used to keep hidden-variable combinations behaviorally coherent.

Coupling	**Rule**
Reservation price from price sensitivity	multiplicative adjustment with base 1.08, slope -0.18, clipped to [0.75, 1.20]
Walk-away threshold from price sensitivity	additive adjustment with slope 0.10 around center 1.00
Walk-away threshold from patience	additive adjustment with slope -0.04 around center 5
Counter strength from belief obscurity	additive adjustment with slope 0.15 around center 0.50

Appendix C

User manual

This user manual explains how to install the benchmark package, run the main command-line workflow, and inspect the generated outputs. It is intended for a user who wants to execute the program and check its performance, not for a maintainer who wants to modify the implementation. Implementation structure and extension details are described separately in Appendix D.

C.1 Requirements

The benchmark is designed to run from the repository root in a Unix-like command-line environment. The full learned-agent workflow assumes the following software and hardware:

- Linux;
- Conda or Miniconda;
- Python 3.10;
- an NVIDIA GPU with a CUDA 12.x-compatible driver/runtime for the full PPO, Dreamer, CLIP, and JAX-based workflow.

Some lightweight commands, such as checking script help messages or inspecting existing outputs, can be run without completing the full training workflow. Full reproduction of the reported learned-agent experiments is computationally more demanding.

C.2 Installation

Open a terminal in the repository root. Run the command blocks in the order shown. If a command line ends with a backslash, copy the whole multi-line block as one command; otherwise, the lines can be run one by one.

```
conda create -n pricing-agent python=3.10 -y
conda activate pricing-agent
```

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

The `requirements.txt` installation is the main installation path for the released package. It is expected to install the retained PPO, DreamerV3, CLIP, and JAX dependencies needed by the benchmark. After installation, the user should verify that the intended GPU-backed PyTorch and JAX setup is actually visible at runtime.

Check that PyTorch is installed and can see the GPU:

```
python -c "import torch"
python -c "import torch; print(torch.cuda.is_available())"
```

The first command should finish without an import error. The second command should print `True`; if it prints `False`, the CUDA-enabled PyTorch installation is not being used.

Check that DreamerV3 and JAX import correctly:

```
python -c "import dreamerv3, embodied"
python -c "import jax; print(jax.devices())"
```

The final command should list a CUDA device. If it lists only a CPU device, the Dreamer runs will not use the intended GPU backend.

Figure C.1 shows an example of the expected terminal output after these PyTorch, DreamerV3, and JAX checks have been run successfully.

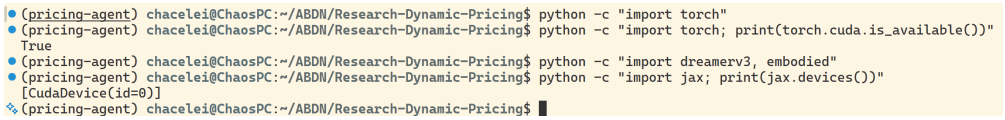

Figure C.1: Example terminal output after the PyTorch, DreamerV3, and JAX environment checks.

C.3 Benchmark data

The workflow expects the fixed persona bank and CLIP semantic assets under:

- `datasets/persona_bank/bank50k_s123/`;
- `datasets/clip_semantics/e350_clip_text_v1.json`.

If these files are already present, the user can proceed directly to the experiment commands below. If the persona bank is missing, rebuild it with:

```
python scripts/data/build_persona_bank.py \
  --count 50000 \
```

```
  --seed 123 \
  --output-dir datasets/persona_bank/bank50k_s123 \
  --overwrite
```

If the CLIP semantic file is missing, rebuild it with:

```
python scripts/data/build_clip_semantics.py \
  --catalog-path catalog/e350_core_catalog.yaml \
  --output-path datasets/clip_semantics/e350_clip_text_v1.json \
  --model-name ViT-B-32 \
  --pretrained openai \
  --seed 123 \
  --overwrite
```

C.4 Running experiments

In the delivered package, the benchmark is intended to be run through the Python command-line scripts shown below.

Example PPO workflow:

```
python scripts/agents/train_ppo.py \
  --config-path configs/ppo/ablations/bank50k/clip.yaml \
  --seed 123

python scripts/agents/run_benchmark.py \
  --config-path configs/ppo/ablations/bank50k/clip.yaml \
  --seed 123
```

Example Dreamer workflow:

```
python scripts/agents/train_dreamer.py \
  --config-path configs/dreamer/ablations/bank50k/clip.yaml \
  --seed 123

python scripts/agents/run_benchmark_dreamer.py \
  --config-path configs/dreamer/ablations/bank50k/clip.yaml \
  --seed 123
```

Benchmark-only TTA configurations reuse the Dreamer + CLIP checkpoint produced by the standard Dreamer workflow above and do not require a separate TTA training run. For example:

```
python scripts/agents/eval_dreamer.py \
  --config-path configs/dreamer/ablations/bank50k/clip_tta_v2.yaml \
  --seed 123

python scripts/agents/run_benchmark_dreamer.py \
  --config-path configs/dreamer/ablations/bank50k/clip_tta_v2.yaml \
  --seed 123
```

Further command examples are collected in the thesis command summary:

`scripts/commands/thesis_commands.md`

C.5 Expected outputs

The easiest way to understand the outputs is to follow them step by step. Different commands generate different kinds of files, and the user does not need to inspect every directory in the package in order to confirm success.

After rebuilding the benchmark data. If the user runs the persona-bank builder, the directory `datasets/persona_bank/bank50k_s123/` should contain files such as `train.jsonl`, `val.jsonl`, `test.jsonl`, `persona_bank.jsonl`, and `manifest.json`. If the user runs the CLIP semantic builder, the file `datasets/clip_semantics/e350_clip_text_v1.json` should be created or refreshed. These files are the main fixed data assets used by the benchmark.

After a PPO benchmark run. The PPO commands write their outputs under the configured `artifacts/ppo/...` directory. The most important result for an end user is the benchmark report JSON stored under a `reports/` subdirectory. This report contains the aggregate benchmark metrics for the requested PPO workflow.

After a Dreamer benchmark run. The Dreamer commands write their outputs under the configured `artifacts/dreamer/...` directory. As with PPO, the main user-facing result is the benchmark report JSON written under a `reports/` subdirectory. This file records the aggregate benchmark metrics for the selected Dreamer configuration.

How to tell that the run succeeded. In normal use, the user does not need to inspect every intermediate file. A practical success check is simply:

- after data preparation, `manifest.json` exists in the expected persona-bank directory;
- after PPO benchmarking, a benchmark report JSON appears under the corresponding `artifacts/ppo/.../reports/` directory;
- after Dreamer benchmarking, a benchmark report JSON appears under the corresponding `artifacts/dreamer/.../reports/` directory;

- the terminal command finishes without missing persona-bank, checkpoint, or import errors.

In other words, the main outputs that most users need to confirm are the fixed benchmark data files and the final benchmark report JSON files. The surrounding artifact directories may also contain checkpoints, logs, and auxiliary metadata, but those are supporting files rather than the first outputs that a beginner needs to inspect.

Figure C.2 shows an example of the terminal-side confirmation that a benchmark report file has been written under the expected artifact directory.

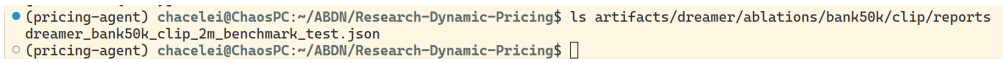

Figure C.2: Example terminal output showing a generated benchmark report file under the artifact directory.

C.6 Troubleshooting

If a command fails immediately, first check that it is being run from the repository root and that the `pricing-agent` environment is active. If the persona bank is missing, rerun the persona-bank builder directly:

```
python scripts/data/build_persona_bank.py \
  --count 50000 \
  --seed 123 \
  --output-dir datasets/persona_bank/bank50k_s123 \
  --overwrite
```

If a benchmark stage cannot find a model checkpoint, run the corresponding training stage first and confirm that the configuration path matches the intended artifact directory. If JAX reports only a CPU device, check the JAX installation and device visibility again before running Dreamer experiments. If CLIP semantic generation fails, confirm that PyTorch can see the GPU and that `datasets/clip_semantics/` can be written.

Appendix D

Maintenance manual

This appendix documents the implementation-facing information needed to inspect, modify, extend, and debug the benchmark codebase. Unlike the user manual in Appendix C, which focuses on running the program, this maintenance manual explains how the package is organized, where the main runtime components are located, which configuration files control the benchmark, and where generated data and reports are stored. The directory and path descriptions below are written against the delivered package structure rather than against a larger local development workspace. Changes to persona generation, reward profiles, evaluation splits, or agent-facing observations can change the benchmark contract and should therefore be tracked explicitly.

D.1 Installation and build procedure

The project is a Python research codebase and does not require a separate compilation step. In practice, "build" in this package means preparing the Python environment, verifying the main entry points, and confirming that the retained benchmark data can be loaded from the package root. In the delivered submission package, the concrete installation and first-run setup steps are given in `README.md`. A maintainer should therefore follow the same installation sequence as in Appendix C, but treat it as a stepwise validation workflow rather than only as a user-facing setup routine.

The recommended order is: create and activate the `pricing-agent` Python 3.10 environment; install the retained dependency set from `requirements.txt`; run the import and device checks from Appendix C; run lightweight script-entry checks; run the included smoke test; and only then move on to data regeneration, training, or benchmarking. The first script-entry checks should confirm that the main commands resolve correctly from the package root:

```
python scripts/data/build_persona_bank.py --help
python scripts/agents/run_benchmark.py --help
python scripts/agents/run_benchmark_dreamer.py --help
```

The next check should be the included smoke script:

```
python scripts/dev/run_env_smoke.py
```

This script instantiates the environment, runs a short random-policy loop, and prints summary metrics. Successful installation therefore means more than "the imports did not fail": the import checks should pass, JAX should expose the intended CUDA device when Dreamer support is required, the three `-help` commands should resolve from the package root, and the smoke script should complete without path or environment errors. After that, the maintainer should verify that the retained persona bank exists under `datasets/persona_bank/bank50k_s123/`; if it does not, the `prepare` stage from the helper script should be rerun before any longer PPO or Dreamer workflow is attempted.

D.2 Hardware, software, and resource requirements

The runtime environment used for the workflow is Linux with Python 3.10 under Conda or Miniconda. A CUDA 12.x-compatible NVIDIA GPU is expected for the full experiment pipeline, because the learned-agent workflow uses GPU-backed PyTorch and JAX components. CPU-only execution is still useful for lightweight maintenance work such as inspecting files, checking configuration paths, running `-help` commands, and executing the smoke script, but it is not the intended setup for reproducing the full PPO, Dreamer, CLIP, and TTA workflow.

The standard Python dependencies are declared in `requirements.txt`. Maintainers should treat that file as the primary dependency reference for the retained release package, and then use the PyTorch and JAX device checks from Appendix C to confirm that the intended GPU-backed runtime is actually visible. In other words, the maintenance workflow depends on the same environment as the user workflow, but with the additional expectation that the maintainer may need to rerun preprocessing steps, inspect generated reports, or trace a failure back to its script and configuration source.

For routine maintenance, it is reasonable to reserve at least 500 MB of free disk space for the source tree, fixed benchmark data, regenerated semantic assets, and basic report outputs. This is enough for lightweight inspection, smoke checks, path validation, and some data-regeneration tasks. Full learned-agent reruns need substantially more headroom, because PPO and Dreamer can produce checkpoints, logs, and benchmark reports under `artifacts/`. In practice, a maintainer planning to rerun those workflows should expect disk use to grow well beyond the minimum baseline and should treat a CUDA-capable GPU as part of the normal setup rather than as an optional acceleration path.

D.3 Repository and file organization

The runtime package is organized around a small number of top-level directories and root files. At the package root, `README.md` provides the user-facing entry documentation, `requirements.txt` records the standard Python dependency set, and `LICENSE` records

the software license. The remaining contents are grouped by role rather than by experiment run.

- `src/`: the core Python implementation of the environment and pricing-agent modules;
- `scripts/`: runnable entry points for data preparation, training, benchmarking, evaluation, plotting, and small development checks;
- `configs/`: YAML configuration files for persona priors, hidden-trait mappings, PPO settings, Dreamer settings, reward profiles, and retained experiment variants;
- `catalog/`: the fixed E350 vehicle customization catalog used by the benchmark;
- `datasets/`: the fixed persona-bank and CLIP semantic assets included with the package.

Within these directories, the structure remains role-based. The `src/` tree is divided into `src/pricing_env/` for the simulator and environment logic, and `src/pricing_agent/` for heuristic, PPO, and Dreamer-side agent code. The `scripts/` tree is divided into `agents/`, `data/`, `common/`, `plots/`, `commands/`, and `dev/`, so that training and benchmark entry points are kept separate from configuration loaders, plotting utilities, and smoke-style utilities. The `configs/` tree is divided by agent family and experiment type, with separate PPO, Dreamer, reward-profile, and persona-generation files rather than one monolithic configuration file.

The clean source package intentionally omits large generated artifact trees such as checkpoints, logs, and archived benchmark outputs. Those files can be regenerated from the code, configurations, and included benchmark data, but they are not treated as part of the package itself.

D.4 Source file guide

The codebase is easier to maintain if file roles are read in groups rather than as one flat directory listing. The files below are therefore organized by maintenance role, so that a reader can move directly to the environment logic, agent adapters, data-building scripts, or benchmark entry points depending on the part of the system being changed.

Environment implementation

- `src/pricing_env/negotiation_env.py`: main episode-level negotiation loop, including reset/step logic, buyer-response handling, reward signals, and episode trace construction.
- `src/pricing_env/gym_wrapper.py`: Gymnasium-style wrapper that converts the raw negotiation environment into fixed-size observations and standard step signatures for agent training and evaluation.

- `src/pricing_env/persona.py`: persona sampling, persona-bank loading, observable-profile sampling, and hidden-trait mapping.
- `src/pricing_env/wtp.py`: willingness-to-pay decomposition, buyer utility, seller utility, and related pricing-side value calculations.
- `src/pricing_env/negmas_backend.py`: persistent NegMAS bargaining session backend and round-level negotiation response logic.
- `src/pricing_env/catalog.py`: catalog loading, one-option-per-dimension sampling, MSRP aggregation, implementation-cost proxy, and aesthetic proxy utilities.
- `src/pricing_env/types.py`: shared dataclass definitions for catalog options, personas, actions, observations, and episode metrics.

Agent implementation

- `src/pricing_agent/baselines.py`: heuristic baseline policies and baseline-evaluation helpers.
- `src/pricing_agent/ppo_env.py`: PPO-facing environment adapter with discrete price-delta actions and PPO-specific reward shaping.
- `src/pricing_agent/world_model/adapter.py`: Dreamer-facing discrete-action adapter that wraps the PPO-style environment for world-model agents.
- `src/pricing_agent/world_model/dreamer_runtime.py`: DreamerV3 runtime integration, configuration assembly, dependency checks, checkpoint handling, and policy-loading utilities.
- `src/pricing_agent/world_model/tta.py`: inference-time adaptation logic, TTA configuration validation, belief-state updates, and candidate reranking.

Data preparation and shared configuration helpers

- `scripts/data/build_persona_bank.py`: deterministic persona-bank generation, split creation, manifest writing, and preview export.
- `scripts/data/build_clip_semantics.py`: offline text-only CLIP semantic feature construction for catalog options.
- `scripts/common/ppo_config.py`: PPO YAML loading, validation, and repository-relative path resolution.
- `scripts/common/dreamer_config.py`: Dreamer YAML loading, validation, and repository-relative path resolution.

Training, evaluation, and benchmark entry points

- `scripts/agents/train_ppo.py`: PPO training entry point and checkpoint/metadata writer.
- `scripts/agents/eval_ppo.py`: PPO evaluation entry point for one trained checkpoint on one selected split.
- `scripts/agents/run_benchmark.py`: shared-episode benchmark entry point for heuristics and PPO.
- `scripts/agents/train_dreamer.py`: DreamerV3 training entry point and checkpoint/metadata writer.
- `scripts/agents/eval_dreamer.py`: Dreamer evaluation entry point for one trained checkpoint on one selected split.
- `scripts/agents/run_benchmark_dreamer.py`: shared-episode benchmark entry point for heuristics and Dreamer.

D.5 Core implementation components

At the core of the implementation is the **benchmark environment**. One episode is built from a fixed catalog bundle, one sampled persona, a round-level willingness-to-pay calculation, and a persistent NegMAS bargaining session. The environment layer therefore combines four responsibilities that are separated in code but used together at runtime: catalog handling, persona sampling or persona-bank lookup, value and utility computation, and the negotiation backend that advances the bargaining session.

On top of that environment sits the **agent layer**. The heuristic baselines act directly on the wrapper-facing observation and action contract. PPO uses a dedicated environment adapter with discrete price-delta actions and PPO-specific reward shaping. Dreamer uses a separate world-model adapter that flattens the action interface into a Dreamer-compatible discrete policy space. The optional inference-time adaptation module is attached at this layer as an agent-side refinement rather than as part of the benchmark environment itself.

The **script layer** orchestrates these components rather than reimplementing them. Data-building scripts create the fixed persona bank and CLIP semantic asset, training scripts create checkpoints and metadata, evaluation scripts score one trained model on one selected split, benchmark scripts enforce shared-episode comparison across policies, and plotting scripts read saved outputs to generate thesis figures. In this way, the runtime logic stays in `src/`, while `scripts/` remains the entry layer for reproducible workflows.

Finally, the **generated-output layer** records what the earlier components produce. Fixed data assets live under `datasets/`, while benchmark reports, metrics, checkpoints,

and figure outputs are written to artifact directories when those workflows are executed. For maintenance work, this separation matters: changing the simulator or agent code changes how outputs are produced, but the generated files themselves are not the source of truth for the implementation.

D.6 Core data structures and procedures

The implementation is easier to trace if one follows the flow of data through a single episode. A run begins from two fixed inputs: one catalog bundle and one persona record. From these, the environment constructs an observation, receives an action from the policy, advances the bargaining session, updates round-dependent values such as willingness to pay, and records the resulting episode trace and aggregate metrics. Benchmark scripts then repeat this process across many shared episodes and write a report JSON for later analysis.

The first important data object is the **persona record**. In our benchmark, each persona contains observable profile fields, hidden behavioural fields, split membership, source information, and a stable persona identifier. The fixed persona bank stores these records in JSONL form and provides train, validation, and test subsets. The second important object is the **catalog option record**. Each option contains a stable key, a dimension label, an MSRP delta, and an aesthetic-weight field. Episode construction samples one option per retained catalog dimension, so the resulting bundle is already a structured list rather than an untyped string or free-form configuration.

During runtime, these inputs are converted into environment-facing observation and action objects. The raw environment observation contains round index, remaining rounds, recent negotiation history, selected option keys, bundle-level summary values, and the observable part of the persona. The raw action object contains one move token and, when relevant, one offered price in USD. PPO and Dreamer do not consume this raw form directly. PPO uses a discrete move plus price-delta representation, while Dreamer flattens that representation into one discrete action index through its adapter layer. This means the underlying bargaining task remains the same even though the training interfaces differ.

The main output objects are the episode trace, episode metrics, and benchmark report. The episode trace records step-level events such as seller offers, consumer responses, counter-offers, utility values, and terminal causes. Episode metrics compress the completed trajectory into per-episode quantities such as profit, deal reached, rounds used, and walk-away status. Benchmark scripts then aggregate these episode-level outputs across a shared seed-defined episode set and write report JSON files containing policy-level averages, split metadata, seed information, and reproducibility checks such as same-consumer verification.

D.7 Configuration paths and generated files

The maintainership-sensitive constants are concentrated in a small number of YAML files under `configs/`. Observable buyer priors are defined in `configs/us_buyer_distribution_v2.yaml`. Hidden-trait mappings are defined in `configs/persona_hidden_mapping_v1.yaml`. The retained persona schema is defined in `configs/personas_v2.yaml`. PPO-side settings are grouped under `configs/ppo/`, Dreamer-side settings are grouped under `configs/dreamer/`, and reward-specific adjustments are grouped under `configs/reward_profiles/` and the retained reward-sweep subdirectories. The active values are always determined by the specific file passed through `--config-path`, not by the mere existence of a default file elsewhere in the package.

For routine maintenance, the most important benchmark-defining constants come from the selected config files and the direct Python entry points used to build data and run experiments:

- negotiation horizon: `configs/personas_v2.yaml`, key `sampling.max_rounds`;
- retained seed, persona-bank size, and main persona-bank output path: `scripts/data/build_persona_bank.py`, via `--seed123`, `--count50000`, and the pair `--output-dir` plus `datasets/persona_bank/bank50k_s123`;
- active persona-bank file: `paths.persona_bank_path` in the selected PPO or Dreamer config;
- price-bin granularity and price step: `environment.price_bin_count` `environment.price_step_usd`
- initial offer anchor: `environment.initial_offer_markup`
- benchmark episode count and benchmark split: `benchmark.episodes` `benchmark.persona_bank_split`
- reward-shaping values: `environment.no_deal_penalty` `environment.profit_target_usd` `environment.low_profit_penalty` plus Dreamer-specific reward and TTA blocks when those variants are being maintained.

The main data paths are also stable:

- fixed benchmark catalog: `catalog/e350_core_catalog.yaml`;
- retained large persona bank: `datasets/persona_bank/bank50k_s123/`;

- files stored in the retained bank: `train.jsonl`, `val.jsonl`, `test.jsonl`, `persona_bank.jsonl`, and `manifest.json`;

- CLIP semantic asset: `datasets/clip_semantics/e350_clip_text_v1.json`.

In a clean source package, these files are treated as fixed runtime inputs. Benchmark reports, checkpoints, logs, and regenerated figure inputs are treated as generated outputs. Those generated files are written under the artifact paths declared by the selected PPO or Dreamer configuration and can therefore be deleted and regenerated without changing the source tree itself. The package does not depend on any special runtime temporary-file directory beyond normal Python cache files; the main disposable outputs are the generated artifacts under their configured output roots.

D.8 Maintainer workflow and debugging notes

For maintenance work, the recommended validation order is: run the import and device checks from Appendix C; confirm that the main script entry points resolve with `-help`; run `scripts/dev/run_env_smoke.py`; verify that the persona bank and CLIP semantic asset exist; and only then move on to data regeneration, training, or benchmarking. This order keeps environment and path problems separate from longer learned-agent reruns. Maintainers should call the lower-level Python scripts directly when debugging one stage in isolation. In practice, data regeneration is handled by `scripts/data/build_persona_bank.py` and `scripts/data/build_clip_semantics.py`, training is handled by `scripts/agents/train_ppo.py` and `scripts/agents/train_dreamer.py`, and shared-episode benchmarking is handled by `scripts/agents/run_benchmark.py` and `scripts/agents/run_benchmark_dreamer.py`.

The most common failures are path and environment mismatches rather than logic bugs. If a command fails immediately, first confirm that it is being run from the package root and that the intended Conda environment is active. If a benchmark command cannot find the persona bank, rerun the data-preparation stage and recheck `datasets/persona_bank/bank50k_s123/`. If a Dreamer command cannot see the GPU, rerun the JAX device check from Appendix C before changing any code. If a benchmark run cannot find a checkpoint or report directory, inspect the selected config file and confirm that the configured output root matches the retained run being invoked. A second class of maintenance risk comes from contract changes: modifying persona-generation files, reward profiles, observation construction, or evaluation-split settings can silently invalidate cross-run comparability unless the change is tracked explicitly and the affected reports are regenerated.

D.9 Known issues, bug reports, and future maintenance work

The most common **recurring maintenance risks** are straightforward but disruptive: running commands from the wrong working directory, pointing a config file at the wrong persona bank, invoking a benchmark before the expected checkpoint exists, mixing output roots across retained variants, and treating stale generated reports as if they were current outputs. A second recurring risk is configuration mismatch across retained variants. In practice, reward profile, CLIP status, TTA status, checkpoint path, and benchmark report should correspond to the same intended setting; otherwise a command may still run successfully while producing results that no longer match the expected variant being reported. In the retained release package, the benchmark-only TTA path is expected to reuse the checkpoint produced by the standard Dreamer + CLIP training workflow rather than a separately packaged checkpoint artifact. Dreamer-side maintenance has one additional recurring risk, namely a mismatch between the intended GPU-backed JAX installation and the device that JAX actually exposes at runtime. These are the first failure modes to rule out before treating a problem as a simulator or agent bug.

When **reporting a bug**, the minimum useful record is the exact script path, config path, seed, working directory, and the first explicit error message. For benchmark mismatches, the report should also record which persona bank, catalog file, and artifact root were used, because many apparent logic bugs are in fact caused by mixed paths or stale generated outputs. If the failure occurs only for one retained configuration, the report should name that configuration file directly rather than referring to the agent family in general.

The most useful **future maintenance** improvements are practical rather than architectural. The package would benefit from a canonical dependency lock file in addition to `requirements.txt`, a small smoke-test layer that checks the retained data paths and script entry points, and slightly stronger documentation of artifact naming and output-root conventions. It would also benefit from lightweight checks that confirm whether reward profile, CLIP status, TTA status, checkpoint path, and benchmark report all correspond to the same retained variant. A second improvement area is workflow consolidation: PPO-side and Dreamer-side benchmark orchestration still share the same benchmark contract but use separate command entry points, so there is room to reduce repeated path handling and repeated reporting logic while keeping the two training stacks independent.

Appendix E

Extension implementation details

This appendix summarizes the implementation details of the optional CLIP and inference-time adaptation extensions used in the thesis. The goal is reproducibility rather than methodological novelty. Only the settings needed to reconstruct the reported extension results are included here.

E.1 CLIP semantic-feature construction

The CLIP extension is implemented as an offline text-only semantic preprocessing pipeline. No online CLIP inference is performed inside the training or evaluation loop. Instead, each catalog option is assigned one fixed text prompt, encoded once by a pinned CLIP backend, and stored in a reusable semantics artifact. During runtime, the benchmark performs only lookup and aggregation over these precomputed option-level features.

Table E.1: Summary of the CLIP semantic-feature construction used in the benchmark.

Item	**Reported setting**
Artifact path	`datasets/clip_semantics/e350_clip_text_v1.json`
Schema version	`clip_text_semantics_v1`
Backend library	`open-clip-torch`
Model / pretrained pair	`ViT-B-32` with `openai` weights
Tokenizer	`ViT-B-32` tokenizer
Semantic dimension	5
Semantic axes	`aesthetic_luxury`, `aesthetic_sporty`, `aesthetic_modern_tech`, `aesthetic_premium_material`, `aesthetic_visual_impact`
Catalog coverage	20 option-level records, matching the benchmark catalog
Aggregation level	option-level features aggregated into a configuration-level semantic vector
Runtime role	observation-side semantic augmentation for the agent; no change to benchmark dynamics
Legacy scalar proxy	retained alongside CLIP features for compatibility (`legacy_proxy_enabled = true`)

At runtime, the selected option-level CLIP vectors are aggregated into one configuration-level semantic vector and exposed to the agent as an additional observation-side feature.

A projected scalar aesthetic score is also retained in the semantics artifact. These semantic features enrich the bundle representation available to the agent, but they do not replace the base simulator's hand-crafted aesthetic prior or alter the underlying catalog, persona schema, or negotiation mechanics.

E.2 Inference-time adaptation configuration

The reported inference-time adaptation result in Chapter 6 uses the final Dreamer-side paired comparison under the CLIP + soft-shortfall setting. The module is applied only at evaluation time on top of a fixed trained Dreamer checkpoint. It does not update model parameters online, and it does not modify the benchmark environment.

Table E.2: Key settings of the reported inference-time adaptation module.

Item	Reported setting
Configuration file	`configs/dreamer/ablations/bank50k/clip_tta_v2_soft_shortfall.yaml`
Checkpoint family	Dreamer + CLIP + soft-shortfall checkpoint
Adaptation status	enabled at benchmark time only
Mode	candidate reranking over a bounded local action set
Maximum candidate count	8
Offer neighbor bins	3
Maximum absolute price adjustment	300 USD
Imagination horizon	1 step
Belief-state update strengths	$\alpha_{\text{wtp}} = 0.45$, $\alpha_{\text{counter}} = 0.35$, $\alpha_{\text{risk}} = 0.40$
Scoring weights	$w_{\text{policy}} = 0.30$, $w_{\text{value}} = 0.00$, $w_{\text{margin}} = 0.40$, $w_{\text{feasibility}} = 0.08$, $w_{\text{risk}} = 0.05$
Reward setting during evaluation	CLIP-enabled soft-shortfall profile, identical to the paired no-TTA baseline

Under this configuration, the adaptation layer reranks a small local set of candidate actions around the raw Dreamer proposal using only observable negotiation information. In the reported run, the module behaves as a selective correction mechanism rather than as a hard policy replacement: only a small fraction of prediction steps are overridden, and the observed changes are concentrated on offer-side decisions. This appendix table is intended to make that reported module reconstructible without expanding the main text into a full implementation note.