REPRESENTING UNCERTAINTY IN STOCHASTIC PROGRAMMING USING SPNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimization under uncertainty is often made tractable by sampling: expectations and chance constraints are replaced by sample-average counterparts via scenario expansion, where meeting stronger statistical requirements may require larger samples and more sophisticated optimization methods. We take a different route and represent uncertainty using a tractable probabilistic model (TPM), specifically a sum-product network learned from data. Its structural properties (smoothness, decomposability) enable exact marginal inference under the fitted model, allowing us to construct conditional probability queries for constraint violations that depend on the decision variables. We represent these queries using refinable piecewise-linear approximations that admit explicit mixed-integer linear formulations. A learned chance constraint thus becomes part of the optimization problem itself, without explicit scenario expansion: for a fixed circuit architecture and encoding policy, additional training observations do not introduce individual optimization attachments. We evaluate the approach on the well-known newsvendor problem and a stochastic knapsack problem, examining how learning and encoding errors affect decision quality and computational cost. The resulting decisions are evaluated for feasibility under the compiled learned constraint and independently assessed against the uncertainty distribution.

1 INTRODUCTION

Stochastic programming (SP) is a mathematical optimization technique that addresses uncertainty by modeling certain parameters as random variables with associated probability distributions, making it suitable for real-world scenarios. For example, decisions related to inventory and storage capacity often need to be made without knowing future demand, which can lead to shortages or excess costs. A single forecast can obscure important trade-offs; decisions that seem feasible based on that forecast may not hold under different outcomes.

By incorporating uncertain parameters, SP allows for a more comprehensive assessment of potential outcomes and their associated probabilities (Birge & Louveaux, 2011). However, simply considering uncertainty in the objective function does not guarantee reliable results. Minimizing expected inventory costs, for instance, may lead to stock levels that perform well on average while still resulting in an unacceptably high probability of shortages. Therefore, when establishing service targets based on the likelihood of meeting demand, it is crucial to explicitly include those targets in the optimization problem (Van Der Laan et al., 2022).

Chance-constrained programming is an optimization method that addresses uncertainty by ensuring specific constraints are met with a designated minimum probability (Charnes & Cooper, 1959). Decision-makers set an acceptable probability of constraint violation, guiding the optimizer to find the best decision within that limit. This strategy balances cost and reliability, making it particularly beneficial when complete feasibility is prohibitively expensive or impractical.

The challenge lies in evaluating and enforcing complex probability requirements. For each decision, the violation probability must consider all uncertain scenarios where constraints may not be met. While assessing individual outcomes is simpler, calculating this probability often involves multidimensional integrals, and the distribution may need to be estimated from data. A common approach is sample-average approximation (SAA), which estimates the probability as the fraction of sampled scenarios where constraints are violated (Luedtke & Ahmed, 2008; Pagnoncelli et al., 2009).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

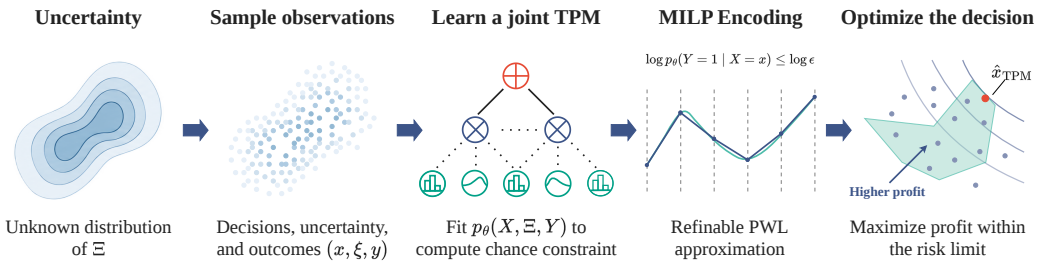

Figure 1: From observations to decisions. Uncertainty samples are paired with decisions and violation labels to train a joint SPN; dotted connections denote omitted circuit layers. Refinable piecewise-linear (PWL) approximations encode the circuit's conditional violation probability as mixed-integer linear constraints, without attaching individual observations. The solver maximizes profit over the encoded feasible set (shaded region), returning the selected decision (red point).

In explicit scenario formulations, increasing the sample size can enhance statistical accuracy, while incorporating additional constraints and violation indicators into the optimization model. Reformulation, strengthening, and decomposition methods aim to manage this computational growth (Küçükyavuz & Jiang, 2022; Porras et al., 2023). We explore a complementary hypothesis:

Can a learned probabilistic model use additional uncertainty data to improve probability estimates and decision quality without proportional growth of the optimization formulation?

To investigate this question, we use sum-product networks (SPNs, Poon & Domingos (2011)) to learn a tractable probabilistic representation from data. The challenge extends beyond evaluating violation probabilities at fixed decisions: their computations must be represented while the solver chooses the decision. Building on existing mixed-integer representations of SPNs (Němeček et al., 2025), we exploit their computational structure to encode conditional violation queries using refinable piecewise-linear approximations. The resulting learned chance constraints combine with the problem's objective and deterministic restrictions. Figure 1 summarizes this pipeline.

Our contributions We present (1) a framework for learning a joint probabilistic representation from uncertainty observations and embedding its conditional probability queries as chance constraints, without attaching each observation individually to the optimization problem; (2) an explicit, refinable piecewise-linear MILP encoding of SPN queries, with circuit-derived approximation-error bounds and strengthening strategies for supported circuit and decision structures; and (3) controlled SP studies examining how training-data size and encoding fidelity affect probability accuracy, formulation size, computational cost and decision quality, with explicit SAA as a sample-based reference.

2 Background and Related Work

2.1 Chance-Constrained Programming

Uncertainty can manifest in the form of an expectation, variance, or quantile of function values containing uncertain parameters (Birge & Louveaux, 2011, p 124). But it can also take the form of a probability that a given constraint holds. Such probability constraints, commonly called *chance constraints*, take the form of $\mathbb{P}\left(G(x, \Xi) > 0\right) \leq \epsilon$, where Ξ represents the uncertain parameters, x represents problem variables, $\epsilon \in [0, 1]$ is the probability threshold, and $G(x, \Xi) > 0$ denotes a violation; equality is permitted. This can be thought of as an "acceptable risk".

Formulations with such constraints fall into the category of chance-constrained programming (CCP) (Charnes & Cooper, 1959), sometimes also referred to as probability-constrained programming (Birge & Louveaux, 2011). We can define a general chance-constrained optimization problem as

$$\begin{aligned} \min_x \; & f(x) \\ \text{s.t. } & \mathbb{P}\left(G(x, \Xi) > 0\right) \leq \epsilon \\ & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{1}$$

Here, $f(x)$ is a deterministic function of the decision, which may represent an expected cost or a fixed numerical approximation of that expectation. The functions g and h describe deterministic restrictions on the decision variables, which may be continuous, integer, or binary. Together with the variable domains, these restrictions define the decision set $\mathcal{X}$. Even when decision variables are continuous and the deterministic constraints define a convex feasible set, adding a chance constraint can make that set nonconvex (Prékopa, 1995).

2.2 Mathematical Optimization Approaches

Converting a chance constraint into a deterministic optimization formulation depends on how uncertainty is described and on the structure of the function G. The main approaches exploit known distributional properties, approximate probabilities using sampled scenarios, or account for uncertainty through sets of possible realizations or distributions.

Analytical reformulations and approximations. For some distributions and constraint structures, the chance constraint admits an equivalent deterministic formulation (Prékopa, 1995; Birge & Louveaux, 2011). When a tractable exact reformulation is unavailable, Nemirovski & Shapiro (2007) construct convex conditions that ensure the chance constraint is met under their assumptions, but may exclude some feasible decisions. Additionally, Nemirovski (2012) reviews related sampling-based and simulation-free approximations for linear and conic convex problems, noting that their tractability does not automatically apply to integer decisions.

Sample-average approximation (SAA). Given N_{SAA} independent uncertainty samples $\xi^{(1)}, \ldots, \xi^{(N_{\mathrm{SAA}})}$, SAA replaces the underlying distribution with an empirical distribution $\widehat{\mathbb{P}}_{N_{\mathrm{SAA}}}$, assigning each realization uniform mass. This approach can be viewed as a mixture of Dirac distributions. The likelihood of violating a given constraint under this distribution precisely equals the proportion of sampled scenarios that do violate the constraint. The chance constraint becomes

$$\widehat{\mathbb{P}}_{N_{\mathrm{SAA}}}(G(x,\Xi) > 0) = \tfrac{1}{N_{\mathrm{SAA}}} \textstyle\sum_{s=1}^{N_{\mathrm{SAA}}} \left[\!\left[G(x,\xi^{(s)}) > 0 \right]\!\right] \leq \epsilon. \tag{2}$$

Pagnoncelli et al. (2009) analyzed the convergence properties of the SAA method, and derived statistical lower bounds on the optimal value, providing a theoretical basis for this sampled formulation. Furthermore, for finite decision sets and independent samples from the uncertainty distribution, Luedtke & Ahmed (2008) derive sample-size bounds ensuring true feasibility with prescribed confidence when the sampled risk limit is stricter than ϵ.

Scenario-dependent constraints and violation indicators can enlarge SAA models as sample size grows. Mixing and quantile inequalities, coefficient strengthening, decomposition, screening, and probabilistic-cover cuts can strengthen or reduce the resulting MIPs (Küçükyavuz, 2012; Song et al., 2014; Küçükyavuz & Jiang, 2022; Porras et al., 2023). In economic dispatch, Cheng et al. (2018, Sec. 5.1, Table 1) report computational difficulties with standard SAA and improvements from partial SAA. These works motivate a statistical–computational trade-off whose severity depends on the problem and formulation.

Scenario enforcement. The scenario approach requires the constraint to hold for every sampled uncertainty realization. This gives a deterministic optimization problem with constraints $G(x,\xi^{(s)}) \leq 0$ for all $s = 1, \ldots, N_{\mathrm{scen}}$. Unlike SAA with a positive violation allowance, this formulation permits no violations among the sampled constraints. Calafiore & Campi (2006) establish probabilistic bounds on the resulting solution's violation probability under structural assumptions, particularly convexity. These classical scenario guarantees do not apply generically to arbitrary integer chance-constrained programs. Dietrich et al. (2024) use nonconvex scenario optimization to estimate reachable sets directly from data, with probabilistic coverage guarantees.

Robust and distributionally robust optimization. Robust optimization requires constraints to hold for every realization in a prescribed uncertainty set; tractability depends on the set and constraint structure (Bertsimas et al., 2011). Distributionally robust chance constraints instead limit violation probability to ϵ under every distribution in an ambiguity set $\mathcal{P}$, rather than SAA's single empirical distribution (Küçükyavuz & Jiang, 2022). Chen et al. (2024) use a Wasserstein ball around the empirical distribution as the ambiguity set. They obtain exact mixed-integer conic reformulations for individual affine chance constraints and joint affine chance constraints with right-hand-side uncertainty, proposing conditions that reduce them to mixed-integer linear programs.

2.3 Learned Probabilistic Models in Optimization

Model learning enhances prediction-driven optimization decisions. Donti et al. (2017) optimize costs by learning conditional distributions of uncertain quantities, differentiating through solutions for convex problems, with applications in inventory management and energy scheduling. In contrast, Mandi et al. (2026) emphasize predicting constraint parameters to prevent infeasibility in decisions, balancing the risk of excluding good options with the need for feasible outcomes. SPNs have been integrated with mathematical optimization. Mauá et al. (2020) reformulate MAP inference in SPNs as combinatorial optimization problems, including mixed-integer linear formulations. Similarly, Němeček et al. (2025) use mixed-integer optimization for generating counterfactual explanations based on learned distributions. These works exploited SPN structure, which facilitates efficient marginal inference (Poon & Domingos, 2011; Choi et al., 2020).

3 Our Method

Our method learns a joint probabilistic model of decisions, uncertainty, and violation labels, then embeds an approximation of its conditional violation probabilities into optimization. Let x denote a candidate decision and ξ a realization of the uncertain parameters. To generate training data, we draw decisions X from a distribution p_X chosen to have positive mass or density throughout the decision domain of interest, and independently draw uncertainty realizations Ξ from the problem's unknown uncertainty distribution. For each pair, we record $Y = [\![G(X, \Xi) > 0]\!]$, where $Y = 1$ indicates violation and $Y = 0$ indicates satisfaction. Because X and Ξ are independent, $\mathbb{P}\left(G(x, \Xi) > 0\right) = \mathbb{P}\left(Y = 1 \mid X = x\right)$, which we obtain by integrating over the uncertainty:

$$\mathbb{P}\left(Y = 1 \mid X = x\right) = \int p_{\Xi|X}\left(\xi \mid x\right) [\![G(x, \xi) > 0]\!] \, \mathrm{d}\xi = \int p_{\Xi,Y|X}\left(\xi, 1 \mid x\right) \, \mathrm{d}\xi. \tag{3}$$

Here, $p_{\Xi,Y|X}\left(\xi, 1 \mid x\right)$ denotes the conditional joint density of the uncertainty realization $\Xi = \xi$ and the violation label $Y = 1$, given the decision $X = x$. Expressing this conditional density in terms of joint and marginal densities gives

$$\int p_{\Xi,Y|X}\left(\xi, 1 \mid x\right) \, \mathrm{d}\xi = \int \frac{p_{\Xi,Y,X}\left(\xi, 1, x\right)}{p_X\left(x\right)} \, \mathrm{d}\xi = \frac{1}{p_X\left(x\right)} \int p_{\Xi,Y,X}\left(\xi, 1, x\right) \, \mathrm{d}\xi. \tag{4}$$

We approximate the joint density $p_{\Xi,Y,X}\left(\xi, 1, x\right)$ appearing inside the integral with a tractable probabilistic model (TPM) learned from labeled observations. We denote the learned joint distribution by p_θ, where θ represents the model parameters, which remain fixed during decision optimization. Marginalizing the uncertainty and conditioning on the decision gives the learned probability:

$$\mathbb{P}\left(G(x, \Xi) > 0\right) \approx \mathbb{P}_\theta\left(Y = 1 \mid X = x\right) = \frac{p_\theta\left(x, 1\right)}{p_\theta\left(x\right)}, \qquad p_\theta\left(x\right) > 0. \tag{5}$$

Here, $p_\theta\left(x, 1\right)$ is the learned joint density of the decision and violation label after marginalizing uncertainty. The denominator $p_\theta\left(x\right) = p_\theta\left(x, 0\right) + p_\theta\left(x, 1\right)$ includes both label outcomes, so their ratio gives the violation probability conditional on the decision. The known decision-sampling density $p_X\left(x\right)$ can replace this denominator when it agrees with the fitted marginal; otherwise, the ratio generally differs from the fitted model's conditional probability. We use the fitted marginal throughout, giving the learned chance constraint:

$$p_\theta\left(x, 1\right) \leq \epsilon \cdot p_\theta\left(x\right). \tag{6}$$

The violation function G needs only to be evaluable to generate training labels; its possibly nonlinear structure does not enter the mixed-integer linear encoding of the learned probability.

3.1 Sum-Product Networks

Probabilistic models that support efficient marginalization while being expressive belong to a class of models called Tractable Probabilistic Models (TPMs) (Choi et al., 2020). Of the many TPMs, we choose Sum-Product Networks (SPNs) (Poon & Domingos, 2011) because of their expressiveness. SPNs are directed acyclic graphs in which the leaves represent simple (in our case, univariate) distributions, and the internal nodes represent more complex distributions either through products

(factorization) or weighted sums (mixtures). The products combine child distributions defined on disjoint sets of variables (a property known as decomposability), whereas the sums combine children distributions over the same set of variables (referred to as smoothness). These structural properties enable exact marginal inference in time linear in the graph size (Choi et al., 2020). Inference is exact for the fitted distribution, but the estimated probabilities remain subject to estimation error. Writing o_v for node v's output and $\mathrm{ch}(v)$ for its children, the internal computations are

$$o_v = \begin{cases} \prod_{j \in \mathrm{ch}(v)} o_j, & v \text{ a product node,} \\ \sum_{j \in \mathrm{ch}(v)} \lambda_{vj} o_j, & v \text{ a sum node,} \end{cases} \tag{7}$$

where λ_{vj} are nonnegative sum weights that sum to one at each sum node. With a sum node at the root, the SPN represents a weighted mixture of its children's distributions over all modeled variables.

Constructing an SPN. The structure can be learned from data or predetermined in advance. The LearnSPN algorithm (Gens & Pedro, 2013) recursively divides variables and groups observations, while Prometheus constructs variable decompositions with shared substructures (Jaini et al., 2018). Alternatively, RAT-SPNs use randomly generated circuit structures followed by parameter learning (Peharz et al., 2020b). In addition to learning the graph, Zhang et al. (2025) have developed algorithms to restructure existing circuits, facilitating inference and reducing circuit depth.

Parameter learning. Let θ be the SPN's sum weights and leaf parameters, which can be fitted using expectation-maximization or gradient-based methods (Peharz et al., 2020a; Choi et al., 2020). SPNs accommodate both joint-likelihood and conditional-likelihood training, enabling label probabilities to be fitted based on observed inputs (Gens & Domingos, 2012). Recent advancements enhance training scalability through GPU execution for large probabilistic circuits (Liu et al., 2024) and distributed learning from partitioned datasets (Seng et al., 2025). We optimize the joint negative log-likelihood (NLL) of decisions, uncertainty, and violation labels:

$$\mathcal{L}(\theta) = -\tfrac{1}{N} \textstyle\sum_{r=1}^{N} \log p_\theta \left(x^{(r)}, \xi^{(r)}, y^{(r)} \right). \tag{8}$$

Here, N is the number of labeled training observations. The experimental setup specifies the circuit structure and the criterion used to select the fitted model.

3.2 From SPN Evaluation to Linear Constraints

Němeček et al. (2025, Section 4) formulate the node computations in Equation 7 as a mixed-integer linear program (MILP) in log space, turning products into linear sums and approximating each sum node by its largest weighted child contribution. Our conditional query requires numerator and denominator circuits depending only on x: uncertainty is marginalized in both, while Y is fixed to 1 for the numerator and marginalized for the denominator.

Because the optimization objective need not involve probability, the encoding must enforce both computations through constraints linking each node's output to those of its children. The resulting root outputs, denoted by $\widehat{\mathrm{SPN}}(x, 1)$ and $\widehat{\mathrm{SPN}}(x)$, approximate $\log p_\theta(x, 1)$ and $\log p_\theta(x)$, respectively. For positive marginals and $0 < \epsilon < 1$, the encoding of Equation 6 is

$$\widehat{\mathrm{SPN}}(x, 1) - \widehat{\mathrm{SPN}}(x) \leq \log \epsilon. \tag{9}$$

Solver feasibility and optimality certificates apply to the encoded model, within the specified solver tolerances. Learning and encoding errors can still render a decision that satisfies this constraint infeasible under the true uncertainty distribution.

3.3 A Piecewise-Linear Approximation of Sum Nodes

Max approximation underestimates each marginal, but their ratio can err in either direction. We retain all child contributions using refinable piecewise-linear (PWL) approximations in log space. For a node output $o_v > 0$, define its log output as $z_v = \log o_v$. In the case of a sum node with m children, we express the contribution to the parent node as $t_j = \log \lambda_{vj} + z_j$, where the weights λ_{vj} are fixed and positive after removing zero-weight edges. We set $t_{\max} = \max_j t_j$ and define the shifted inputs as $u_j = t_j - t_{\max}$. Thus, for a sum node, this log output takes the form

$$z_v = \log \textstyle\sum_{j=1}^{m} e^{t_j} = t_{\max} + \log \left(\sum_{j=1}^{m} e^{u_j} \right). \tag{10}$$

Replacing the exponential and logarithm by PWL interpolants $\widehat{E}$ and $\widehat{H}$ on bounded domains gives

$$\widehat{z}_v = t_{\max} + \widehat{H}\left(\sum_{j=1}^{m} \widehat{E}(u_j)\right). \tag{11}$$

Each interpolant has a MILP representation using segment-selection variables, and additional knots allow its accuracy to be refined. Binary selection and big-M constraints enforce the maximum independently of the objective, extending the formulation of Němeček et al. (2025, Section 4). Sum nodes with a single child require no approximation. Formulation size depends on the retained circuit, leaf encodings, and PWL knot counts, without individual attachments for training observations. Refer to the appendix for details on leaf and maxima encoding (Appendix B.2), the complete PWL formulation and its bounds (Appendix B.4), and the numerical settings (Appendix B.6).

Approximation error. Local PWL approximation errors propagate through the circuit and can change whether the encoded constraint accepts a decision. Finer approximations reduce interpolation error but enlarge the formulation. The circuit structure allows local error bounds to be propagated to the final conditional probability query. Propagated error bounds can conservatively tighten the constraint for the fitted SPN (Appendices B.6–B.7); the experiments use the nominal encoding, so its certificates do not guarantee population feasibility.

Strengthening the compiled formulation. Relaxing discrete decisions weakens their links to auxiliary circuit variables. We therefore tighten variable and PWL domains with propagated affine and conditional bounds and derive callback cuts from the compiled query. These cuts exclude either one infeasible decision or a certified infeasible binary face or categorical group, without using population risk. In matched ablations, cut insertion tightens all six bounds, improves two knapsack incumbents, and yields four optimality certificates versus none without cuts (Appendix F.5.6).

4 Experiments

We use controlled stochastic programming settings to assess probability accuracy, decision quality, and computational cost. After describing model fitting, selection, and evaluation, we examine sensitivity to training-data size and PWL resolution, then compare optimization outcomes with standard explicit SAA as a sample-based reference. The two settings are chance-constrained variants of the classical knapsack and newsvendor problems. We outline them below; their full formulations, parameters, and reference solutions appear in Appendix E.1 and Appendix F.2, respectively.

Knapsack problem. The goal is to choose the most profitable set of items while limiting the probability that their uncertain total weight exceeds capacity (King & Wallace, 2024, p. 41). Profits are known, so uncertainty affects only feasibility; our small instance allows every selection to be assessed, revealing how probability errors change the chosen solution.

Newsvendor problem. A retailer chooses how many units of each product to order before demand is known, seeking to maximize expected profit within a shared storage limit (Van Der Laan et al., 2022). Demand uncertainty affects both profit and feasibility, with a chance constraint limiting the probability that total unmet demand exceeds a specified allowance.

4.1 Experimental Protocol

Each problem instance utilizes ten independently generated datasets. Each one yields one fitted SPN along with four SAA models, each featuring nested samples of 1,000, 10,000, 60,000, and 200,000 uncertainty observations. The training encompasses all legal decisions in both main experiments. The knapsack method allocates observations uniformly across selections. In contrast, the newsvendor method combines comprehensive coverage with additional observations focused near the risk boundary and high-profit orders (details in Appendix F.3.2). All newsvendor methods use an expected-profit objective estimated from the complete demand bank; SAA sample sizes apply only to the chance constraint. The summary of the protocol is provided in Table 6 in the appendix.

Fitting and model selection. Both problems utilize 1,024-component SPN mixtures with univariate leaves: Bernoulli for binary decisions and labels, categorical for integer orders, and Gaussian for modeling uncertainty. Parameters are optimized with Adam to minimize the joint NLL in Equation 8, with checkpoints determined via validation data. The architectural choices, fitting recipes, and checkpoint selection rules are detailed in Appendix E.3.

Evaluation method. We assess the learning error by comparing the population risk $r^\star(x)$ with the native SPN risk $r_\theta(x) = \mathbb{P}_\theta(Y = 1 \mid X = x)$. Additionally, we evaluate the encoding error by measuring the difference between $r_\theta(x)$ and its encoded approximation $\widetilde{r}_\theta(x)$. Our report includes the mean absolute error (MAE) in percentage points (pp) and the false-safe rate, which indicates the fraction of decisions we assessed to be feasible but that are actually infeasible for the population. Using equal time budgets for solvers, we assess returned profit, population feasibility, solver gap, and optimality certification for each method's specific formulation.

To evaluate decision quality, we compute optimal solutions based on the data-generating distributions, which are available only for evaluation. For the knapsack problem, we employ an exact reformulation that utilizes Gaussian weights. For the newsvendor problem, we explore the discrete decision space using analytically computed risk and profit (see Appendices E.2.2 and F.2). These references are used solely for assessment purposes and are not provided to the solvers. Additional details about the solver and assessment processes are in Appendix C.2 and Appendix C.

4.2 SPN Accuracy and Encoding Fidelity

We begin by examining whether increasing the amount of training data enhances the accuracy of probability estimates. In this analysis, we maintain a consistent architecture, encoding policy, and training updates across three datasets for each problem. The training sizes vary from 10,240 to 3,686,400 labeled rows for the knapsack problem and from 301,971 to 30,197,100 for the newsvendor problem. Across the three datasets, dataset-averaged native MAE decreases from 21.255 to 0.798 percentage points for knapsack and from 1.860 to 1.334 percentage points for newsvendor as training size increases, while submitted-model size remains approximately constant under the fixed architecture and encoding policy. However, the newsvendor problem demonstrates diminishing returns, indicating that improved estimates do not always lead to increased profits (refer to Appendix F.5.4).

We assess encoding fidelity and find that PWL reproduces SPN probabilities more accurately than max approximation (see Appendix D). Refining PWL-9/17/33 on unchanged fitted models reduces mean encoding MAE substantially: from 0.54 to 0.08 pp for knapsack and from 0.44 to 0.06 pp for newsvendor, though this results in larger formulations. Returned knapsack decisions change across resolutions, whereas the newsvendor study returns the same orders at PWL-9/17/33. Solver certification does not improve consistently. These studies highlight that additional data enhances probability estimates, while PWL refinement improves their representation in optimization models. However, neither improvement ensures a better decision individually. For full comparisons and model details, see Appendices F.5.5 and F.5.8.

4.3 Solution Quality and Computational Performance

Explicit SAA serves as a sample-based reference (Pagnoncelli et al., 2009; Luedtke & Ahmed, 2008), not a comprehensive benchmark: stronger variants exist (Küçükyavuz & Jiang, 2022; Porras et al., 2023), and differences in information allocation and solver settings preclude general dominance claims.

Stochastic knapsack. The ten-item problem allows for a complete assessment of all $2^{10} = 1,024$ possible selections. Figure 2 illustrates all fifty returns within the population risk-profit landscape. The TPM PWL-33 method yields population-feasible decisions across all ten datasets, achieves the population optimum in seven cases, and confirms its encoded optimum in nine. For all fitted models, the mean native MAE across all 1,024 decisions is 0.756 percentage points, ranging from 0.648 to 0.809 pp. The average population false-safe rate is 1.06%, with values ranging from 0.50% to 1.71%.

Increasing the scenario count improves feasibility but complicates the SAA optimization. SAA-1k solves models quickly but yields only five population-feasible returns. In comparison, SAA-10k achieves seven returns that are both feasible and optimal, with a lower median time of 554.8 seconds versus TPM's 1,065.9 seconds. At 60,000 and 200,000 scenarios, all returns are population-feasible; the time-limited searches remain uncertified and return population-suboptimal decisions. In this campaign, TPM matches SAA-10k's optimal recovery, and all ten returned TPM decisions are population-feasible. Complete results and error analysis are available in Appendix E.5.1.

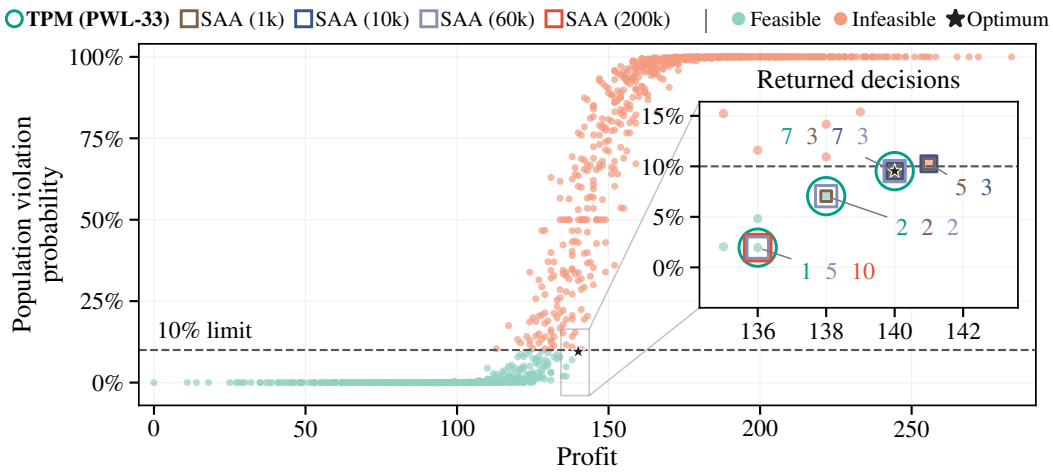

Figure 2: Population risk and profit landscape of 1,024 knapsack decisions with fifty optimization returns. Background colors show feasibility; the dashed line represents the risk limit, while the star marks the population optimum. Inset markers indicate methods, and colored counts follow the legend, with nested markers for coincident returns.

Multi-product newsvendor. The five-product problem results in 6,039,420 valid order vectors (see Figure 3). The TPM PWL-9 method produces population-feasible decisions across all ten datasets, achieves the population optimum in four datasets, and confirms its encoded sample-profit optimum in nine.

Across the ten fitted models, validation-panel native MAE averages 0.892 pp, with a range of 0.829–0.970 pp. The corresponding empirical false-safe rate averages 12.6%, ranging from 10.9% to 16.0% among validation orders predicted to be feasible. Further refinement of the PWL did not enhance the decisions in the sensitivity study, indicating that more accurate probability estimation could be more beneficial than finer encoding in this context. Additionally, incomplete search processes and discrepancies between the sampled and population objectives may contribute to decision errors.

SAA-10k and the larger tested SAA formulations identify the population optimum in every dataset. While SAA-1k is faster, it produces some suboptimal feasible decisions and one population-infeasible decision. The median initialization and search times are 37.3 s for SAA-10k and 313.8 s for TPM. Complete results and error analyses can be found in Appendix F.5.

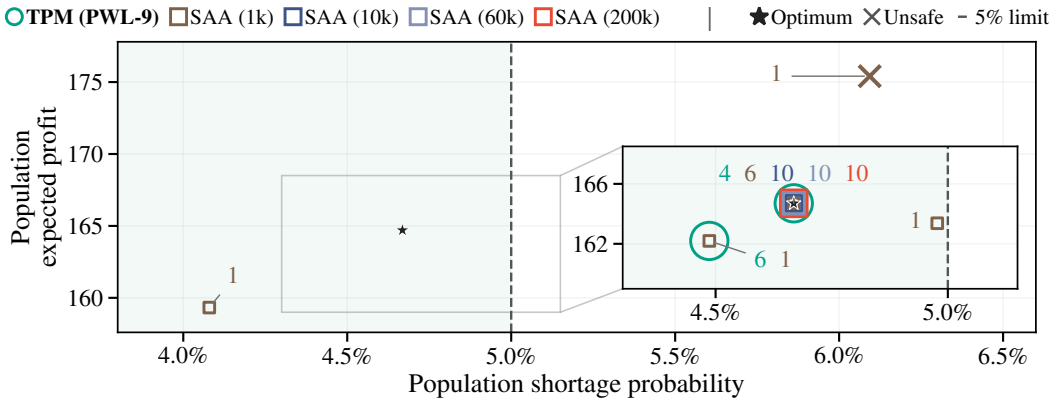

Figure 3: Population risk and profit across fifty newsvendor optimization returns. Shading indicates feasible regions; the dashed line marks the risk limit, the star represents the population optimum, and the cross indicates an unsafe return. The inset distinguishes nearby decisions, with colored counts following the legend order and nested markers highlighting coincident returns.

Formulation size and preparation cost. TPM represents training observations without attaching each observation to the optimization problem individually. However, its compiled formulation can still be large: knapsack PWL-33 has more variables and linear constraints than SAA-60k, despite their different search behaviors. Fitting and compilation also add substantial preparation costs to the reported search times (Table 1).

Table 1: Model sizes and computation times (medians over ten datasets). Sizes precede presolve or rebuilding; variable totals include binaries. TPM also uses 2,050 SOS2 sets. Fitting is separate (+time); search includes initialization and rebuilding. Further detail in Appendix C.3.

	Knapsack				Newsvendor			
Method	Variables (binary)	Linear rows	Build/check (s)	Search (s)	Variables (binary)	Linear rows	Build/check (s)	Search (s)
SAA 1k	1,010 (1,010)	1,001	0.3	2.6	5,608 (1,206)	5,294	0.2	2.4
SAA 10k	10,010 (10,010)	10,001	0.5	554.8	30,202 (5,304)	29,888	0.7	37.3
SAA 60k	60,010 (60,010)	60,001	2.0	1,800.4	76,921 (13,091)	76,607	1.9	351.9
SAA 200k	200,010 (200,010)	200,001	6.1	1,800.2	126,340 (21,328)	126,026	2.8	1,101.6
TPM	102,063 (2,058)	161,987	236.3 (+697.0)	1,065.9	28,143 (2,374)	77,933	30.1 (+2,310.6)	313.8

Sensitivity to risk limit. We assess sensitivity to risk by halving and doubling its original value while maintaining consistent fitted SPNs, encoding policies, SAA samples, and objectives. The violation events and uncertainty distributions remain unchanged. The results in Table 2 indicate that feasibility at the original risk limit does not guarantee it at altered thresholds. For the newsvendor problem, training allocations focus on decisions near the original risk boundary and on orders with high estimated profits (see Appendix F.3.2). These findings highlight the need to adapt this allocation method to address feasibility boundaries for varying risk limits. Additional outcomes are detailed in Appendix F.5.7.

Table 2: Risk-tolerance sensitivity. Each cell reports population-feasible returns, with population optimal returns in parentheses, both out of the three reused datasets. Original risk limits (ϵ): 10% for knapsack and 5% for newsvendor.

Problem	ϵ (%)	**TPM**	SAA 1k	SAA 10k	SAA 60k	SAA 200k
Knapsack	5	3 (3)	3 (3)	3 (3)	3 (3)	3 (3)
	20	1 (0)	0 (0)	0 (0)	3 (0)	3 (0)
Newsvendor	2.5	1 (1)	2 (2)	3 (3)	3 (3)	3 (3)
	10	2 (0)	1 (1)	3 (3)	3 (3)	3 (3)

5 Limitations

Our experiments investigate two synthetic instances with complete decision coverage and post hoc configuration selection. A next step is to extend this evaluation to larger problems, unseen decisions, and real-world data. Solver certificates are tied to the encoded model, meaning discrepancies during learning can impact population feasibility. When reusing a fitted model at different risk levels, feasibility is not guaranteed; thus, broader training coverage or additional targeted data could enhance robustness. Our experiments utilize shallow mixtures and explicit SAA formulations with varying information budgets, highlighting the computational traits of the formulations compared. While fitting and compilation incur initial costs, the learned representation can be reused across optimization instances, potentially amortizing these optimization costs. Investigating the lasting benefits in broader architectures and repeated-decision settings is a promising area for future research.

6 Discussion and Conclusion

This work introduces a framework that learns a joint SPN from uncertainty observations, representing conditional probability queries as MILP constraints. The piecewise-linear approximation can be enhanced, and using circuit-derived bounds and cuts strengthens the optimization model. Experiments show that additional training data can enhance probability estimates without increasing formulation complexity. These experiments provide a proof of concept on two finite, fully covered decision domains; they do not establish population-feasibility guarantees or superiority over SAA. The key contribution is decoupling the data used to learn the uncertainty distribution from the representation used for optimization, so that formulation complexity is governed by the learned circuit and its encoding rather than by the amount of training data.

AI USE STATEMENT

A generative AI tool assisted with manuscript editing, literature checks, and the reporting-only figure and table generator. This revision used persisted experimental evidence and did not run new fitting, compilation, or optimization experiments. The quantitative claims were checked against the stored experiment artifacts. The authors remain responsible for the final text and claims.

REPRODUCIBILITY STATEMENT

We provide an **anonymized code repository** that contains the implementation and experimental pipeline used in this study. This repository supports various levels of reproducibility, depending on the available computational resources. A lightweight smoke test runs the entire pipeline end-to-end. The full experimental configurations allow for the reproduction of the reported experiments, contingent on the necessary computational resources and execution time. Additionally, the experiment artifacts can be accessed directly to reproduce the analyses and figures presented in the paper, eliminating the need to rerun the computationally intensive experiments. The repository includes detailed instructions, configurations, and scripts for these workflows. Further experimental details, such as data generation, model configurations, and MILP formulation, as well as solver settings, are documented in the relevant sections and appendices of the paper.

REFERENCES

Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. doi: 10.1137/080734510. URL https://epubs.siam.org/doi/10.1137/080734510.

John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2011. ISBN 978-1-4614-0236-7 978-1-4614-0237-4. doi: 10.1007/978-1-4614-0237-4. URL https://link.springer.com/10.1007/978-1-4614-0237-4.

G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, May 2006. ISSN 1558-2523. doi: 10.1109/TAC.2006.875041. URL https://ieeexplore.ieee.org/abstract/document/1632303.

A. Charnes and W. W. Cooper. Chance-Constrained Programming. *Management Science*, 6 (1):73–79, October 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.73. URL https://pubsonline.informs.org/doi/10.1287/mnsc.6.1.73.

Zhi Chen, Daniel Kuhn, and Wolfram Wiesemann. Technical Note—Data-Driven Chance Constrained Programs over Wasserstein Balls. *Operations Research*, 72(1):410–424, January 2024. ISSN 0030-364X. doi: 10.1287/opre.2022.2330. URL https://pubsonline.informs.org/doi/10.1287/opre.2022.2330.

Jianqiang Cheng, Richard Li-Yang Chen, Habib N. Najm, Ali Pinar, Cosmin Safta, and Jean-Paul Watson. Chance-constrained economic dispatch with renewable energy and storage. *Computational Optimization and Applications*, 70(2):479–502, June 2018. ISSN 1573-2894. doi: 10.1007/s10589-018-0006-2. URL https://doi.org/10.1007/s10589-018-0006-2.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: a unifying framework for tractable probabilistic models. tex.note: ˜, 2020. URL https://yoojungchoi.github.io/files/ProbCirc20.pdf.

Elizabeth Dietrich, Alex Devonport, and Murat Arcak. Nonconvex scenario optimization for data-driven reachability. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, pp. 514–527. PMLR, June 2024. URL https://proceedings.mlr.press/v242/dietrich24a.html.

Priya Donti, Brandon Amos, and J. Zico Kolter. Task-based End-to-end Model Learning in Stochastic Optimization. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3fc2c60b5782f641f76bcefc39fb2392-Abstract.html.

Robert Gens and Pedro Domingos. Discriminative learning of Sum-Product Networks. In *Advances in neural information processing systems*, volume 25, pp. 2159–2167. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/hash/573f7f25b7b1eb79a4ec6ba896debefd-Abstract.html.

Robert Gens and Domingos Pedro. Learning the Structure of Sum-Product Networks. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 873–880. PMLR, May 2013. URL https://proceedings.mlr.press/v28/gens13.html.

Priyank Jaini, Amur Ghose, and Pascal Poupart. Prometheus : Directly Learning Acyclic Directed Graph Structures for Sum-Product Networks. In *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*, pp. 181–192. PMLR, August 2018. URL https://proceedings.mlr.press/v72/jaini18a.html.

Alan J. King and Stein W. Wallace. *Modeling with Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, 2024. ISBN 978-3-031-54549-8 978-3-031-54550-4. doi: 10.1007/978-3-031-54550-4. URL https://link.springer.com/10.1007/978-3-031-54550-4.

Simge Küçükyavuz. On mixing sets arising in chance-constrained programming. *Mathematical Programming*, 132(1):31–56, April 2012. ISSN 1436-4646. doi: 10.1007/s10107-010-0385-3. URL https://doi.org/10.1007/s10107-010-0385-3.

Simge Küçükyavuz and Ruiwei Jiang. Chance-constrained optimization under limited distributional information: A review of reformulations based on sampling and distributional robustness. *EURO Journal on Computational Optimization*, 10:100030, January 2022. ISSN 2192-4406. doi: 10.1016/j.ejco.2022.100030. URL https://www.sciencedirect.com/science/article/pii/S2192440622000065.

Anji Liu, Kareem Ahmed, and Guy Van Den Broeck. Scaling Tractable Probabilistic Circuits: A Systems Perspective. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 30630–30646. PMLR, July 2024. URL https://proceedings.mlr.press/v235/liu24a.html.

James Luedtke and Shabbir Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008. doi: 10.1137/070702928. URL https://epubs.siam.org/doi/10.1137/070702928.

Jayanta Mandi, Marianne Defresne, Senne Berden, and Tias Guns. Feasibility-Aware Decision-Focused Learning for Predicting Parameters in the Constraints. *Advances in Neural Information Processing Systems*, 38:163285–163309, April 2026. doi: 10.52202/085713-5451. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ef158a0d4f5364741a571b8d1d44fb1b-Abstract-Conference.html.

Denis Deratani Mauá, Heitor Ribeiro Reis, Gustavo Perez Katague, and Alessandro Antonucci. Two Reformulation Approaches to Maximum-A-Posteriori Inference in Sum-Product Networks. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, pp. 293–304. PMLR, February 2020. URL https://proceedings.mlr.press/v138/maua20a.html.

Arkadi Nemirovski. On safe tractable approximations of chance constraints. *European Journal of Operational Research*, 219(3):707–718, June 2012. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.11.006. URL https://www.sciencedirect.com/science/article/pii/S0377221711009933.

Arkadi Nemirovski and Alexander Shapiro. Convex Approximations of Chance Constrained Programs. *SIAM Journal on Optimization*, 17(4):969–996, January 2007. ISSN 1052-6234. doi: 10.1137/050622328. URL https://epubs.siam.org/doi/abs/10.1137/050622328.

Jiří Němeček, Tomáš Pevný, and Jakub Mareček. Generating likely counterfactuals using sum-product networks. In *The thirteenth international conference on learning representations*, 2025. URL https://openreview.net/forum?id=rGyi8NNqB0.

B. K. Pagnoncelli, S. Ahmed, and A. Shapiro. Sample Average Approximation Method for Chance Constrained Programming: Theory and Applications. *Journal of Optimization Theory and Applications*, 142(2):399–416, August 2009. ISSN 0022-3239, 1573-2878. doi: 10.1007/s10957-009-9523-6. URL http://link.springer.com/10.1007/s10957-009-9523-6.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van Den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 7563–7574. PMLR, November 2020a. URL https://proceedings.mlr.press/v119/peharz20a.html.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 334–344. PMLR, August 2020b. URL https://proceedings.mlr.press/v115/peharz20a.html.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690, November 2011. doi: 10.1109/ICCVW.2011.6130310. URL https://ieeexplore.ieee.org/abstract/document/6130310.

Álvaro Porras, Concepción Domínguez, Juan Miguel Morales, and Salvador Pineda. Tight and Compact Sample Average Approximation for Joint Chance-Constrained Problems with Applications to Optimal Power Flow. *INFORMS Journal on Computing*, 35(6):1454–1469, November 2023. ISSN 1091-9856. doi: 10.1287/ijoc.2022.0302. URL https://pubsonline.informs.org/doi/abs/10.1287/ijoc.2022.0302.

András Prékopa. *Stochastic programming*, volume 324 of *Mathematics and its applications*. Springer, Dordrecht, 1 edition, 1995. ISBN 978-0-7923-3482-8. doi: 10.1007/978-94-017-3087-7. URL https://link.springer.com/book/10.1007/978-94-017-3087-7.

Jonas Seng, Florian Peter Busch, Pooja Prasad, Devendra Singh Dhami, Martin Mundt, and Kristian Kersting. Scaling Probabilistic Circuits via Data Partitioning. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, pp. 3701–3717. PMLR, July 2025. URL https://proceedings.mlr.press/v286/seng25a.html.

Yongjia Song, James R. Luedtke, and Simge Küçükyavuz. Chance-Constrained Binary Packing Problems. *INFORMS Journal on Computing*, 26(4):735–747, November 2014. ISSN 1091-9856. doi: 10.1287/ijoc.2014.0595. URL https://pubsonline.informs.org/doi/abs/10.1287/ijoc.2014.0595.

Niels Van Der Laan, Ruud H. Teunter, Ward Romeijnders, and Onur A. Kilic. The data-driven newsvendor problem: Achieving on-target service-levels using distributionally robust chance-constrained optimization. *International Journal of Production Economics*, 249:108509, July 2022. ISSN 09255273. doi: 10.1016/j.ijpe.2022.108509. URL https://linkinghub.elsevier.com/retrieve/pii/S0925527322001025.

Juan Pablo Vielma. Mixed Integer Linear Programming Formulation Techniques. *SIAM Review*, 57 (1):3–57, January 2015. ISSN 0036-1445. doi: 10.1137/130915303. URL https://epubs.siam.org/doi/10.1137/130915303.

Honghua Zhang, Benjie Wang, Marcelo Arenas, and Guy Van den Broeck. Restructuring Tractable Probabilistic Circuits. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, pp. 2566–2574. PMLR, April 2025. URL https://proceedings.mlr.press/v258/zhang25f.html.

This appendix collects the technical and empirical details supporting the main paper. Appendix A sets out notation and shared conventions, and Appendix B develops the SPN encoding. Appendices D, E and F collect the setup, learning, optimization and results for the scalar diagnostic, stochastic knapsack and multi-product newsvendor, respectively.

A Notation and Conventions

Lowercase letters are used to represent scalar or vector values. Vectors are not distinguished by boldface, and subscripts indicate their coordinates. All logarithms are natural logarithms. Uppercase letters such as X, Ξ, and Y represent random variables or vectors, with their realizations denoted as x, ξ, and y, respectively. Calligraphic letters are used to denote sets or the loss function $\mathcal{L}$, while $\mathbb{R}$ represents the set of real numbers. Superscripts in parentheses are used to index observations. Hats indicate encoded approximations unless specified otherwise; $\widetilde{r}_\theta$ represents encoded risk, and $r^\star$ denotes population risk. We use mixed-integer linear optimization (MILO) for the modeling and solution process. The explicit linear segment-selection encoding is a mixed-integer linear program (MILP). In the experiments, we supplied the same PWL graphs to Gurobi through its SOS2 interface as an implementation choice; the solver may then reformulate those sets internally. Tables 3 and 4 organize the shared notation by use, while experiment-specific symbols are defined locally.

Table 3: Core mathematical notation. Symbols with several uses are qualified by context.

General notation and operators	
$\mathbb{P}(\cdot)$	Population probability under the stated uncertainty law.
$\mathbb{E}(\cdot)$	Expectation under the stated uncertainty distribution.
$\widehat{\mathbb{P}}_{N_{\mathrm{SAA}}}$	Empirical distribution assigning uniform mass to the N_{SAA} uncertainty draws.
$[\![\cdot]\!]$	Indicator of an event.
$(\cdot)^\top$	Transpose of a matrix or vector.
$\Phi,\ \Phi^{-1}$	Standard normal cumulative distribution function and its inverse.
N	Number of labeled training rows.
N_{SAA}	Number of uncertainty draws in SAA.
N_{scen}	Number of draws in scenario enforcement.
Chance-constrained optimization	
x	Candidate decision.
$\mathcal{X}$	Decision domain satisfying deterministic restrictions, including integrality.
$\Xi,\ \xi$	Random uncertainty and a realization of it.
$f(x)$	Deterministic optimization objective.
$g(x)$	Deterministic inequality restrictions.
$h(x)$	Deterministic equality restrictions.
$G(x,\xi)$	Violation function: $G>0$ violates the requirement; $G\le 0$ satisfies it.
ϵ	Allowed violation probability; finite log encodings require $0<\epsilon<1$.
$r^\star(x)$	Population violation probability at decision x.
$r_\theta(x),\ \widetilde{r}_\theta(x)$	Native fitted-model risk and its encoded approximation.
Probabilistic models and SPNs	
X	Sampled decision variable.
$p_X,\ p_X(x)$	Decision-design distribution and its density or mass at x.
$Y,\ y$	Violation label and its value: $Y=[\![G(X,\Xi)>0]\!]$.
p_θ	Learned joint density or mass function.
θ	Fitted model parameters, fixed during optimization.
$p_\theta(x,y)$	Decision–label joint density or mass after integrating out uncertainty.
$p_\theta(x)$	Decision marginal of the fitted model.
$\mathbb{P}_\theta(Y=1 \mid X=x)$	Learned conditional violation probability.
$\mathcal{L}(\theta)$	Mean joint negative log-likelihood of labeled training observations.
v	Circuit-node index.
j	Child index at a sum or product node.
$\mathrm{ch}(v)$	Children of circuit node v.
o_v	Output of node v.
ℓ_v	Density or probability-mass function at leaf v.
λ_{vj}	Nonnegative weight from sum node v to child j; weights sum to one.

Table 4: Mixed-integer linear optimization encoding notation. Root subscripts identify the fixed label (0 or 1) or a marginalized label.

Log-space quantities	
$z_v,\ \widehat{z}_v$	Native log output $\log o_v$ and the encoded approximation at node v.
$\widehat{\mathrm{SPN}}(x,1),\ \widehat{\mathrm{SPN}}(x)$	Encoded roots approximating $\log p_\theta\,(x,1)$ and $\log p_\theta\,(x)$.
$z_1,\ z_0$	Root log outputs with Y fixed to 1 and 0, respectively.
z_{marg}	Root log output with Y marginalized.
Sum-node transformation	
$m,\ m_v$	Number of retained children of a sum node (of node v).
t_j	Weighted child log contribution $\log \lambda_{vj} + z_j$.
$t_{\max}$	Maximum child score $\max_j t_j$.
u_j	Shifted score $t_j - t_{\max} \le 0$.
e_j	Encoded exponential value $\widehat{E}(u_j)$ for child j.
Q	Sum of encoded exponentials, $\sum_j e_j \in [1, m]$.
$F_v(t),\ \widehat{F}_v(t)$	Native and encoded sum-node outputs at fixed child scores t.
PWL approximation	
$\widehat{E}$	PWL approximation to $\exp$ on shifted scores.
$\widehat{H}$	PWL approximation to $\log$ on summed contributions.
$f,\ \widehat{f}$	Function and its interpolant in the generic PWL construction.
$u,\ \omega$	PWL input and output, with $\omega = \widehat{f}(u)$.
$a_0, \ldots, a_R$	Ordered interpolation knots.
$R,\ r$	Number of PWL segments and segment index.
$\delta_{E,v}$	Uniform error of the full exponential table at node v, including its guard.
$\delta_{H,v}$	Uniform logarithm-approximation error at node v.
η_v	Local log-output error bound $m_v\delta_{E,v} + \delta_{H,v}$.
Δ_v	Propagated uniform log-error bound at node v.
$\Delta_1,\ \Delta_0$	Log-error bounds for roots with $Y = 1$ and $Y = 0$.
Δ_{marg}	Log-error bound for the marginalized root.
Auxiliary variables, bounds, and indices	
d_j	Binary selector of a maximizing child; exactly one is active.
b_r	Binary selector for PWL segment r.
$\alpha_r,\ \beta_r$	Nonnegative endpoint weights on segment r; $\alpha_r + \beta_r = b_r$.
$\mathcal{K}_i$	Categories of decision coordinate i.
d_{ik}	Exactly-one indicator for category $k \in \mathcal{K}_i$.
$\underline{t}_j,\ \overline{t}_j$	Lower and upper bounds on child score t_j.
$\underline{t},\ \overline{t}$	Smallest lower bound and largest upper bound on child scores.
$\overline{B},\ \overline{B}_v$	Width of a shifted-score interval $[-\overline{B}, 0]$, optionally indexed by node.
$\iota \in \{1, 0, \mathrm{marg}\}$	Query-root index: violation label, success label, or marginalized label.

B MILP ENCODING OF SPNS

B.1 OVERVIEW

This appendix concludes the construction of probability queries discussed in subsection 3.2. As previously explained, both the numerator and denominator utilize the same decision variables along with fixed fitted parameters. Uncertainty is marginalized in both queries; however, the violation label is only fixed in the numerator. The root outputs denoted in the main text by $\widehat{\mathrm{SPN}}(x, 1)$ and $\widehat{\mathrm{SPN}}(x)$ are written here as $\widehat{z}_1(x)$ and $\widehat{z}_{\mathrm{marg}}(x)$, respectively.

The encoding operates in log space, converting product-node computations into sums of child outputs. The following subsections explain how we propose to encode leaf nodes under this regime, and the special case of the sum-node operations, comparing max approximation against the piecewise-linear counterpart. We then provide numerical safeguards and error bounds that could be considered when implementing the encoding. The node-level construction applies to smooth, decomposable circuits with supported leaf encodings and finite valid bounds. The experiments evaluate only shallow fixed mixtures and therefore do not establish computational scalability for deeper circuits.

B.2 LEAF NODES AND MARGINALIZATION

The encoding uses logarithmic functions, which require that all retained leaf values and query outputs are positive. We remove zero-weight edges and floor discrete leaf masses at 10^{-12} before taking logarithms. If this flooring alters a fitted mass, it is necessary to account for its impact on the query within any error bounds related to native inference. A zero denominator renders the conditional probability undefined, while a zero numerator gives zero risk and a log-risk of $-\infty$, which the finite encoding cannot represent. Detailed information about the experimental policies can be found in Appendix B.6, and the error accounting is described in Appendix B.6.

Table 5 summarizes the representations for a retained leaf v with density or mass ℓ_v. After the stated floor, finite-domain leaves use exact log lookups; categorical indicators satisfy $\sum_{k\in\mathcal{K}_i} d_{ik} = 1$. A normalized marginalized leaf contributes $\log 1 = 0$, while continuous decision leaves in the scalar diagnostic use a bounded PWL approximation. For the campaign mixtures, each component log score sums the decision lookups and constants; the mixture root still requires log-sum-exp, encoded in the following subsections.

B.3 MAXIMUM ENCODING

After the considerations in the previous section, each retained weighted child has a finite log score t_j. Flooring, leaf interpolation, or approximation at earlier circuit nodes can change that score, so the bounds $t_j \in [\underline{t}_j, \bar{t}_j]$ must cover the encoded values. Let $\bar{t} = \max_j \bar{t}_j$ and introduce binary selectors d_j. The constraints

$$\sum_{j=1}^{m} d_j = 1, \qquad t_{\max} \geq t_j, \qquad t_{\max} \leq t_j + (\bar{t} - \underline{t}_j)(1 - d_j), \qquad d_j \in \{0, 1\} \tag{12}$$

Exactly one child is selected ($d_j = 1$). For that child, the third inequality becomes $t_{\max} \leq t_j$; together with $t_{\max} \geq t_j$ for every child, this makes t_j the maximum. For an unselected child, the big-M coefficient $(\bar{t} - \underline{t}_j)$ makes the third inequality nonrestrictive: the maximum cannot exceed $\bar{t}$, and that child's score cannot fall below $\underline{t}_j$. Thus the constraints enforce $t_{\max} = \max_j t_j$ regardless

Table 5: Log-space representations for retained leaves. The first four are exact under the stated positive-support condition; continuous log densities use a declared PWL approximation.

Leaf treatment	Representation of z_v
Binary decision	$(1 - x_i) \log \ell_v(0) + x_i \log \ell_v(1)$
Categorical decision	$\sum_{k\in\mathcal{K}_i} d_{ik} \log \ell_v(k)$
Fixed label $Y = \bar{y}$	$\log \ell_v(\bar{y})$
Marginalized variable	$\log 1 = 0$
Continuous variable	PWL approximation of $\log \ell_v(x_i)$

of the optimization objective. The lower inequalities extend the branch-selection formulation of Němeček et al. (2025, Section 4), which invokes likelihood maximization for tightness. Replacing log-sum-exp by this maximum satisfies

$$0 \leq \log\Big(\textstyle\sum_{j=1}^{m} e^{t_j}\Big) - t_{\max} \leq \log m, \tag{13}$$

so max approximation underestimates the sum-node output, with the upper bound attained by equal contributions. For retained finite scores, the gap vanishes only when $m = 1$; with multiple children, every e^{t_j} is positive and the gap is strictly positive. If zero-valued contributions are allowed before finite-log pruning, equality instead requires only one positive contribution. Marginalization can make several branches contribute even in deterministic circuits (Choi et al., 2020, Section 8.2). Our construction instead uses $t_{\max}$ to shift the inputs and retains the remaining contributions through the PWL correction developed next.

B.4 Piecewise-Linear Exponential

With $t_{\max}$ encoded as in Appendix B.3, the shifted inputs $u_j = t_j - t_{\max}$ are nonpositive, and at least one equals zero. To approximate their exponential values, we use piecewise-linear (PWL) interpolation: for a function $f : \mathbb{R} \to \mathbb{R}$, we evaluate it at ordered points $a_0 < \cdots < a_R$ (called knots) and connect neighboring points on its graph with straight lines. This defines the function $\widehat{f}$. For an input u, the encoding selects a segment containing u and interpolates between its endpoints to obtain $\omega = \widehat{f}(u)$. We represent the selection with binary variables b_r:

$$\textstyle\sum_{r=1}^{R} b_r = 1, \qquad b_r \in \{0,1\}, \quad r = 1,\ldots,R. \tag{14a}$$

We then use weights α_r and β_r to describe the position between the left and right endpoints. Linking these weights to b_r makes them sum to one on the selected segment and zero on every other segment:

$$\alpha_r + \beta_r = b_r, \qquad \alpha_r, \beta_r \geq 0, \quad r = 1,\ldots,R. \tag{14b}$$

The weights must reproduce the input u from the endpoint locations a_{r-1} and a_r:

$$u = \textstyle\sum_{r=1}^{R}(a_{r-1}\alpha_r + a_r\beta_r). \tag{14c}$$

Applying those same weights to the endpoint values $f(a_{r-1})$ and $f(a_r)$ places the output on the line joining them:

$$\omega = \textstyle\sum_{r=1}^{R}\big(f(a_{r-1})\alpha_r + f(a_r)\beta_r\big). \tag{14d}$$

Together, these constraints enforce $\omega = \widehat{f}(u)$ using only linear expressions, since the knot locations and function values are fixed coefficients. To bound the exponential inputs, let $\underline{t} = \min_j \underline{t}_j$ be the smallest child-score lower bound. The maximum score cannot exceed $\overline{t}$ from Appendix B.3, so every shifted input $u_j = t_j - t_{\max}$ lies in $[-\overline{B}, 0]$, where $\overline{B} = \overline{t} - \underline{t}$. We construct $\widehat{E}$ on this interval, with a knot at zero so that $\widehat{E}(0) = 1$.

The constraints in Equation 14 give the explicit disaggregated convex-combination (DCC) MILP formulation used in our derivation (Vielma, 2015): each possible segment has a binary selector, and weights interpolate along the selected segment. Repeating this construction across the circuit introduces a binary for every segment of every PWL term. For computational convenience, the campaign implementation instead supplied each interpolation graph through Gurobi's special ordered set of type 2 (SOS2) interface, which permits positive weights at no more than two neighboring knots. With identical knots, endpoint values, and input domains, the DCC formulation and SOS2 interface represent the same PWL graph at feasible solutions; their extended formulations and continuous relaxations need not coincide. The solver may retain or reformulate the SOS2 sets according to its settings, and we make no claim that this implementation choice yields a stronger relaxation.

B.5 Piecewise-Linear Logarithm

Let Q represent the sum of the approximate exponential values. At least one of the shifted inputs u_j is equal to zero, and the chosen exponential curve meets the conditions $\widehat{E}(0) = 1$ and $0 \leq \widehat{E}(u) \leq 1$ within its encoded domain. As a result, one term of Q equals one, while the others lie within the

interval $[0, 1]$. This implies that Q falls within the range $[1, m]$. Next, we construct a second PWL function $\widehat{H}$ over this interval to approximate $\log Q$. We then add $t_{\max}$ to obtain the approximate logarithmic output $\widehat{z}_v$ for the sum node. By utilizing the constraints presented in Equation 14 to encode both PWL functions, we achieve the desired representation

$$e_j = \widehat{E}(u_j), \qquad Q = \textstyle\sum_{j=1}^{m} e_j, \qquad \widehat{z}_v = t_{\max} + \widehat{H}(Q). \tag{15}$$

The guarded curve used in the campaigns and its approximation-error accounting are described in Appendix B.6.

B.6 Numerical Safeguards and Bounds

We use a guarded PWL exponential function for values of $u_j \leq 0$, where e^{u_j} reflects each child's contribution relative to the largest contribution. The guard function replaces this quantity with zero when $e^{u_j} \leq 10^{-3}$, which means we do not need detailed interpolation for the long negative tail. A linear segment of width 10^{-3} in u_j reconnects to the base curve. The complete saved curve extends over $u_j \in [-500, 0]$, and only the parts necessary at each node may be retained during compilation. Base exponential knots are spaced logarithmically based on input magnitude, while logarithmic knots encompass the sum of shifted contributions.

For a fixed fitted SPN, let $z_v(x)$ denote its native log output at node v and $\widehat{z}_v(x)$ the output from exactly evaluating the encoded node. We first bound the approximation error at a sum node, then show how local errors propagate through the circuit over the decision domain $\mathcal{X}$. The sum-node bounds include the guarded curve, while any modified leaves contribute their own error bounds; all encoded inputs must lie within their approximation domains. The reported campaigns used the nominal PWL constraint without these margins. However, the derivation shows how local approximation errors propagate to the query roots and how validated bounds could support a conservative constraint for the fitted SPN.

B.6.1 Local error at a sum node

For fixed weighted child scores $t = (t_j)_{j=1}^{m_v}$ at sum node v, the construction in Appendices B.3–B.5 gives the native and encoded outputs

$$F_v(t) = t_{\max} + \log\Big(\textstyle\sum_{j=1}^{m_v} e^{u_j}\Big), \qquad \widehat{F}_v(t) = t_{\max} + \widehat{H}_v(Q). \tag{16}$$

For each shifted input u_j, assume $0 \leq \widehat{E}_v(u_j) \leq 1$ and $\widehat{E}_v(0) = 1$. Consider that the shifted-input range $[-\overline{B}_v, 0]$ covers the encoded node. We then define the errors over the full exponential and logarithm curves, including any guard:

$$\delta_{E,v} = \sup_{u \in [-\overline{B}_v, 0]} |\widehat{E}_v(u) - e^u|, \qquad \delta_{H,v} = \sup_{Q \in [1, m_v]} |\widehat{H}_v(Q) - \log Q|. \tag{17}$$

One shifted input is zero, contributing one to both $\sum_j e^{u_j}$ and Q; all other terms lie in $[0, 1]$. Both sums therefore lie in $[1, m_v]$, where the logarithm is 1-Lipschitz. The m_v exponential errors and the logarithm-table error in Equation 17 thus give

$$|\widehat{F}_v(t) - F_v(t)| \leq \eta_v, \qquad \eta_v = m_v \delta_{E,v} + \delta_{H,v}. \tag{18}$$

To verify Equation 18, cancel the common $t_{\max}$ and add and subtract $\log Q$. This separates the logarithm-table error from the error in the summed exponential terms:

$$\begin{aligned} |\widehat{F}_v(t) - F_v(t)| &= \Big|\widehat{H}_v(Q) - \log \textstyle\sum_j e^{u_j}\Big| \\ &\leq |\widehat{H}_v(Q) - \log Q| + \Big|\log Q - \log \textstyle\sum_j e^{u_j}\Big| \\ &\leq \delta_{H,v} + \Big|Q - \textstyle\sum_j e^{u_j}\Big| \\ &= \delta_{H,v} + \Big|\textstyle\sum_j \big(\widehat{E}_v(u_j) - e^{u_j}\big)\Big| \\ &\leq \delta_{H,v} + \textstyle\sum_j |\widehat{E}_v(u_j) - e^{u_j}| \\ &\leq \delta_{H,v} + m_v \delta_{E,v}. \end{aligned} \tag{19}$$

The proof holds the child scores fixed, isolating the approximation at this sum node. For ordinary linear interpolation, the error is bounded by one eighth of the squared segment width times the largest absolute second derivative. Since $|(e^u)''| \le 1$ for $u \le 0$ and $|(\log q)''| \le 1$ for $q \ge 1$, the ordinary exponential segments and the logarithm table satisfy

$$|\widehat{E}_v(u) - e^u| \le h_{E,v}^2/8, \qquad \delta_{H,v} \le h_{H,v}^2/8.$$

Here $h_{E,v}$ and $h_{H,v}$ are the largest segment widths of the respective ordinary tables. The first inequality alone does not bound $\delta_{E,v}$: that quantity also includes error from the exponential guard's plateau and transition.

B.6.2 Propagation through the circuit

Start with a uniform log-error bound $|\widehat{z}_\ell(x) - z_\ell(x)| \le \Delta_\ell$ for each retained leaf ℓ. An exactly represented leaf has $\Delta_\ell = 0$; this includes normalized leaves replaced by one under marginalization. These leaf bounds initialize a bottom-up recursion: when processing an internal node v, Δ_j is the bound already established for child j, whether a leaf or another internal node.

$$\Delta_v = \begin{cases} \sum_{j \in \mathrm{ch}(v)} \Delta_j, & v \text{ is a product node,} \\ \max_{j \in \mathrm{ch}(v)} \Delta_j + \eta_v, & v \text{ is a sum node.} \end{cases} \tag{20}$$

We now justify the two cases in Equation 20. At a product node, the exact and encoded log outputs are sums of their respective child outputs, so

$$|\widehat{z}_v - z_v| \le \textstyle\sum_{j \in \mathrm{ch}(v)} |\widehat{z}_j - z_j| \le \sum_{j \in \mathrm{ch}(v)} \Delta_j. \tag{21}$$

At a sum node, the exact log-sum-exp output changes by no more than the largest change in a child score: $|F_v(a) - F_v(b)| \le \max_j |a_j - b_j|$ for any child-score vectors a and b. Because the weights are unchanged, $t_j = \log \lambda_{vj} + z_j$ and $\widehat{t}_j = \log \lambda_{vj} + \widehat{z}_j$, so each contribution changes by at most Δ_j. Combining this with the local encoding error η_v gives

$$\begin{aligned} |\widehat{z}_v - z_v| &= |\widehat{F}_v\big(\widehat{t}\big) - F_v(t)| \\ &\le |\widehat{F}_v\big(\widehat{t}\big) - F_v\big(\widehat{t}\big)| + |F_v\big(\widehat{t}\big) - F_v(t)| \\ &\le \eta_v + \max_{j \in \mathrm{ch}(v)} \Delta_j. \end{aligned} \tag{22}$$

Starting from the leaf bounds, these cases establish by induction that $|\widehat{z}_v(x) - z_v(x)| \le \Delta_v$ throughout $\mathcal{X}$, including at the violation numerator and marginalized denominator roots. Shared subgraphs do not invalidate the bounds; no independence of approximation errors is assumed. For a sum node with one child and an exact encoding, $\eta_v = 0$.

B.7 Complete Chance-Constraint Formulation

The chance constraint in Equation 9 is formulated using two different encodings of the fitted SPN from each experiment. Both encodings use the same decision variable x but employ separate auxiliary variables to marginalize Ξ. However, only the numerator is constrained to ensure that $Y = 1$. Each encoding enforces equalities for leaves, products, maxima, and PWL functions, as described in Appendices B.2 to B.5. The outputs of their queries, denoted as $\widehat{z}_1$ and $\widehat{z}_{\mathrm{marg}}$, satisfy the condition $\widehat{z}_1 - \widehat{z}_{\mathrm{marg}} \le \log \epsilon$, along with the deterministic decision restrictions defined by the problem.

Let z_1 and z_{marg} be the native fitted-SPN log outputs. Suppose Δ_1 and Δ_{marg} bound the errors of their encoded outputs for every $x \in \mathcal{X}$, as derived in Appendix B.6. The native log violation probability then satisfies the following upper bound:

$$z_1 - z_{\mathrm{marg}} \le \widehat{z}_1 - \widehat{z}_{\mathrm{marg}} + \Delta_1 + \Delta_{\mathrm{marg}}. \tag{23}$$

Consequently, the following stronger encoded constraint is sufficient for feasibility under the fitted SPN:

$$\widehat{z}_1 - \widehat{z}_{\mathrm{marg}} + \Delta_1 + \Delta_{\mathrm{marg}} \le \log \epsilon. \tag{24}$$

The same reasoning applies to a success query that fixes $Y = 0$. Let z_0 be its native log output and Δ_0 its encoding-error bound. The following constraint ensures a fitted-SPN success probability of at least $1 - \epsilon$.

$$\widehat{z}_0 - \widehat{z}_{\mathrm{marg}} - \Delta_0 - \Delta_{\mathrm{marg}} \ge \log(1 - \epsilon). \tag{25}$$

Both queries are encoded separately, so their approximate probabilities need not sum to one.

Applying these constraints would require error bounds valid for every decision in $\mathcal{X}$, including any changes to the leaves, and allowances for numerical solver error. The reported campaigns did not validate or impose such margins; they used the nominal PWL constraint. Even a validated margin would certify feasibility only under the fitted SPN, not the unknown true distribution.

C Experimental Protocol and Assessment

The knapsack and newsvendor campaigns each utilize ten datasets of a single, fixed problem instance. Every legal decision is represented in each training bank (see Table 6). The SAA scenario sets are nested within their corresponding TPM training uncertainty draws.

An SAA draw can be reused across different decisions; however, a TPM row is linked to one specific decision, draw, and outcome, which means that having equal counts does not necessarily indicate equal information.

Only the explicitly tested SAA formulations are compared. Additionally, the data size, encoding, and risk-limit studies reuse datasets or fits instead of incorporating independent replications.

Solvers utilize designated data, while population references evaluate returned decisions only after campaign choices have been finalized. Analytical information used to create the knapsack benchmark and scalar diagnostic is documented in Appendices E.1 and D.2. The newsvendor population reference is outlined in Appendix F.2.

Fixed-decision panels assess learned probability and encoding errors, and solver bounds and certificates pertain to each optimized formulation. The knapsack assessment weights all 1,024 decisions, whereas the newsvendor assessment employs designated order panels with empirical validation targets, making their mean absolute errors (MAEs) and false-safe rates not directly comparable.

Knapsack PWL-33 underwent the same fit resolution checks, while the newsvendor retained PWL-9 (see Appendix F.3.6).

C.1 Model and Training-Recipe Selection

We explored model structures and training hyperparameters before fixing the recipes used in the main experiments. Candidates included fixed SPN structures, LearnSPN, and Prometheus. Early hyperparameter screening used L9 designs: nine balanced combinations of three-level factors, allowing a small set of fits to explore settings such as component count and learning rate. Promising recipes received further fits with additional initializations, followed by targeted architecture and capacity sweeps guided by validation performance.

This development process led to the fixed 1,024-component mixtures used for knapsack and newsvendor, trained by joint NLL minimization. We then held the architecture and training recipe fixed across the datasets, selecting each fitted model through validation. The final checkpoint criteria differ by problem: joint validation NLL for knapsack and conditional probability error, with joint NLL as a tie-breaker, for newsvendor (Appendices E.3.5 and F.3.6). The scalar diagnostic uses its separate exploratory LearnSPN selection procedure (Appendix D.2). Table 7 collects the final knapsack and newsvendor training settings; Table 9 reports those of the scalar diagnostic.

Table 6: Primary campaign protocol. Training banks cover all legal decisions; the newsvendor validation panels reuse uncertainty draws.

Property	Knapsack	Newsvendor
Decision variables	10 binary	5 integer orders + activations
Legal decisions	1,024	6,039,420
Main risk limit	10%	5%
Datasets	10	10
Labeled training rows	3,686,400	30,197,100
Joint-NLL validation rows	409,600	20,000
Risk-checkpoint panel	Not used	512 orders / 100k shared draws
Components	1,024	1,024
Fitting objective	Joint NLL	Joint NLL
Checkpoint criterion	Joint validation NLL	Empirical risk MAE
Selected encoding	PWL-33	PWL-9
SAA risk scenarios	1k / 10k / 60k / 200k	1k / 10k / 60k / 200k
Profit objective	Deterministic	Common full-bank sample average

Table 7: Final SPN training recipes for the main experiments.

	Knapsack	Newsvendor
Model and initialization		
Architecture	One sum over 1,024 product nodes	One sum over 1,024 product nodes
Decision leaves	Bernoulli	Categorical on $\{0, \ldots, U_i\}$
Uncertainty	Gaussian	Gaussian
Label leaves	Bernoulli	Bernoulli
Uncertainty scaling	Mean and standard deviation	Divide demand coordinate i by U_i
Initialization	Training-row centers; equal mixture weights	20 full-data Lloyd iterations; standardized clustering coordinates
Gaussian scale floor	0.05 after standardization	$1/(\sqrt{12}U_i)$ after scaling
Parameter fitting		
Optimizer	Adam	Adam
Precision	float64	float64
Updates	4,000	3,000
Effective batch size	131,072	256,000
Cosine learning rate	$0.01 \to 0.0001$	$0.003 \to 10^{-5}$
Adam momentum factors	$(0.9, 0.999)$	$(0.8, 0.998)$
Adam numerical stabilizer	10^{-8}	$10^{-8}/\sqrt{2}$
Gradient-norm clipping	10	10
Weight decay	0	0
Checkpoint selection		
Validation frequency	Initially, then every 20 updates	Initially, then every 25 updates
Ranking	Lowest joint validation NLL	Lowest equally weighted bin MAE

C.2 Shared Solver Conventions

Table 8 outlines the shared and method-specific solver settings. Compilation and qualification processes are treated separately from the allowance for search time; any minor overshoots in the time limit are recorded. During time-limited runs, the reported metrics include the incumbent solution, the bound, and the gap. If a run does not yield an incumbent solution, no decision will be returned. Additionally, any failures in callback validation will render the solution invalid. Case-specific initialization, qualification, and strengthening details can be found in Appendices E.4 and F.4. Because some solver settings and implementation mechanisms are method-specific, elapsed-time and search comparisons describe the tested implementations rather than a solver-neutral ranking of the underlying approaches.

For the integer-profit knapsack, an absolute bound gap below one would certify the optimal objective value. The uncertified SAA-60k/200k runs end with absolute gaps of 29–34, so the zero-gap setting does not alter their reported certification status.

C.3 Resources and Computational Accounting

The campaigns utilized two Intel Xeon Gold 5218R CPUs (40 cores, 80 threads), 503 GiB of usable RAM, and RTX A6000 GPUs (48 GiB each, supporting float64 fitting). The timings reported reflect these runs rather than isolated benchmarks.

The reported totals cover four stages: data and model preparation, model fitting, compilation and qualification, and initialization and search. Population assessment is reported separately and is not included in these totals. For the knapsack PWL comparisons, the totals include the preparation and fitting costs of the parent data reused by each comparison. For newsvendor, the totals also include the TPM fitting time and an allocated share of preparation and experimental-design work that is common across methods. The method-specific execution time includes compilation, search, rebuilding, and callback execution.

In Table 31, stage medians are reported alongside medians of the per-run accounted totals. These quantities need not add up because shared newsvendor preparation and design costs, as well as

Table 8: Gurobi settings for the primary campaigns. Method-specific choices are shown separately; settings not listed here or in the case-specific appendices use their saved defaults.

Both methods	
Gurobi version	13.0.2
Threads; seed	10; 0
Time allowance	1,800 s (initialization + search)
`FeasibilityTol`	10^{-8}
`IntFeasTol`	10^{-8}
`MIPGap`	0
`MIPGapAbs`	0
Method-specific search	
TPM `MIPFocus`	3
Newsvendor SAA `MIPFocus`	1
TPM only	
`NumericFocus`	3
`OptimalityTol`	10^{-8}
`DualReductions`	0
`PreCrush`	1
`Presolve`	automatic
`PreSOS2BigM`	1
`PreSOS2Encoding`	3

some execution overhead, are included in the totals but do not have separate stage columns. These allocations describe the accounting convention used for this campaign; they are not estimates of standalone deployment cost. Formulation dimensions are measured before presolve and cuts, and peak memory is either reported by stage or unavailable.

D SCALAR PROBABILITY DIAGNOSTIC

D.1 EXPERIMENTAL SETUP

Our first test of the method consisted of a scalar problem that can be seen as a single-product version of the newsvendor problem with a service requirement from Appendix F.1.1. Specifically, it defines no shortage allowance ($L = 0$) and just one item (order quantity $x = q_1$). Under Gaussian demand, the shortage probability is known analytically, providing a controlled comparison of learned probabilities and their encodings. In this case, we used the LearnSPN method Gens & Pedro (2013) to build the SPN model. The learning configuration was selected using analytical probability errors, making this an exploratory diagnostic (Appendix D.2). Sampling and fixed-decision solver checks are detailed in Appendix D.1.1 and Appendix D.4.

We set the order quantity to $x \in [40, 160]$, uncertain demand as $\Xi \sim \mathcal{N}(100, 20^2)$, and the label as $Y = [\![\Xi > x]\!]$. The analytical violation probability is therefore

$$r^\star(x) = 1 - \Phi\left(\tfrac{x-100}{20}\right), \tag{26}$$

where Φ is the standard normal CDF.

D.1.1 SAMPLING DESIGN

To construct observations (x, ξ, y), where ξ is the realized demand, we divide $[40, 160]$ into 120 stratified intervals of width one. We allocate observations approximately equally across these intervals, sample order quantities uniformly within each interval, and draw demand independently. Training, validation, and held-out partitions use separate random streams. Both safe and violating observations are retained.

D.2 SPN CONSTRUCTION

For this scalar diagnostic, LearnSPN constructs the circuit from data rather than requiring a fixed mixture architecture. It recursively separates approximately independent groups of variables into product nodes and clusters observations into sum nodes, allowing different mixture components to capture different patterns in the data. When splitting stops, Gaussian leaves model the continuous variables and Bernoulli leaves model the violation label. Table 9 reports the configuration.

The minimum observation-slice size controls when this recursion stops. We select it using analytical probability errors at the 120 stratum midpoints, making this an exploratory diagnostic with access to the generating law. The selected model is then frozen for compilation; two additional fitting seeds assess sensitivity without replacing it.

Table 9: Scalar sampling and LearnSPN settings. The selected observation-slice size is reused for compilation.

Setting	Value
Training / validation / held-out rows	20,000 / 10,000 / 20,000
Decision strata	120 strata of width 1
Minimum observation slice	50, 100, 200; selected: 50
Minimum feature slice	2
Output channels	1
Row clustering	k-means, $k = 2$
Feature partitioning	RDC, threshold 0.3
Partition seeds (training / validation / held-out)	2026082801 / 2026082802 / 2026082803
Canonical fitting seed	2026082811
Additional fitting seeds	2026082812, 2026082813

D.3 ENCODING SELECTION

This exploratory comparison examines the probability error introduced by max approximation and the additional formulation size required by PWL. Max approximation keeps only the largest

Table 10: Scalar PWL selection and encoding size. PWL-5/9/17 use 5/9/17 base interpolation knots, defining 4/8/16 segments. Errors are relative to native SPN probabilities; PWL-9 is the selected resolution. Counts cover only the encoding; memory covers compiler diagnostics, excluding fitting.

	Validation error		Encoding size			Peak memory
Encoding	Max.	Mean	Continuous	Binary	Constraints	(MiB)
Max	–	–	29,421	4,900	17,157	716.3
PWL-5	0.066946	0.022866	166,509	24,484	53,877	720.4
PWL-9	0.058337	0.022966	396,621	39,172	68,565	787.8
PWL-17	0.058809	0.022854	1,209,357	68,548	97,941	1200.6

weighted child contribution at each sum node; PWL retains the other contributions through an approximation whose resolution can be increased. We encode the same fitted SPN using max approximation and PWL-5/9/17, keeping the leaf approximations fixed to isolate changes in the sum-node encoding.

Native SPN inference serves as a benchmark for encoding fidelity, focusing on comparison rather than learning error. We determine PWL resolution based on the largest absolute probability error at the 120 stratum midpoints, resolving ties with the mean error, and then select a lower resolution within a tolerance of 10^{-12}. This results in choosing PWL-9 without relying on analytical probabilities or held-out observations. As shown in Table 10, PWL-17 increases complexity but only marginally improves mean error while worsening maximum error. Refinement does not guarantee a monotonic improvement in conditional queries due to persistent leaf errors and varying cancellations of numerator and denominator errors. Table 11 compares the max approximation and selected PWL encoding across a separate grid of 121 integer quantities.

For reproducibility, the decision interval is normalized to $[0, 1]$. Gaussian leaf log densities are interpolated over five equally spaced knots using the logarithmic disaggregated convex-combination (DLOG) formulation. Each marginal contains 613 continuous leaves; sum nodes have at most two children and use the disaggregated convex-combination (DCC) formulation for their PWL terms. PWL-5/9/17 denote base knot counts. We followed the safeguards in Appendix B.6. These settings belong to this diagnostic and differ from those of the main optimization experiments.

D.4 Solver Consistency Checks

To check the solver representation, we fix x to 40, 100, and 160 for each of the four encodings and solve the attached constraints with HiGHS 1.15.1, without an objective. Each check starts from a complete assignment computed by direct evaluation of the approximate circuit. All twelve complete assignments are accepted, and the returned probabilities agree with direct evaluation while constraint residuals remain below 10^{-6}. These checks establish agreement at the three tested decisions. The graph equalities in Appendix B define the represented value; the checks do not establish accuracy throughout the interval or optimize a free decision.

D.5 Additional Results

Figure 4 compares native inference with max and PWL-9, alongside held-out observations. In probability units, native inference has MAE 0.01273 against the analytical probabilities. PWL-9 reduces compilation MAE relative to native inference from 0.24281 for max to 0.02273. Compilation error remains appreciable and has both signs, so PWL does not provide a conservative risk bound. The held-out observations supply the empirical rates in Figure 4. Each stratum contains 166 or 167 observations, and its interval is a pointwise 95% Wilson interval for the observed violation frequency. These intervals describe sampling uncertainty in the binwise rates, not uncertainty in the fitted SPN. Model predictions are evaluated at fixed quantities, whereas each empirical rate averages over quantities sampled throughout its stratum.

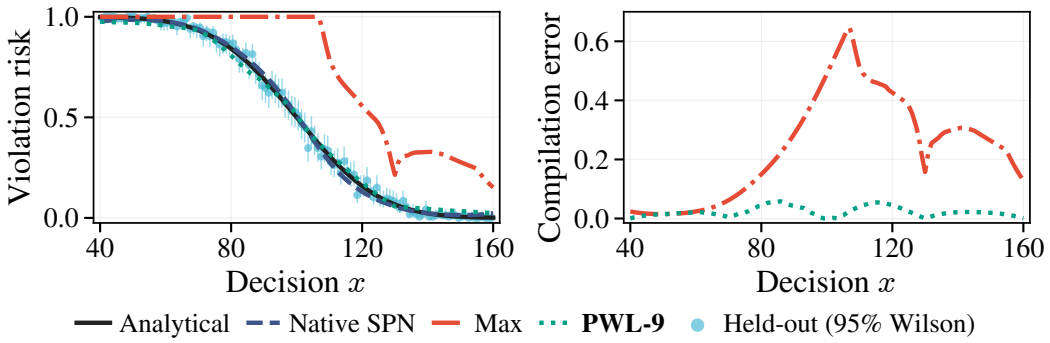

Figure 4: Fixed-decision scalar diagnostic: shortage probabilities (left) and absolute compilation errors relative to native SPN inference (right). Points show held-out frequencies; bars show pointwise 95% Wilson intervals.

Table 11: Scalar errors on 121 quantities. Total error is relative to analytical probability; compilation error is relative to native inference. The last column counts adjacent probability increases greater than 10^{-12}.

Estimator	Total MAE	Total max.	Compiler MAE	Compiler max.	Upward
Native	0.012728	0.035275	0.000000	0.000000	19
Max	0.240901	0.625868	0.242807	0.645529	11
PWL-9	0.017934	0.041575	0.022735	0.058087	0

E STOCHASTIC KNAPSACK

E.1 INSTANCE CONSTRUCTION

E.1.1 PROBLEM FORMULATION

As a simple example of a SP problem with a chance constraint, consider a modification of the classical knapsack problem. In the deterministic problem, $x_i = 1$ selects item i, whose value is c_i and fixed weight is $\bar{w}_i$; the total weight cannot exceed capacity W. The stochastic version replaces the fixed weights by a random vector Ξ and limits the probability of overloading the knapsack to ϵ, while the values remain deterministic:

$$\begin{aligned} &\max_x c^\top x && \max_x c^\top x \\ &\text{s.t. } \bar{w}^\top x \le W \quad \rightarrow \quad && \text{s.t. } \mathbb{P}\left(\Xi^\top x \le W\right) \ge 1-\epsilon \\ &x \in \{0,1\}^n && x \in \{0,1\}^n, \end{aligned}$$

Here $G(x,\xi) = \xi^\top x - W$, so a total weight equal to capacity is safe. Evaluating this probability and representing the resulting constraint in a tractable way can be difficult, particularly when the weight distribution is accessible only through observations.

E.1.2 INSTANCE AND POPULATION REFERENCE

For the problem's instance we define ten items with independent, untruncated Gaussian item weights $\Xi_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and risk limit $\epsilon = 0.10$. To avoid crafting a specific instance, we randomly generate the problem parameters rather than choosing individual weights and profits by hand. Mean weights are drawn uniformly from the integers 10 through 50; standard deviations are random fractions $\gamma_i \in [0.10, 0.20]$ of these means, and profits are random multiples $v_i \in [0.8, 1.2]$ of them, with both factors drawn uniformly. Thus $\sigma_i = \gamma_i \mu_i$, while c_i is $v_i \mu_i$ rounded to a positive integer (half-even rounding, with a minimum of one). Capacity is half the sum of mean weights, $W = 0.5 \sum_i \mu_i$. Table 12 reports the resulting parameters.

Before searching, we specified criteria for a diagnostic instance: a nondegenerate fraction of feasible decisions, a unique optimum close to the risk limit, and a nearby infeasible decision with higher profit. These criteria make errors near the feasibility boundary consequential for decision quality. We then scanned seeds in order within the range 0–999 and accepted the first qualifying instance, seed 34, which gives $W = 151$. The seed was selected by these predefined properties, not by TPM or SAA performance. The resulting instance is a deliberately selected diagnostic case, not a representative random draw.

E.2 EXACT AND SAA BASELINES

E.2.1 ANALYTICAL ASSESSMENT AND EXACT OPTIMUM

Independent Gaussian item weights make the total selected weight Gaussian, enabling analytical evaluation of the problem. For a nonempty decision, its mean is $\mu(x) = \sum_i \mu_i x_i$ and its standard

Table 12: Canonical ten-item generator output. Profit is deterministic, μ_i is mean consumption, σ_i its standard deviation, and $\gamma_i = \sigma_i/\mu_i$. Displayed decimals are rounded only for reporting.

Item	Profit c_i	Mean μ_i	S.d. σ_i	γ_i
1	35	43	5.8354	0.1357
2	28	24	4.5567	0.1899
3	27	33	6.4501	0.1955
4	30	29	3.8893	0.1341
5	18	19	2.0833	0.1096
6	11	10	1.7764	0.1776
7	14	15	2.7018	0.1801
8	37	45	7.6723	0.1705
9	35	34	6.5748	0.1934
10	48	50	6.5500	0.1310

deviation is $\sigma(x) = \sqrt{\sum_i \sigma_i^2 x_i}$, giving

$$r^\star(x) = 1 - \Phi\Big(\tfrac{W-\mu(x)}{\sigma(x)}\Big), \tag{27}$$

while the empty decision has risk zero. We evaluate all 1,024 decisions and select the highest-profit feasible one to obtain the exact reference optimum; 405 decisions are feasible. The unique optimum has profit 140 and risk 9.527%; a profit-141 competitor has risk 10.25%.

E.2.2 Exact Gaussian reformulation

The known Gaussian distribution also allows us to solve the chance-constrained problem directly, providing an independent check of the enumerated optimum. The chance constraint requires the mean selected weight plus a safety margin to remain within capacity. For independent Gaussian weights and $0 < \epsilon \le 0.5$, this gives the exact mixed-integer second-order cone formulation

$$\begin{aligned} \max_x \quad & c^\top x \\ \text{s.t.} \quad & \mu^\top x + \Phi^{-1}(1-\epsilon)\,\|(\sigma_1 x_1, \dots, \sigma_n x_n)\|_2 \le W, \\ & x \in \{0,1\}^n. \end{aligned} \tag{28}$$

Since $x_i^2 = x_i$, the norm equals the total weight's standard deviation $\sigma(x)$; the Gaussian quantile scales this variability to the required confidence level. Gurobi certifies profit 140, and the formulation's feasibility classification agrees with analytical evaluation at all 1,024 decisions. This reference exploits the known Gaussian distribution. In applications where uncertainty is accessible through observations or simulation but lacks a tractable analytical model, such a reformulation may be unavailable, motivating a learned approach.

E.2.3 Explicit SAA formulation

For the Gaussian knapsack, the explicit SAA risk constraint uses one violation indicator for each sampled consumption vector:

$$\begin{aligned} \max_{x,z} \quad & c^\top x \\ \text{s.t.} \quad & (\xi^{(s)})^\top x \le W + M_s z_s, && s = 1, \dots, N_{\mathrm{SAA}}, \\ & \sum_{s=1}^{N_{\mathrm{SAA}}} z_s \le \lfloor \epsilon N_{\mathrm{SAA}} \rfloor, && x_i, z_s \in \{0,1\}. \end{aligned} \tag{29}$$

Here $\xi^{(s)}$ contains the sampled item weights in scenario s. Setting $z_s = 1$ allows that scenario to exceed capacity, while the second constraint limits how many scenarios may do so. We choose $M_s = \max(0, \sum_i \max(\xi_i^{(s)}, 0) - W)$ to cover the largest possible overload in each scenario, including when some sampled weights are negative. We compare four scenario-set sizes, each larger set containing the smaller ones, using the solver settings in Appendix C.2.

E.3 Data Generation and SPN Training

E.3.1 Observations and decision allocation

For each dataset, we allocate observations equally across all 1,024 binary decisions, drawing 3,600 weight vectors per decision for training and 400 for validation. This gives 3,686,400 training rows and 409,600 validation rows (Table 6). Every row uses an independent weight draw; the same uncertainty samples are not reused across decisions. Separate reproducible PCG64 random streams generate the training data, validation data, and random row ordering for each dataset. We retain both safe and violating outcomes, including negative Gaussian weights, and treat equality with capacity as safe. For SAA, we shuffle the training weight vectors without using their labels, discard their associated decisions and labels, and take progressively larger prefixes so that each scenario set contains the smaller ones. Covering the complete decision space enables exhaustive probability and feasibility assessment, but does not test prediction at unseen decisions.

E.3.2 NESTED DATASETS FOR THE TRAINING-SIZE STUDY

For three datasets, we construct progressively larger training subsets with 10, 100 and 1,000 observations per decision, totaling 10,240, 102,400 and 1,024,000 rows. We use a fixed random ordering within each decision, so every larger subset contains the smaller ones; selection does not depend on weights or outcomes. The existing fits with 3,600 observations per decision provide the largest-data comparison. For each subset, we recompute standardization means and standard deviations from its safe observations and apply them to all observations, retaining both labels. The same 409,600 independent validation observations are used at every training size.

E.3.3 ARCHITECTURE AND MARGINALIZED QUERY

Following the model exploration described in Appendix C.1, we selected an SPN with a single sum node over $K = 1{,}024$ product components and fixed this architecture across datasets. Each component combines Bernoulli distributions for the decisions and violation label with Gaussian distributions for the uncertain weights. For $n = 10$ items, the resulting joint model is

$$p_\theta(x, \xi, y) = \sum_{k=1}^{K} \pi_k \left[\prod_{i=1}^{n} \mathrm{Bern}(x_i; a_{ki})\right] \left[\prod_{i=1}^{n} \mathcal{N}(\xi_i; m_{ki}, s_{ki}^2)\right] \mathrm{Bern}(y; b_k). \tag{30}$$

Here $k \in \{1, \ldots, K\}$ indexes the mixture components, π_k is the weight of component k, a_{ki} and b_k are its probabilities for $X_i = 1$ and $Y = 1$, and m_{ki} and s_{ki} are the mean and standard deviation of its Gaussian uncertainty leaf Ξ_i. All mixture weights and leaf parameters are learned.

This structure also gives a simple expression for the conditional violation probability. After marginalizing the uncertain weights, each component's violation probability is weighted by how well that component explains the chosen decision. The learned conditional risk $r_\theta(x) = \mathbb{P}_\theta(Y = 1 \mid X = x) = p_\theta(x, 1)/p_\theta(x)$ is therefore

$$r_\theta(x) = \textstyle\sum_k \rho_k(x) b_k, \qquad \rho_k(x) = \frac{\pi_k \prod_i \mathrm{Bern}(x_i; a_{ki})}{\sum_j \pi_j \prod_i \mathrm{Bern}(x_i; a_{ji})}. \tag{31}$$

$\rho_k(x)$ is the fitted mixture weight of component k conditional on decision x.

Each training label records the physical violation event, but the fitted product mixture does not enforce this deterministic relation throughout its support. Thus $r_\theta(x)$ can differ from the event probability obtained by integrating the model's fitted uncertainty conditional at x. The uncertainty observations influence fitting but are marginalized at optimization time.

E.3.4 JOINT-LIKELIHOOD TRAINING RECIPES

Table 7 is the compact summary of the final training recipe. This subsection adds only the knapsack-specific details needed to interpret and reproduce that summary. Knapsack fitting minimizes the joint NLL in Equation 8.

Initialization. The same initializer is used for every knapsack fit, with seed 1 fixed for reproducibility rather than selected by comparing seeds. The 1,024 component centers are placed on sampled training rows and receive equal mixture weights. At a center whose observed binary decision is x_i, its decision leaf starts with probability 0.85 on that observed bit and 0.15 on the other bit: when $x_i = 0$, the initial probability of $X_i = 1$ is 0.15; when $x_i = 1$, it is 0.85. These are starting values, not a decision rule; joint-NLL optimization learns the final probabilities. Gaussian scales start at one in standardized coordinates. Standardization statistics use safe training rows, but both labels remain in the likelihood.

Optimization exposure. Each fit processes 524,288,000 row presentations, including repeated passes. For the 10, 100, 1,000 and 3,600 observations-per-decision fits in the training-size study, this corresponds to 51,200, 5,120, 512 and approximately 142.22 effective epochs. Microbatch losses are divided by the fixed effective batch size; batches may cross epoch boundaries, gradients accumulate across microbatches, and one Adam update follows gradient clipping.

E.3.5 CHECKPOINT SELECTION

Knapsack evaluates full joint validation NLL initially and every 20 updates, retaining the earliest strictly best checkpoint. For comparisons across differently standardized subsets, raw-weight NLL

adds the Gaussian Jacobian term $\sum_i \log s_i$, with s_i the subset standard deviation; this correction does not change selection within a fit. Encoding-resolution selection is documented with the shared configuration conventions in Appendix F.3.6.

E.4 Compilation and Optimization Settings

The numerical policies in Appendix B.6 and solver settings in Appendix C.2 apply. The following material specifies the knapsack initialization, qualification and binary strengthening.

Knapsack qualifies the frozen query before search, selects a start from 128 seeded candidate decisions using the method's own feasibility test, and validates reported incumbents against that same query. Business binaries have branching priority 100. The native/encoded exhaustive assessments are diagnostics, not alternative optimizer returns.

E.4.1 Knapsack encoding qualification

The encoded success query is $p_\theta\left(x, 0\right) / p_\theta\left(x\right)$ with log bounds $[\log(0.9), 0]$. Native-inference agreement: direct conditional evaluation is compared with SPN marginalization at all 1,024 decisions, with tolerance 10^{-9}. Encoded-assignment checks: for every encoded-feasible decision, direct evaluation supplies a complete auxiliary assignment, and linear rows, bounds, integrality and SOS adjacency are checked with residual tolerance 10^{-6}. The graph equalities in Appendix B establish the represented function at a fixed decision; successful assignment checks validate its implementation on the tested decisions.

Returned-solution checks: solver output values at incumbents are compared with direct compiled-query evaluation; saved predictions then independently audit excluded decisions and generated faces. The published ten-dataset view reports passing audits with no encoded-feasible decision excluded. These finite-domain checks validate implementation and solver handoff, separately from the analytical validity argument in Appendix E.4.2. Population Gaussian risks enter the post-search assessment only; they do not guide these checks or callback exclusions.

E.4.2 Strengthening the compiled formulation

This section explains how static strengthening improves search while preserving the feasible set of the compiled knapsack model. The static rows and optional callback cuts refer to the saved PWL query, including its guarded exponential, fitted denominator and numerical policy; the full probability constraint remains in the model throughout search.

Query structure and static strengthening The following representation makes query bounds available for tightening without changing the encoded threshold. After marginalization, each query in the fitted mixture is a log-sum of affine component scores over binary decisions $x \in \{0,1\}^n$. For the knapsack queries, $\iota \in \{0, \mathrm{marg}\}$ fixes the success label or marginalizes it. Write $t_{\iota j}(x) = \beta_{\iota j} + a_{\iota j}^\top x$ and $M_\iota(x) = \max_j t_{\iota j}(x)$. Its encoded output is

$$\widehat{z}_\iota(x) = M_\iota(x) + \widehat{H}_\iota\left(\textstyle\sum_{j=1}^{m_\iota} \widehat{E}_\iota(t_{\iota j}(x) - M_\iota(x))\right). \tag{32}$$

Within each query, all components share the same exponential curve. The saved curves are nondecreasing, $0 \le \widehat{E}_\iota \le 1$, $\widehat{E}_\iota(0) = 1$, and the logarithm curve covers $[1, m_\iota]$. Here the numerator fixes $Y = 0$. Writing $\widehat{s}(x)$ for the encoded log-success ratio, the knapsack uses

$$\widehat{s}(x) = \widehat{z}_0(x) - \widehat{z}_{\mathrm{marg}}(x), \qquad \log(1-\epsilon) \le \widehat{s}(x) \le 0. \tag{33}$$

The lower bound imposes the risk limit, while the upper bound restricts the separately approximated success ratio to at most one. Static strengthening and callback cuts preserve the intersection of these two bounds. In the saved ten-dataset PWL-33 view, exhaustive query values exceed one at no decision, so the upper restriction removes none of the 1,024 decisions in these instances. Separately approximated success and violation queries need not give complementary encoded probabilities.

Before search, affine bounds and bounds on pairwise score differences tighten the maximum, shifted inputs and subsequent PWL operations. Restricting a table to a tighter interval preserves its existing interpolant: new interval endpoints take values from that curve, rather than reevaluating the exponential or logarithm. The knapsack implementation represents these curves with SOS2 interpolation.

Even with a fixed architecture and encoding policy, learned parameters can change the bounds and hence the retained PWL segments, so the precise formulation dimensions can differ across fits.

Three additional sets of constraints connect the auxiliary variables more closely to the decisions. First, for maximum selectors $d_{\iota j}$, let $\zeta_{\iota j i}$ be a continuous copy of decision coordinate x_i under component j. These copies satisfy

$$0 \le \zeta_{\iota j i} \le d_{\iota j}, \qquad \textstyle\sum_j \zeta_{\iota j i} = x_i, \qquad M_\iota = \textstyle\sum_j \left(\beta_{\iota j} d_{\iota j} + \sum_i a_{\iota j i} \zeta_{\iota j i}\right). \tag{34}$$

When a selector is one, its copies equal x and all other copies vanish, so this equality agrees with the selected maximum. Second, the selector of a winning component forces its exponential interpolation weight onto the endpoint $(0, 1)$; a component whose shifted upper bound is negative cannot win. Third, bound propagation with each individual decision fixed to zero and one gives bounds $[L^b_{\chi i}, U^b_{\chi i}]$ for an auxiliary variable χ, conditional on $x_i = b$. We impose

$$L^0_{\chi i} + (L^1_{\chi i} - L^0_{\chi i}) x_i \le \chi \le U^0_{\chi i} + (U^1_{\chi i} - U^0_{\chi i}) x_i. \tag{35}$$

For binary x_i, these inequalities recover the appropriate conditional bounds. They strengthen the relaxation without imposing a different probability threshold.

Mechanism: excluding decisions and groups The callback uses the compiled evaluator to exclude tested vertices and, when possible, larger binary faces; these cuts accelerate search without adding a new probability constraint. At a solved node relaxation, the callback rounds the decision coordinates to a binary candidate $\bar{x}$. It evaluates the saved compiled query only if $\|x^{\mathrm{LP}} - \bar{x}\|_1 < 1 - \tau$, with $\tau = 10^{-6}$, so that the corresponding exclusion inequality would cut the current relaxation point. If $\widehat{s}(\bar{x})$ lies outside the interval in Equation 33 by more than τ, the candidate can be excluded. These evaluations are cached; no exhaustive decision table or analytical population risk guides separation.

The callback then attempts to exclude a larger group. For a set of fixed coordinates I, define the binary face

$$\mathcal{F}(I, \bar{x}) = \{x \in \{0,1\}^n : x_i = \bar{x}_i \text{ for all } i \in I\}. \tag{36}$$

If an upper bound U_F on $\widehat{s}$ over this face satisfies $U_F < \log(1-\epsilon) - \tau$, every decision in the face violates the compiled success requirement. The valid exclusion inequality is

$$\textstyle\sum_{\substack{i \in I \\ \bar{x}_i = 0}} x_i + \sum_{\substack{i \in I \\ \bar{x}_i = 1}} (1 - x_i) \ge 1. \tag{37}$$

It requires at least one fixed coordinate to change and therefore excludes exactly that face among binary decisions. Starting with all coordinates fixed, the routine tests releasing each remaining coordinate and accepts the release with the smallest bound, provided the bound still proves infeasibility. It repeats until no further release is certified. If no face certificate is obtained, the callback retains the single-decision cut justified by direct evaluation. It submits these inequalities as user cuts; they are not lazy constraints defining the probability model.

Validity of the group bound The group certificate is valid because its bound is an upper bound on the implemented compiled success expression over the entire face. To bound both query outputs on a face, subtract a common affine reference $r(x)$ from every score in both queries. The implementation chooses the denominator component with largest score at $\bar{x}$. For each fixed x, subtracting $r(x)$ shifts the maximum by the same amount and leaves all relative inputs unchanged. Consequently it subtracts $r(x)$ from each encoded output and cancels in their difference, even with the guarded PWL curves.

Write the centered scores as $h_{\iota j}(x) = \gamma_{\iota j} + c_{\iota j}^\top x$. On the continuous box containing $\mathcal{F}(I, \bar{x})$, their exact affine extrema are

$$\begin{aligned} L_{\iota j} &= \gamma_{\iota j} + \sum_{i \in I} c_{\iota j i} \bar{x}_i + \sum_{i \notin I} \min(0, c_{\iota j i}), \\ U_{\iota j} &= \gamma_{\iota j} + \sum_{i \in I} c_{\iota j i} \bar{x}_i + \sum_{i \notin I} \max(0, c_{\iota j i}). \end{aligned} \tag{38}$$

Let $M_\iota^- = \max_j L_{\iota j}$ and $M_\iota^+ = \max_j U_{\iota j}$. Monotonicity of the saved exponential curve gives bounds on the sum of shifted contributions:

$$\begin{aligned} Q_\iota^- &= \max\left\{1, \sum_j \widehat{E}_\iota(L_{\iota j} - M_\iota^+)\right\}, \\ Q_\iota^+ &= \min\left\{m_\iota, \sum_j \widehat{E}_\iota\big(\min\{0, U_{\iota j} - M_\iota^-\}\big)\right\}. \end{aligned} \tag{39}$$

For interval endpoints below the table domain, the bound evaluator extends the guard's zero plateau; the actual encoded inputs must still lie in their declared domains. At least one relative input is zero and every contribution is at most one, which justifies intersecting the sum bounds with $[1, m_\iota]$. Monotonicity of the logarithm curve then yields

$$\begin{aligned} Z_\iota^- &= M_\iota^- + \widehat{H}_\iota(Q_\iota^-), \qquad Z_\iota^+ = M_\iota^+ + \widehat{H}_\iota(Q_\iota^+), \\ \widehat{s}(x) &\le U_F := Z_0^+ - Z_{\mathrm{marg}}^-, \qquad x \in \mathcal{F}(I, \bar{x}). \end{aligned} \tag{40}$$

Indeed, each centered maximum lies in $[M_\iota^-, M_\iota^+]$, each relative input lies between the arguments used in Equation 39, and the common reference cancels from the conditional query. Thus the exclusion test proves validity of Equation 37 in exact arithmetic. The argument bounds the actual interpolants, including their guard segments; it does not replace them with native SPN inference.

Numerical checks and failure policy. The numerical checks test that the strengthening and callback preserve the compiled feasible set on the campaign instances. The implementation expands affine intervals by 10^{-9}, expands the summed-contribution bounds by 10^{-8} before intersecting with $[1, m_\iota]$, and expands each output interval and the final conditional upper bound by 10^{-8}. The exclusion test additionally uses the 10^{-6} log-space buffer. These are fixed floating-point allowances, not a formally verified interval-arithmetic error bound or a statistical confidence margin. For the ten-item campaign, complete assignments of all encoded-feasible decisions are checked against the static model before search; after search, enumeration checks the recorded face bounds and verifies that no cut excludes an encoded-feasible decision. These checks validate the finite campaign instances and do not replace the bound derivation or establish numerical validity for arbitrary larger instances. At each incumbent, direct compiled evaluation is compared with the solver's output and threshold; an inconsistency or callback exception is recorded and terminates the run. Such a run fails validation rather than silently receiving a feasibility or optimality certificate.

Scope The scope qualification is therefore about the implemented encoding and tested instances, not about population feasibility. Single-decision exclusion requires binary decisions and a faithful evaluator of the compiled query. The group bound and static strengthening described here additionally exploit affine component scores and shared monotone PWL curves, as supplied by the knapsack mixture. They are not consequences of smoothness and decomposability alone and are not claimed for arbitrary deep circuits or continuous decision domains. The cuts preserve feasible binary decisions of the specified encoded model; they neither make its probabilities conservative relative to the fitted SPN nor guarantee population feasibility. Any solver optimality certificate remains a statement about that encoded model, subject to its numerical tolerances and validation checks. The optional-cut ablation retains the static strengthening and the full probability constraint, so it measures the additional search effect of the callback rather than that of the entire encoding.

E.5 Additional Results

E.5.1 Knapsack campaign

Native probabilities use log-sum-exp inference in the fitted mixture; encoded probabilities use a direct evaluator of the optimizer's saved PWL curves. Encoded knapsack risk is one minus encoded success. Probability MAE weights all 1,024 decisions equally, while boundary MAE uses population risks in $[0.5\epsilon, 1.5\epsilon]$. Feasibility classifications use the stated risk limit and saved comparison tolerance. Each false-safe or false-infeasible rate states its denominator; an empty denominator is undefined.

The IDs in Table 13 and Table 14 identify ten sampling replications of one knapsack instance. The tables retain all fifty primary decisions, including the PWL-33 amendment and all SAA time limits.

Table 13: Knapsack campaign, dataset blocks 0–4. Risk is population risk, reported as a percentage; solver gap is a percentage; F is Y/N population feasibility. Profit is the population value and bound is the optimized knapsack-profit bound. Gaps are rounded; certificates are counted separately.

ID	Method	Decision	Profit	Risk	F	Bound	Gap	Time (s)
0	SAA 1k	0101000011	141.00	10.251	N	141.00	0.00	3.7
0	SAA 10k	0101000011	141.00	10.251	N	141.00	0.00	520.2
0	SAA 60k	0101111010	136.00	1.952	Y	169.00	24.26	1,800.1
0	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.2
0	**TPM**	0101111010	136.00	1.952	Y	140.00	2.94	1,800.4
1	SAA 1k	0100110011	140.00	9.527	Y	140.00	0.00	2.0
1	SAA 10k	0101000011	141.00	10.251	N	147.00	4.26	1,800.1
1	SAA 60k	0101111010	136.00	1.952	Y	169.00	24.26	1,800.5
1	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.1
1	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	1,072.9
2	SAA 1k	0100110011	140.00	9.527	Y	140.00	0.00	2.1
2	SAA 10k	0100110011	140.00	9.527	Y	140.00	0.00	460.9
2	SAA 60k	0101111010	136.00	1.952	Y	169.00	24.26	1,800.4
2	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.4
2	**TPM**	0101101001	138.00	7.049	Y	138.00	0.00	936.3
3	SAA 1k	0101000011	141.00	10.251	N	141.00	0.00	2.7
3	SAA 10k	0100110011	140.00	9.527	Y	156.00	11.43	1,800.0
3	SAA 60k	0101111010	136.00	1.952	Y	169.00	24.26	1,800.6
3	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.1
3	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	991.3
4	SAA 1k	0101101001	138.00	7.049	Y	138.00	0.00	2.4
4	SAA 10k	0100110011	140.00	9.527	Y	140.00	0.00	544.8
4	SAA 60k	0101101001	138.00	7.049	Y	169.00	22.46	1,800.3
4	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.3
4	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	1,192.0

Exhaustive assessment gives population-best profit 140 in every dataset. The native model's best feasible profit is 140 in nine fits and 138 in dataset 2; the encoded model gives 140 in eight fits and 138 in datasets 2 and 6. Across the 10,240 decision–fit pairs, native and encoded feasibility classifications differ in 11. Relative to population feasibility, the encoded model accepts 48 infeasible pairs and rejects 36 feasible pairs. These disagreements describe learned and compiled feasible-set changes, before any solver-search outcome is considered. The returned solutions provide the separate search account. Dataset 0's TPM return has profit 136, bound 140 and a 2.94% gap, although its native and encoded reference optima both have profit 140; the better encoded-feasible decision establishes incomplete search as the source of this loss. The remaining TPM searches certify their encoded optima, so any remaining shortfall from population-best profit must be distinguished from solver gap and from feasible-set changes. The ten datasets are sampling replications, not ten problem instances or ten independently varied initialization seeds.

The joint summary and comparisons across both problems are collected in Appendix F.5. The following tables retain the knapsack records for each supporting analysis.

E.5.2 Supporting records

The following records support the diagnostics discussed in Appendix F.5; reused fits and native-only rows are identified in the captions.

Training-data and model-capacity. These records retain the training-size and capacity outcomes. The comparative interpretation is in Appendix F.5.4.

Encoding-fidelity. These same-fit resolution records support the comparative interpretation in Appendix F.5.5.

Optional cuts. These matched pairs support the comparative interpretation in Appendix F.5.6.

Table 14: Knapsack campaign, dataset blocks 5–9. Risk is population risk, reported as a percentage; solver gap is a percentage; F is Y/N population feasibility. Profit is the population value and bound is the optimized knapsack-profit bound. Gaps are rounded; certificates are counted separately.

ID	Method	Decision	Profit	Risk	F	Bound	Gap	Time (s)
5	SAA 1k	0101000011	141.00	10.251	N	141.00	0.00	5.6
5	SAA 10k	0100110011	140.00	9.527	Y	140.00	0.00	457.2
5	SAA 60k	0101111010	136.00	1.952	Y	169.00	24.26	1,800.3
5	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.2
5	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	1,213.7
6	SAA 1k	0101101001	138.00	7.049	Y	138.00	0.00	2.4
6	SAA 10k	0100110011	140.00	9.527	Y	140.00	0.00	1,002.4
6	SAA 60k	0101101001	138.00	7.049	Y	169.00	22.46	1,800.1
6	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.3
6	**TPM**	0101101001	138.00	7.049	Y	138.00	0.00	1,204.5
7	SAA 1k	0100110011	140.00	9.527	Y	140.00	0.00	8.8
7	SAA 10k	0101000011	141.00	10.251	N	141.00	0.00	435.3
7	SAA 60k	0100110011	140.00	9.527	Y	169.00	20.71	1,800.4
7	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.1
7	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	1,058.9
8	SAA 1k	0101000011	141.00	10.251	N	141.00	0.00	2.5
8	SAA 10k	0100110011	140.00	9.527	Y	140.00	0.00	564.8
8	SAA 60k	0100110011	140.00	9.527	Y	169.00	20.71	1,800.8
8	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.2
8	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	953.6
9	SAA 1k	0101000011	141.00	10.251	N	141.00	0.00	8.4
9	SAA 10k	0100110011	140.00	9.527	Y	154.00	10.00	1,800.1
9	SAA 60k	0100110011	140.00	9.527	Y	169.00	20.71	1,800.4
9	SAA 200k	0101111010	136.00	1.952	Y	170.00	25.00	1,800.2
9	**TPM**	0100110011	140.00	9.527	Y	140.00	0.00	956.9

Table 15: Knapsack training-size outcomes. An asterisk marks reused full-data endpoints; MAE and boundary MAE are percentage points, gap is percent, and F marks population feasibility. — denotes no returned optimization result; such cases are not classified as unsafe.

Rows	ID	MAE	Boundary	Profit	F	Gap	Time (s)
10,240	0	21.368	28.192	—	—	—	1,800.2
10,240	1	21.252	27.766	—	—	—	1,800.1
10,240	2	21.145	28.134	—	—	—	1,800.1
102,400	0	3.211	5.560	138.00	N	0.00	1,206.4
102,400	1	3.234	5.855	136.00	Y	1.47	1,800.1
102,400	2	3.158	6.013	136.00	Y	1.47	1,800.1
1,024,000	0	0.983	2.090	140.00	Y	0.00	1,046.7
1,024,000	1	0.948	1.819	140.00	Y	0.00	902.3
1,024,000	2	1.002	1.956	136.00	Y	2.94	1,800.1
3,686,400*	0	0.809	1.423	136.00	Y	2.94	1,800.4
3,686,400*	1	0.807	1.399	140.00	Y	0.00	1,072.9
3,686,400*	2	0.778	1.532	138.00	Y	0.00	936.3

Risk limit. These frozen-model risk-limit records support the comparative interpretation in Appendix F.5.7.

Table 16: Knapsack model-capacity study: native MAE in percentage points; FS and FI are false-safe and false-infeasible counts over all decisions; fit time is in seconds. K1024 reuses the primary fit, while K512/K2048 are native-only; — denotes unavailable optimization results.

K	ID	MAE	FS	FI	Fit (s)	Profit	F
512	0	1.466	5	4	361.1	—	—
512	1	1.169	5	6	359.4	—	—
512	2	1.377	8	8	361.5	—	—
1024	0	0.809	3	3	761.0	136.00	Y
1024	1	0.807	4	6	1,567.4	140.00	Y
1024	2	0.778	4	4	773.4	138.00	Y
2048	0	0.518	3	1	1,165.5	—	—
2048	1	0.500	1	1	1,170.0	—	—
2048	2	0.499	4	1	1,175.2	—	—

Table 17: Knapsack same-fit encoding sensitivity. Error is mean absolute difference from native probability in percentage points, gap is percent, and time is in seconds. Resolutions reuse fits and are not additional independent fitting replications.

Knots	ID	MAE	Variables	Profit	Gap	Time (s)
9	0	0.566	53,167	140.00	0.00	737.8
9	1	0.532	53,069	141.00	0.00	822.1
9	2	0.557	53,126	140.00	0.00	846.5
9	3	0.532	53,188	141.00	0.00	874.0
9	4	0.547	53,204	140.00	0.00	806.1
9	5	0.539	53,190	136.00	3.68	1,800.1
9	6	0.547	53,176	140.00	0.00	1,046.1
9	7	0.539	52,547	136.00	2.94	1,800.2
9	8	0.543	53,175	139.00	0.00	734.7
9	9	0.542	53,180	136.00	2.94	1,800.2
17	0	0.163	69,453	136.00	2.94	1,800.6
17	1	0.168	69,262	141.00	0.00	938.4
17	2	0.166	69,379	140.00	0.00	1,093.4
17	3	0.170	69,498	141.00	0.00	1,030.7
17	4	0.172	69,526	136.00	2.94	1,800.4
17	5	0.172	69,497	136.00	3.68	1,800.1
17	6	0.165	69,472	136.00	2.94	1,800.2
17	7	0.159	68,213	136.00	2.94	1,800.4
17	8	0.171	69,476	140.00	0.00	1,119.4
17	9	0.178	69,481	140.00	0.00	773.9
33	0	0.083	102,025	136.00	2.94	1,800.4
33	1	0.080	101,644	140.00	0.00	1,072.9
33	2	0.082	101,879	138.00	0.00	936.3
33	3	0.081	102,112	140.00	0.00	991.3
33	4	0.082	102,176	140.00	0.00	1,192.0
33	5	0.083	102,112	140.00	0.00	1,213.7
33	6	0.084	102,060	138.00	0.00	1,204.5
33	7	0.078	99,569	140.00	0.00	1,058.9
33	8	0.086	102,066	140.00	0.00	953.6
33	9	0.083	102,082	140.00	0.00	956.9

Table 18: Knapsack matched optional-cut results. Each pair holds the fitted model, base encoding, starts and budget fixed. Gap is percent, time is in seconds, and all returned decisions in these pairs are population-feasible.

ID	Cuts	Profit	Bound	Gap	Time (s)
0	off	136.00	191.00	40.44	1,800.3
0	on	136.00	140.00	2.94	1,800.1
1	off	136.00	192.00	41.18	1,800.2
1	on	140.00	140.00	0.00	1,042.7
2	off	136.00	184.00	35.29	1,800.2
2	on	138.00	138.00	0.00	1,283.0

Table 19: Knapsack frozen-model/scenario reuse outcomes, part 1. Risk limit, population risk and solver gap are percentages; F marks population feasibility and time is in seconds. All three datasets and all five methods are retained at each threshold.

Limit	ID	Method	Profit	Risk	F	Gap	Time (s)
1.0	0	SAA 1k	131.00	0.745	Y	0.00	1.7
1.0	0	SAA 10k	131.00	0.745	Y	0.00	68.4
1.0	0	SAA 60k	131.00	0.745	Y	8.40	1,800.0
1.0	0	SAA 200k	122.00	0.008	Y	22.13	1,800.1
1.0	0	**TPM**	136.00	1.952	N	0.00	770.1
1.0	1	SAA 1k	131.00	0.745	Y	0.00	1.5
1.0	1	SAA 10k	131.00	0.745	Y	0.00	60.7
1.0	1	SAA 60k	131.00	0.745	Y	10.69	1,800.1
1.0	1	SAA 200k	122.00	0.008	Y	22.13	1,800.1
1.0	1	**TPM**	136.00	1.952	N	0.00	889.2
1.0	2	SAA 1k	131.00	0.745	Y	0.00	1.5
1.0	2	SAA 10k	131.00	0.745	Y	0.00	69.8
1.0	2	SAA 60k	131.00	0.745	Y	6.11	1,800.1
1.0	2	SAA 200k	122.00	0.008	Y	22.13	1,800.3
1.0	2	**TPM**	136.00	1.952	N	0.00	857.3
5.0	0	SAA 1k	136.00	1.952	Y	0.00	3.0
5.0	0	SAA 10k	136.00	1.952	Y	0.00	330.8
5.0	0	SAA 60k	136.00	1.952	Y	16.91	1,800.3
5.0	0	SAA 200k	136.00	1.952	Y	18.38	1,800.2
5.0	0	**TPM**	136.00	1.952	Y	0.00	764.0
5.0	1	SAA 1k	136.00	1.952	Y	0.00	2.4
5.0	1	SAA 10k	136.00	1.952	Y	0.00	393.8
5.0	1	SAA 60k	136.00	1.952	Y	16.91	1,800.1
5.0	1	SAA 200k	136.00	1.952	Y	18.38	1,800.3
5.0	1	**TPM**	136.00	1.952	Y	0.00	924.9
5.0	2	SAA 1k	136.00	1.952	Y	0.00	2.7
5.0	2	SAA 10k	136.00	1.952	Y	0.00	258.1
5.0	2	SAA 60k	136.00	1.952	Y	16.91	1,800.4
5.0	2	SAA 200k	136.00	1.952	Y	16.91	1,800.4
5.0	2	**TPM**	136.00	1.952	Y	0.00	852.3

Table 20: Knapsack frozen-model/scenario reuse outcomes, part 2. Risk limit, population risk and solver gap are percentages; F marks population feasibility and time is in seconds. All three datasets and all five methods are retained at each threshold.

Limit	ID	Method	Profit	Risk	F	Gap	Time (s)
15.0	0	SAA 1k	141.00	10.251	Y	0.00	3.3
15.0	0	SAA 10k	141.00	10.251	Y	10.64	1,800.1
15.0	0	SAA 60k	140.00	9.527	Y	25.71	1,800.3
15.0	0	SAA 200k	136.00	1.952	Y	30.15	1,800.2
15.0	0	**TPM**	140.00	9.527	Y	0.00	720.0
15.0	1	SAA 1k	141.00	10.251	Y	0.00	2.3
15.0	1	SAA 10k	141.00	10.251	Y	8.51	1,800.0
15.0	1	SAA 60k	141.00	10.251	Y	24.82	1,800.5
15.0	1	SAA 200k	136.00	1.952	Y	30.15	1,800.3
15.0	1	**TPM**	141.00	10.251	Y	0.00	1,077.3
15.0	2	SAA 1k	141.00	10.251	Y	0.00	2.8
15.0	2	SAA 10k	141.00	10.251	Y	0.00	1,619.2
15.0	2	SAA 60k	141.00	10.251	Y	24.82	1,800.3
15.0	2	SAA 200k	136.00	1.952	Y	30.15	1,800.2
15.0	2	**TPM**	141.00	10.251	Y	0.00	916.8
20.0	0	SAA 1k	143.00	20.422	N	0.00	3.2
20.0	0	SAA 10k	143.00	20.422	N	9.79	1,800.1
20.0	0	SAA 60k	141.00	10.251	Y	28.37	1,800.3
20.0	0	SAA 200k	136.00	1.952	Y	35.29	1,800.2
20.0	0	**TPM**	143.00	20.422	N	0.00	981.2
20.0	1	SAA 1k	143.00	20.422	N	0.00	2.5
20.0	1	SAA 10k	143.00	20.422	N	0.00	1,077.8
20.0	1	SAA 60k	141.00	10.251	Y	28.37	1,800.2
20.0	1	SAA 200k	136.00	1.952	Y	35.29	1,800.1
20.0	1	**TPM**	143.00	20.422	N	0.00	1,005.1
20.0	2	SAA 1k	143.00	20.422	N	0.00	2.3
20.0	2	SAA 10k	143.00	20.422	N	13.99	1,800.0
20.0	2	SAA 60k	141.00	10.251	Y	28.37	1,800.1
20.0	2	SAA 200k	136.00	1.952	Y	35.29	1,800.2
20.0	2	**TPM**	136.00	1.952	Y	5.15	1,800.1

F MULTI-PRODUCT NEWSVENDOR

F.1 INSTANCE CONSTRUCTION

F.1.1 PROBLEM FORMULATION

The newsvendor problem models inventory decisions made before demand is observed, with service-level variants imposing a chance constraint. We consider a retailer who chooses orders for n products before observing demand $D = (D_1, \ldots, D_n)$ in a single selling period. For each product i, the integer quantity q_i specifies the order, and the binary variable y_i^{act} indicates whether a positive order is placed. Each unit sold earns revenue r_i; each unsold unit retains continuation value v_i; each ordered unit incurs cost c_i; and placing a positive order incurs a fixed cost f_i. Writing $(t)_+ = \max(0, t)$, the profit under a demand realization d is

$$\Pi(q, y^{\mathrm{act}}, d) = \sum_{i=1}^{n} \left[r_i \min(d_i, q_i) + v_i (q_i - d_i)_+ - c_i q_i - f_i y_i^{\mathrm{act}} \right].$$

Orders share storage capacity B, with each unit of product i using a_i units of storage and each order bounded by U_i. The service requirement limits the probability that total unmet demand exceeds an allowance $L \geq 0$ to at most ϵ. The resulting formulation maximizes expected profit:

$$\begin{aligned} \max_{q, y^{\mathrm{act}}} \quad & \mathbb{E}(\Pi(q, y^{\mathrm{act}}, D)) \\ \text{s.t.} \quad & \mathbb{P}\left(\textstyle\sum_{i=1}^{n} (D_i - q_i)_+ > L\right) \leq \epsilon, \\ & \textstyle\sum_{i=1}^{n} a_i q_i \leq B, \\ & y_i^{\mathrm{act}} \leq q_i \leq U_i y_i^{\mathrm{act}}, \qquad i = 1, \ldots, n, \\ & q \in \mathbb{Z}_{\geq 0}^{n}, \qquad y^{\mathrm{act}} \in \{0, 1\}^n . \end{aligned} \tag{41}$$

The linking constraints make $y_i^{\mathrm{act}} = 1$ exactly when $q_i > 0$. For a specified demand distribution, expected profit is a deterministic function of these decisions, while the chance constraint limits the risk of excessive shortage. In the notation of Equation 1, this corresponds to $x = (q, y^{\mathrm{act}})$, $\Xi = D$, and $G(x, \xi) = \sum_i (\xi_i - q_i)_+ - L$; a total shortage equal to L is permitted.

F.1.2 GENERATING INSTANCE

Table 21 specifies the five-product instance. Independent demands satisfy $D_i = \max\{0, \lfloor Z_i + 1/2 \rfloor\}$, where $Z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. There is no upper-demand clipping. The legal orders satisfy $0 \leq q_i \leq U_i$ and $\sum_i a_i q_i \leq 100$, giving 6,039,420 orders. The service event is $\sum_i (D_i - q_i)_+ > 3$, with main risk limit 0.05.

F.2 BASELINES

F.2.1 POPULATION REFERENCE SOLUTION

The demand mass at zero is $\Phi((1/2 - \mu_i)/\sigma_i)$; at integer $d \geq 1$ it is the difference of the normal CDF at $d + 1/2$ and $d - 1/2$. To evaluate risk, the reference procedure convolves per-product shortage masses in buckets $0, \ldots, L$, accumulating all larger shortages in an overflow bucket. Thus the unbounded demand tail is retained in the violation probability. A direct enumeration of safe

Table 21: Five-product population instance. Demand is a nonnegative rounded Gaussian with the underlying normal mean and standard deviation shown. Storage capacity is 100 and the shortage allowance is 3.

Item	μ_i	σ_i	r_i	c_i	v_i	f_i	a_i	U_i
1	1.8223	0.3081	26	18	9	26	1	100
2	4.7423	1.4123	49	33	7	38	3	33
3	11.2545	3.1530	43	26	4	58	2	50
4	11.9515	2.7147	34	18	3	43	1	100
5	6.6330	1.2450	38	26	8	32	3	33

shortage tuples provides a cross-check; near-boundary decisions receive higher-precision evaluation. Population profit can equivalently be computed from the finite sums

$$m_i(q_i) = \sum_{t=1}^{q_i} \mathbb{P}\left(D_i \geq t\right), \qquad \mathbb{E}(\Pi(q, y^{\mathrm{act}}, D)) = \sum_i \left[(r_i - v_i) m_i(q_i) + (v_i - c_i) q_i - f_i y_i^{\mathrm{act}}\right]. \tag{42}$$

These formulas assess a returned order independently of either optimization representation.

The risk evaluation compares the convolution with direct safe-tuple enumeration at tolerance 10^{-12}. If they disagree or lie near the risk limit, it retries at 80, 160 and 320 decimal digits; unresolved comparisons raise an error. Profit tables use 80-digit evaluation, and comparisons allow ties within 10^{-9}.

The global reference search branches over legal orders. A dynamic program bounds the best separable population profit of each remaining storage-constrained suffix. A second pruning rule uses monotonicity of shortage risk: if setting every remaining quantity to its upper bound still violates the risk limit, no legal completion of that prefix is feasible. Profit bounds and this optimistic risk test account for every pruned completion; numerical tie handling retains the lexicographically first maximizing order. The saved coverage certificate partitions all 6,039,420 orders into 4,995,463 profit-pruned, 1,043,858 risk-pruned and 99 explicitly evaluated leaves, with 786 risk calls. Under the stated numerical policy, it identifies the reference order $(2, 6, 14, 15, 8)$, population profit 164.7024109742 and risk 4.66%. This is a population optimum; the campaign solvers maximize the common sample-profit objective. Profit pruning requires the suffix upper bound to fall more than 10^{-9} below the incumbent; risk pruning requires the optimistic completion to exceed the risk limit. The search covers the finite decision domain under these checked numerical comparisons, but does not provide a formal directed-rounding certificate.

F.2.2 Explicit SAA formulation

For newsvendor, N_{SAA} raw demand draws are grouped into distinct vectors d^g with multiplicities w_g. For each distinct vector d^g, the SAA formulation introduces shortage variables $h_{gi} \geq 0$ and a violation indicator v_g:

$$h_{gi} \geq d_i^g - q_i, \qquad \sum_i h_{gi} \leq L + M_g v_g, \qquad \sum_g w_g v_g \leq \lfloor \epsilon N_{\mathrm{SAA}} \rfloor. \tag{43}$$

The implemented bound is $M_g = \max\{0, \sum_i d_i^g - L\}$. Groups with $M_g = 0$ retain an indicator fixed to zero. At a fixed order, setting $h_{gi} = \max\{d_i^g - q_i, 0\}$ minimizes every shortage variable, so the projected risk constraint tests exactly the physical shortage event. Grouping identical vectors is exact because their shortage events coincide. Shared exactly-one quantity indicators also represent the separable sample-profit objective, whose coefficients use the full demand bank independently of the risk-scenario count.

These are the tested explicit formulations, including their scenario-specific safeguards and duplicate compression. They do not cover all strengthened or decomposed SAA methods discussed in section 2.

F.3 Data Generation and SPN Training

F.3.1 Observations and partitions

Each newsvendor training row also pairs an allocated order with a fresh demand vector, retaining both outcomes. The 30,197,100-row bank covers all 6,039,420 legal orders. A separate bank of 100,000 demand vectors is reused across 512 validation orders, producing 51,200,000 outcome evaluations, not that many independent demands. The 20,000-row joint validation view is a separate fixed dataset. SAA uses nested risk-scenario subsets of the training demand bank; all methods use the full bank to form the common sample-profit coefficients. Demand, validation and post-freeze assessment roles are recorded separately in the saved manifests.

F.3.2 Decision allocation and its selection

The order allocation is a chosen sampling design, separate from the unknown demand law. Repeated orders carry more weight in the joint NLL, while each row pairs its order with a fresh demand draw.

Newsvendor uses seven frozen allocation components. The broad component receives 7,549,275 rows (25%). Four risk-screening pools receive 1,887,319, 1,887,319, 1,887,319 and 1,887,318 rows (together 25%). They contain orders with empirical risks in $[0, 2.5\%]$, $(2.5\%, 7.5\%]$, $(7.5\%, 15\%]$ and $(15\%, 25\%]$, screened using the same frozen 20,000-demand development bank. Two profit pools receive 7,549,275 rows each (together 50%): the highest-profit 1% of legal orders and the next 4%. The broad component is a frozen full-coverage order sequence. Each remaining component samples its pool uniformly with replacement; every repeated order receives a fresh demand. The risk pools originate from saved development screening and empirical-risk assessments, whereas the profit pools rank only the frozen sample-profit objective, with legal-order index breaking ties. They do not use the population oracle. The risk bands are disjoint, as are the two profit pools; the broad sequence covers the domain and an order may belong to both a risk and a profit pool. The screened pools remain fixed across datasets, while training demands are redrawn independently.

The allocation concentrates observations where risk estimation and objective value can affect decisions while preserving broad coverage. It follows development of broad, boundary and profit-focused allocations; the final recipe was frozen for the paired campaign. These development choices are not a controlled demonstration that this allocation is optimal. Subsequent data-size and capacity studies assess the frozen recipe and must not be interpreted as independent confirmation preceding its selection.

F.3.3 Nested training-size subsets

The newsvendor training-size study uses replications 001–003. Nested permutation prefixes within each of the seven components, with largest-remainder allocation, preserve the component proportions up to integer rounding. The smaller banks contain 301,971 and 3,019,710 fresh-demand rows, covering approximately 211,000 and 1,026,000 distinct orders, respectively; the 30,197,100-row fits are reused. Initialization and preprocessing use each arm's available training data. Validation and the full-bank sample-profit objective remain fixed. Thus this study reduces data for fitting the probability model, not every source of information available to the optimization pipeline. It changes both the number of demand observations and the number of distinct orders seen during fitting, so their effects cannot be separated by this comparison.

The shared subset accounting note and the knapsack construction are in Appendix E.3.2.

F.3.4 Architecture and marginalized query

For newsvendor, the same product-mixture construction replaces each binary decision leaf with a categorical mass function over $\{0, \ldots, U_i\}$ and uses five Gaussian demand leaves and one Bernoulli violation leaf per component. The activation variables are determined by the orders and need no separate probabilistic leaves. Both main campaigns use $K = 1,024$. Their unfactored component scopes contain 21 variables for knapsack and 11 for newsvendor; after uncertainty marginalization, only the decision variables and the fixed or marginalized outcome remain. Implementation-level batched modules need not correspond one-to-one to these scalar factors. The benchmark rounds and clips demand to nonnegative integers, but the fitted Gaussian leaves evaluate a continuous density at each observed demand value; they do not integrate probability over integer bins or add dequantization noise. Exact marginalization therefore applies to this normalized continuous surrogate, whose density at an integer is not the probability of that integer event.

The conditional formula in Equation 31 applies with categorical products in place of Bernoulli products. Although uncertainty parameters disappear from the query, their likelihood contributions affect the shared mixture parameters during fitting. A mixture trained directly on (X, Y) can represent the same conditional family. The present experiments do not establish that observing uncertainty improves estimation over that alternative.

F.3.5 Joint-likelihood training recipe

The joint-NLL objective and parameter conventions in Appendix E.3.4 apply here as well.

Table 7 reports the final newsvendor recipe. Initialization uses a discrete pseudocount of 0.5, and gradient accumulation starts with microbatches of 128,000 rows. Its epoch-end partial-batch policy gives 452,956,500, 754,927,500 and 767,727,500 row presentations at the three training sizes, or

1,500, 250 and approximately 25.424 epochs. Fixed updates therefore imply neither equal data exposure nor convergence. Microbatch NLL sums are divided by the actual batch size, including an epoch-end partial batch. Gradients accumulate across microbatches, then are clipped before one Adam update.

F.3.6 CHECKPOINT AND CONFIGURATION SELECTION

Newsvendor evaluates initially and every 25 updates. Its 512 fixed orders comprise 64 orders in each of eight frozen screening bins. Checkpoints are ranked lexicographically by equally weighted bin MAE against empirical risks from the validation demands, then joint validation NLL, then earliest update. The gradient objective remains joint NLL throughout; a conditional validation criterion is not a conditional training loss. The same 512 orders, bin assignments and empirical targets are used for all fits. Those targets share one 100,000-demand validation bank; the separate 20,000-row joint-validation view breaks checkpoint ties. The allocation's 20,000-demand screening bank is a different development input. Independent training-demand replications therefore do not make these design and selection choices independent.

The primary newsvendor encoding remains PWL-9. The uniform knapsack PWL-33 view was adopted after inspecting same-fit resolution comparisons, replacing the original PWL-9 view without refitting. Accordingly, these results describe a development-informed policy; a future confirmation campaign should predeclare its encoding resolution. The supporting analyses retain all selected checkpoints and do not replace primary outcomes.

F.4 COMPILATION AND OPTIMIZATION SETTINGS

The shared numerical policies and solver settings are in Appendices B.6 and C.2. The following paragraphs specify newsvendor initialization and categorical strengthening.

Newsvendor uses a common 10,000-order initialization panel, choosing the highest sample-profit order feasible for each method and resolving ties lexicographically. If no candidate qualifies, search starts without an incumbent. TPM gives quantity selectors branching priority 100 and uses a first search phase of 60 seconds, followed by incumbent-supported rebuilding when applicable; rebuilding consumes the same cumulative allowance. SAA uses `MIPFocus=1` and business-variable priority 100. Initial TPM qualification has a separate 1,500-second cap. No post-search tie solves replace a returned order.

F.4.1 CATEGORICAL STRENGTHENING AND REBUILDING

The categorical formulation shares quantity indicators across the two probability queries. Exactly-one domain bounds, selected-maximum links and curve-preserving interval tightening strengthen its relaxation. These operations retain the original compiled probability constraint.

The optional group separator starts from a compiled-infeasible order. It expands coordinate category sets $\mathcal{G}_i$ only after checking every business-feasible completion in the proposed new Cartesian slice against the saved compiled query. The budget is 2,048 probability evaluations per expansion procedure; an unfinished check cannot certify an added category. Starting with the singleton order and applying this rule inductively establishes that every legal order in $\prod_i \mathcal{G}_i$ is compiled-infeasible. With category indicators d_{ik} from Table 4, the resulting exclusion is

$$\sum_i \sum_{k \in \mathcal{G}_i} d_{ik} \leq n - 1. \tag{44}$$

This preserves all encoded-feasible orders. It is a finite-completion argument, distinct from the analytical binary-face bound above. The implementation requires log-risk to exceed the threshold by 10^{-8}; that floating-point buffer is not a rigorous interval-arithmetic certificate. The same probability queries can identify primal candidates, which are independently checked before submission.

Rebuilding preserves potentially improving decisions rather than every feasible order. Write the common separable sample-profit objective as $\sum_i \phi_i(q_i)$ and let V be a validated incumbent value. With $\eta = 10^{-7}$, the implementation removes category k of product i only if

$$\phi_i(k) + \sum_{j \neq i} \max_t \phi_j(t) < V - 2\eta. \tag{45}$$

This upper bound ignores coupling restrictions, so every removed completion has sample profit below V. The rebuilt model retains a profit floor $V - \eta$ and the original PWL curves. If U_R is its solver bound, then $\max\{V, U_R\}$ bounds the original optimum; any earlier full-domain bound can tighten it further. A failed or timed-out rebuild leaves the validated incumbent and any earlier bound available, while rebuilding and search share the cumulative allowance.

In the cut ablation, only optional cut insertion is suppressed. Group queries, primal submissions, static strengthening and rebuilding remain active. Its outcomes therefore isolate insertion, not the entire callback or all strengthening.

F.5 Additional Results

Native probabilities use log-sum-exp inference in the fitted mixture; encoded probabilities evaluate the same saved PWL curves used by the optimizer. Risk is the direct violation query. Fixed-order and screening panels remain separate; the true-boundary panel uses population risks in $[0.025, 0.075]$. Duplicate entries retain their weights, missing assessments are not treated as zero, and saved probabilities are not clipped.

The following tables retain every reported primary outcome and all results from the supporting analyses, including unsafe returns, time limits and missing measurements. They extend the interpretations in subsection 4.3 and Appendix F.5.3 under the shared protocol of Appendix C.

F.5.1 Primary campaign results

Campaign scope. The primary newsvendor campaign uses ten fitted replications, with five optimization methods assessed for each replication. The separate risk-sensitivity probe reuses three fitted replications (datasets 1–3), evaluates all five methods, and reports compact comparisons at risk limits 2.5% and 10%; the complete stored threshold set is retained in the appendix tables.

Table 22 and Table 23 retain the accepted PWL-9 campaign. All ten TPM returns are population-feasible and four recover the population optimum; nine searches certify their encoded sample-profit optima. All SAA-10k, SAA-60k and SAA-200k returns recover the population optimum, illustrating a setting where the explicit reference is particularly competitive. Table 24 separates the population, native and encoded risk at each TPM return and distinguishes population profit from the optimized objective. The shared sample-profit objective introduces a separate estimation layer; returned-point probability errors and solver bounds alone do not identify the globally optimal native decision. The unresolved TPM search remains in the reported outcomes. Neither an earlier filtered transfer nor a later PWL-17 comparison replaces it. At each returned TPM order, population, native and encoded risks are all below the 5% limit. Thus the table shows no feasibility-classification disagreement at these particular orders. The solver gaps concern encoded sample profit, whereas a shortfall from the population-optimal profit is assessed under a different objective. These pointwise values cannot apportion that shortfall among objective sampling, probability errors elsewhere in the feasible set and incomplete search.

F.5.2 Complete campaign outcomes

Table 25 collects feasibility, optimum recovery, solver certification and search-time summaries for the two campaigns visualized in the main text.

F.5.3 Supporting diagnostics

The following supporting diagnostics interpret the primary campaigns through data size and model capacity, encoding fidelity, optional cuts, risk-limit reuse, formulation cost, and historical filtering. They are not independent confirmation of development choices. Knapsack-specific records appear in Appendix E.5; the joint risk probe uses three fitted knapsack replications (datasets 0–2), all five methods, and the complete stored threshold set.

F.5.4 Data-size and model-capacity results

For a fixed fitted architecture and encoding policy, additional training rows do not enter the submitted model individually; fitted parameters and retained PWL segments can still change. The follow-

Table 22: Newsvendor campaign, dataset blocks 1–5. Risk and solver gap are percentages; F indicates population feasibility. Profit is the population value; bound and gap refer to the optimized objective (sample profit for newsvendor). Gaps are rounded; certificates are counted separately.

ID	Method	Decision	Profit	Risk	F	Bound	Gap	Time (s)
1	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.72	0.00	2.6
1	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.72	0.00	29.9
1	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.72	0.00	406.6
1	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.72	0.00	1,692.2
1	**TPM**	2,5,15,15,8	162.19	4.486	Y	162.21	0.00	511.5
2	SAA 1k	2,6,14,16,7	163.37	4.977	Y	163.36	0.00	2.9
2	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.69	0.00	39.5
2	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.69	0.00	495.8
2	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.69	0.00	1,538.7
2	**TPM**	2,6,14,15,8	164.70	4.668	Y	164.69	0.00	476.3
3	SAA 1k	2,6,15,15,7	159.32	4.079	Y	159.33	0.00	3.2
3	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.71	0.00	41.2
3	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.71	0.00	459.9
3	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.71	0.00	1,368.6
3	**TPM**	2,5,15,15,8	162.19	4.486	Y	191.05	17.79	1,800.3
4	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.75	0.00	6.3
4	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.75	0.00	62.3
4	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.75	0.00	284.0
4	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.75	0.00	1,254.6
4	**TPM**	2,5,15,15,8	162.19	4.486	Y	162.25	0.00	275.4
5	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.73	0.00	2.1
5	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.73	0.00	31.1
5	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.73	0.00	427.7
5	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.73	0.00	1,201.3
5	**TPM**	2,5,15,15,8	162.19	4.486	Y	162.23	0.00	378.6

ing studies test whether they improve the fitted probabilities and resulting decisions, and whether increasing model capacity offers a similar benefit.

Figure 5 compares nested training banks with 1,024 components and the campaign's encoding policy held fixed. The knapsack study uses 10,240, 102,400 and 1,024,000 new training rows, with the 3,686,400-row fits reused as endpoints. The newsvendor study uses 301,971 and 3,019,710 rows, with the 30,197,100-row endpoints reused. Each size has three dataset replicates.

Knapsack mean native MAE decreases from 21.255 to 3.201, 0.978 and 0.798 pp across these sizes. All three smallest fits return no incumbent; exhaustive post-search assessment finds no decision feasible under their encoded constraints. At 102,400 rows, two of three returned decisions are population-feasible; at 1,024,000 rows, all three are feasible and two recover the population optimum. For newsvendor, mean native MAE on the uniform assessment panel decreases from 1.860 to 1.336 and 1.334 pp. The corresponding boundary MAEs are 1.118, 0.921 and 0.903 pp. Feasible returns increase from two of three to three of three, although profit does not improve monotonically with additional data.

Feasibility classification also remains imperfect. At the 10% limit, the native knapsack false-safe counts over all 1,024 decisions are 0/0/0, 5/6/4, 5/5/4 and 3/4/4 across the four ascending training sizes and datasets 0–2; the smallest fits instead reject all 405 truly feasible decisions. On the newsvendor panel of 523 near-5% entries per fit, native false-safe counts are 47/34/30, 35/23/26 and 25/29/26 across the three sizes. The corresponding encoded counts are 43/41/31, 40/26/28 and 26/35/34. These panel entries can repeat orders, and neither sequence supports a monotonic safety claim.

The initial formulations remain about 102,000 variables for knapsack and 28,000–30,000 for newsvendor. Their respective linear-constraint counts stay at 161,987 and 77,933; fitted bounds can change the exact variable count. Thus, additional observations improve probability estimates in these studies without proportional growth in submitted user-model size. The comparison fixes the

Table 23: Newsvendor campaign, dataset blocks 6–10. Risk and solver gap are percentages; F indicates population feasibility. Profit is the population value; bound and gap refer to the optimized objective (sample profit for newsvendor). Gaps are rounded; certificates are counted separately.

ID	Method	Decision	Profit	Risk	F	Bound	Gap	Time (s)
6	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	3.7
6	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	40.5
6	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	394.0
6	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	1,001.9
6	**TPM**	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	244.5
7	SAA 1k	2,5,15,15,8	162.19	4.486	Y	162.23	0.00	2.0
7	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	38.9
7	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	232.9
7	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.74	0.00	849.7
7	**TPM**	2,5,15,15,8	162.19	4.486	Y	162.23	0.00	305.5
8	SAA 1k	2,6,14,15,7	175.41	6.093	N	175.41	0.00	2.1
8	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.70	0.00	35.6
8	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.70	0.00	253.4
8	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.70	0.00	929.4
8	**TPM**	2,5,15,15,8	162.19	4.486	Y	162.18	0.00	322.1
9	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.68	0.00	2.2
9	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.68	0.00	25.3
9	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.68	0.00	309.8
9	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.68	0.00	997.9
9	**TPM**	2,6,14,15,8	164.70	4.668	Y	164.68	0.00	300.6
10	SAA 1k	2,6,14,15,8	164.70	4.668	Y	164.64	0.00	2.2
10	SAA 10k	2,6,14,15,8	164.70	4.668	Y	164.64	0.00	30.4
10	SAA 60k	2,6,14,15,8	164.70	4.668	Y	164.64	0.00	258.5
10	SAA 200k	2,6,14,15,8	164.70	4.668	Y	164.64	0.00	835.4
10	**TPM**	2,6,14,15,8	164.70	4.668	Y	164.64	0.00	254.2

update schedule rather than epochs or convergence, and smaller newsvendor banks also cover fewer distinct orders. Selected checkpoints and timing retain the fixed-update and historical-endpoint qualifications in Appendix E.3.4; subset construction is detailed in Appendix E.3.2.

Table 15 and Table 26 give every paired outcome, retaining no-return and unsafe-return cases.

Model capacity. The capacity results in Table 16 and Table 27 hold full-data size fixed. Mean native knapsack MAE decreases from 1.337 to 0.798 and 0.505 pp at 512, 1,024 and 2,048 components, but the new endpoints were not optimized. Newsvendor mean MAE likewise decreases from 1.373 to 1.334 and 1.234 pp, while population-feasible returns change from 3/3 to 3/3 and 1/3 and encoded certificates from 3/3 to 2/3 and 0/3. The 2,048-component model improves average error but returns two population-infeasible orders: their encoded risks are 4.836% and 4.922%, versus population risks 6.332% and 5.396%. Native risks at those orders were not saved, so the threshold disagreements cannot be assigned to fitting or compilation. All three searches also remain uncertified; that is a separate search outcome.

Training-data and model-capacity records. These records give the individual results underlying Appendix F.5.4. Reused full-data fits and native-only capacity assessments retain their original qualifications.

Knapsack records are in Tables 15 and 16.

F.5.5 Encoding fidelity and resolution sensitivity

Holding each fitted model fixed isolates how encoding refinement changes probability fidelity, formulation size and search. The scalar comparison appears in Appendix D.3; the following comparison concerns the two optimization problems.

Table 24: TPM returned-decision assessment in every primary dataset. Risks are percentages. Population profit is distinguished from the optimized objective (sample profit in newsvendor). O denotes own-model optimality; T denotes the search limit. These pointwise values do not establish global native optima.

Domain	ID	Population risk	Native	Encoded	Population profit	Objective	Status
Knapsack	0	1.952	0.723	0.596	136.000	136.000	T
Knapsack	1	9.527	8.954	8.564	140.000	140.000	O
Knapsack	2	7.049	6.053	5.665	138.000	138.000	O
Knapsack	3	9.527	8.097	8.099	140.000	140.000	O
Knapsack	4	9.527	8.402	7.836	140.000	140.000	O
Knapsack	5	9.527	8.136	7.647	140.000	140.000	O
Knapsack	6	7.049	6.928	6.530	138.000	138.000	O
Knapsack	7	9.527	7.329	7.346	140.000	140.000	O
Knapsack	8	9.527	8.255	7.507	140.000	140.000	O
Knapsack	9	9.527	8.195	7.670	140.000	140.000	O
Newsvendor	1	4.486	4.452	4.520	162.193	162.205	O
Newsvendor	2	4.668	4.628	4.310	164.702	164.690	O
Newsvendor	3	4.486	4.875	4.884	162.193	162.199	T
Newsvendor	4	4.486	4.282	4.024	162.193	162.245	O
Newsvendor	5	4.486	4.845	4.683	162.193	162.226	O
Newsvendor	6	4.668	4.884	4.517	164.702	164.735	O
Newsvendor	7	4.486	4.225	3.790	162.193	162.227	O
Newsvendor	8	4.486	4.617	4.858	162.193	162.184	O
Newsvendor	9	4.668	4.236	4.463	164.702	164.677	O
Newsvendor	10	4.668	4.841	4.657	164.702	164.639	O

Table 25: Complete campaigns: counts out of ten independent datasets per domain. Optimality certificates concern each method's own formulation; time is median initialization plus search, including newsvendor rebuilding.

Domain	Method	Population feasible	Population optimum	Certified	Time (s)
Knapsack	SAA 1k	5	3	10	2.6
	SAA 10k	7	7	7	554.8
	SAA 60k	10	3	0	1,800.4
	SAA 200k	10	0	0	1,800.2
	TPM	10	7	9	1,065.9
Newsvendor	SAA 1k	9	6	10	2.4
	SAA 10k	10	10	10	37.3
	SAA 60k	10	10	10	351.9
	SAA 200k	10	10	10	1,101.6
	TPM	10	4	9	313.8

Table 17 includes all ten fixed knapsack fits at PWL-9/17/33; Table 28 includes the three fixed newsvendor fits. These are changes to encoding, not additional training replications. In knapsack, mean native-to-encoded MAE falls from approximately 0.544 to 0.168 and 0.082 percentage points. Returned profits change across resolutions; PWL-33 yields nine encoded optimality certificates, while one search still reaches its time limit. The improvement in probability fidelity does not isolate its effect from search. In newsvendor, mean native-to-encoded MAE decreases from approximately 0.441 to 0.157 and 0.064 pp, yet the same orders are returned at all three resolutions. The median initial variable counts grow from roughly 28,000 to 42,000 and 69,000. The certificate counts are 2/3, 3/3 and 1/3, showing that greater fidelity need not improve search monotonically. Figure 6 displays the errors and observed ranges. The primary campaign remains PWL-9. Knapsack's uniform PWL-33 publication view was selected after this sensitivity analysis, as disclosed in Appendix F.3.6. The scalar comparison in Appendix D.3 holds its leaf interpolation fixed; refining sum-node grids alone does not remove leaf or numerical-guard error (Appendix B.6).

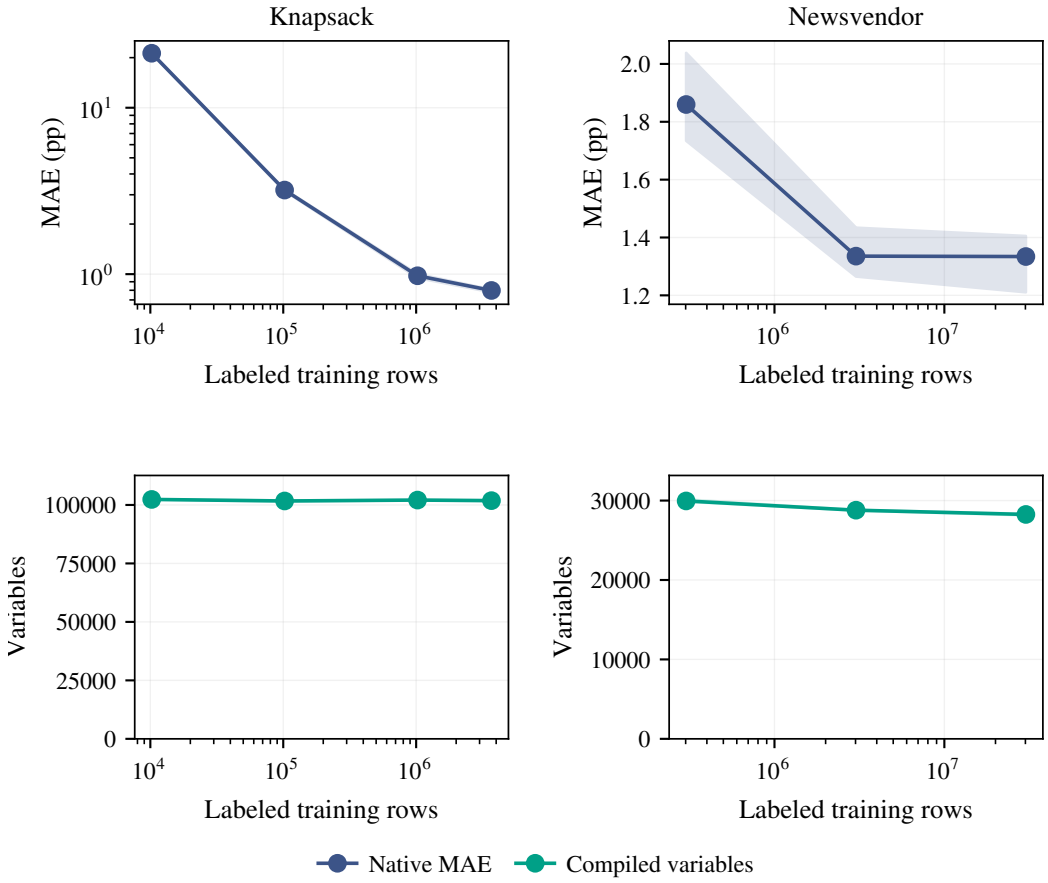

Figure 5: Training-data sensitivity at fixed architecture and encoding policy. Lines connect means over three datasets; shaded regions span the observed minimum and maximum, not confidence intervals. The largest-size endpoints reuse the main fits. Native MAE uses all 1,024 knapsack decisions and the newsvendor uniform-order assessment panel; these are different assessment populations.

Table 26: Newsvendor training-size outcomes. An asterisk marks reused full-data endpoints. MAE and boundary MAE are percentage points; gap is percent. No-return cases are not classified as unsafe. Boundary definitions differ across domains as specified in the assessment protocol.

Rows	ID	MAE	Boundary	Profit	F	Gap	Time (s)
301,971	1	1.734	1.122	165.38	N	0.00	363.7
301,971	2	1.807	1.109	162.19	Y	0.00	429.5
301,971	3	2.038	1.123	162.19	Y	0.00	477.4
3,019,710	1	1.308	0.827	164.70	Y	0.00	244.9
3,019,710	2	1.434	1.001	162.19	Y	0.00	378.2
3,019,710	3	1.265	0.936	159.32	Y	0.00	364.3
30,197,100*	1	1.405	0.908	162.19	Y	0.00	511.5
30,197,100*	2	1.211	0.845	164.70	Y	0.00	476.3
30,197,100*	3	1.387	0.956	162.19	Y	17.79	1,800.3

Encoding-fidelity records. The scalar tables separate resolution selection from final assessment. The campaign tables compare resolutions on unchanged fitted models, as discussed in Appendix F.5.5.

Table 27: Newsvendor model-capacity study: native MAE in percentage points, false-safe (FS) and false-infeasible (FI) counts, fit time, and returned profit/feasibility. K1024 reuses the primary fit; K512/K2048 include optimization runs. Counts cover the population-risk boundary panel.

K	ID	MAE	FS	FI	Fit (s)	Profit	F
512	1	1.368	41	58	1,252.9	162.19	Y
512	2	1.409	37	55	1,251.7	162.19	Y
512	3	1.343	38	46	1,254.9	162.19	Y
1024	1	1.405	25	60	2,449.5	162.19	Y
1024	2	1.211	29	35	2,441.7	164.70	Y
1024	3	1.387	26	57	2,433.8	162.19	Y
2048	1	1.241	30	38	4,289.2	164.70	Y
2048	2	1.181	18	31	4,301.5	172.90	N
2048	3	1.278	31	38	4,304.1	166.24	N

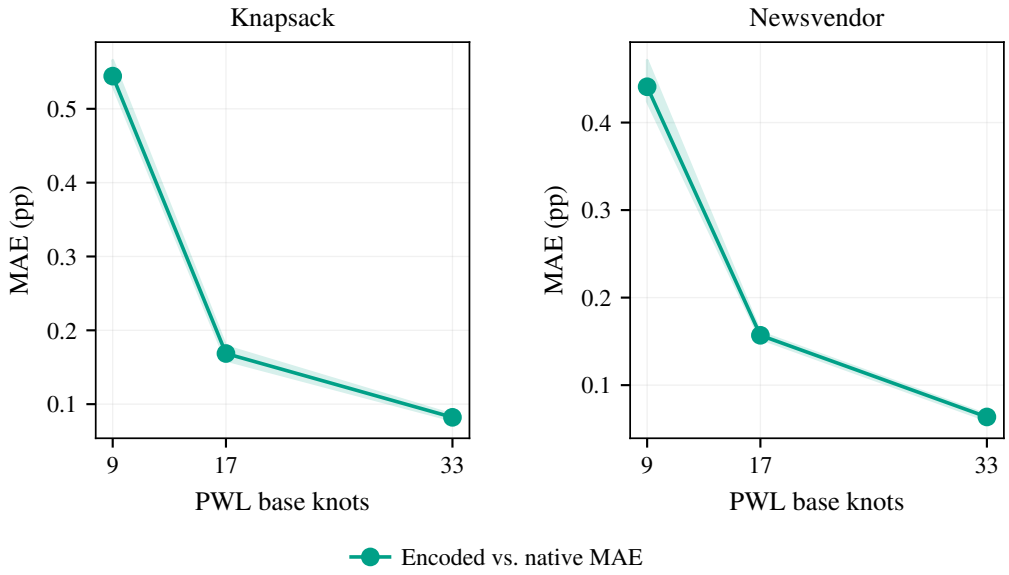

Figure 6: Encoding sensitivity on unchanged fitted models: ten knapsack fits and three newsvendor fits. Lines show mean absolute probability error relative to native inference; bands span dataset minima and maxima. The two panels use different decision-assessment populations.

Scalar records are in Appendices D.3 and D.5; knapsack records are in Table 17.

F.5.6 Matched optional-cut comparisons

Each of the three pairs per domain holds the fitted model, compiled probability constraint, static strengthening, solver settings, start and budget fixed; saved pair checks confirm matching compiled records and starts. The knapsack off arm suppresses optional decision and face cuts but retains incumbent checks. The newsvendor off arm suppresses optional cut insertion while retaining group evaluation, primal submissions and rebuilding.

Table 18 shows improved bounds in all three knapsack pairs and improved returned profit in two. Table 29 shows identical returned orders within each newsvendor pair but tighter bounds with insertion: two on-arm searches certify optimality, whereas no off-arm search does. Every returned decision in these pairs is population-feasible. These results establish benefits for bounds and certification under the tested budget; they do not establish that the framework cannot return useful

Table 28: Newsvendor same-fit encoding sensitivity. Error is mean absolute difference from native probability in percentage points; gap is percent. Resolutions reuse fits and are not additional independent fitting replications.

Knots	ID	MAE	Variables	Profit	Gap	Time (s)
9	1	0.424	28,275	162.19	0.00	511.5
9	2	0.471	28,345	164.70	0.00	476.3
9	3	0.427	28,139	162.19	17.79	1,800.3
17	1	0.153	42,039	162.19	0.00	438.2
17	2	0.160	42,218	164.70	0.00	439.4
17	3	0.157	41,792	162.19	0.00	510.3
33	1	0.065	69,584	162.19	25.02	1,800.4
33	2	0.060	69,954	164.70	0.00	381.9
33	3	0.066	69,098	162.19	18.70	1,800.6

Table 29: Newsvendor matched optional-cut results. Each pair holds the fitted model, base encoding, starts and budget fixed. Gap is percent; all returned decisions in these pairs are population-feasible. Newsvendor disables insertion only, retaining other callback operations.

ID	Cuts	Profit	Bound	Gap	Time (s)
1	off	162.19	248.06	52.93	1,800.6
1	on	162.19	162.21	0.00	634.1
2	off	164.70	246.86	49.89	1,800.4
2	on	164.70	164.69	0.00	577.0
3	off	162.19	248.06	52.93	1,800.3
3	on	162.19	196.02	20.85	1,800.3

decisions without optional cuts. Callback work is charged to search, and the newsvendor off arm still incurs group-query overhead.

Optional-cut records. These matched pairs support Appendix F.5.6. Each comparison disables only its specified operation; the newsvendor off arm retains other callback work.

The knapsack pairs are listed in Table 18.

F.5.7 Additional risk-tolerance results

The risk-limit reuse studies hold fitted models and scenario banks fixed while changing risk limits. Each domain uses three fitted replications and all five optimization methods. Table 19–Table 20 contain all sixty knapsack returns at 1%, 5%, 15% and 20%; Table 30 contains all thirty newsvendor returns at 2.5% and 10%. The compact comparison retains 5% and 20% for knapsack and 2.5% and 10% for newsvendor; the complete stored threshold set remains in the appendix tables. The original allocation and checkpoint selection were designed for the primary limits, not these new tasks. The event definition and generating law remain fixed: changing knapsack capacity W or newsvendor shortage allowance L would change the labels, and distribution shift would change the target probability. Population-optimal profits are assessed separately at each limit: 131, 136, 141 and 142 for knapsack in ascending limit order, and 136.580 and 188.985 for newsvendor. Table 4.3 summarizes the matching half/double-risk comparisons across both problems; Appendix F.5.7 retains the additional thresholds and detailed search outcomes.

At knapsack's 1% limit, all three TPM solutions are encoded-optimal but population-infeasible. Newsvendor TPM returns are population-feasible in only 1/3 cases at 2.5% and 2/3 at 10%. The corresponding SAA outcomes remain alongside these failures. Thus reuse is operationally possible, but transferring the same fitted representation does not guarantee safe decisions at a new confidence requirement. These studies do not establish an amortized computational advantage.

Table 30: Newsvendor frozen-model/scenario reuse outcomes, part 1. Risk limit, population risk and solver gap are percentages. F marks population feasibility. All three datasets and all five methods are retained at each threshold.

Limit	ID	Method	Profit	Risk	F	Gap	Time (s)
2.5	1	SAA 1k	136.58	2.083	Y	0.00	1.7
2.5	1	SAA 10k	136.58	2.083	Y	0.00	18.9
2.5	1	SAA 60k	136.58	2.083	Y	0.00	167.4
2.5	1	SAA 200k	136.58	2.083	Y	0.00	987.9
2.5	1	**TPM**	136.58	2.083	Y	48.88	1,800.3
2.5	2	SAA 1k	136.61	2.901	N	0.00	1.9
2.5	2	SAA 10k	136.58	2.083	Y	0.00	17.0
2.5	2	SAA 60k	136.58	2.083	Y	0.00	194.2
2.5	2	SAA 200k	136.58	2.083	Y	0.00	791.4
2.5	2	**TPM**	148.62	2.926	N	0.00	429.8
2.5	3	SAA 1k	136.58	2.083	Y	0.00	1.6
2.5	3	SAA 10k	136.58	2.083	Y	0.00	15.9
2.5	3	SAA 60k	136.58	2.083	Y	0.00	107.5
2.5	3	SAA 200k	136.58	2.083	Y	0.00	698.9
2.5	3	**TPM**	148.62	2.926	N	0.00	475.7
10.0	1	SAA 1k	190.99	10.025	N	0.00	3.6
10.0	1	SAA 10k	188.99	8.631	Y	0.00	71.7
10.0	1	SAA 60k	188.99	8.631	Y	0.00	786.1
10.0	1	SAA 200k	188.99	8.631	Y	6.73	1,801.0
10.0	1	**TPM**	190.99	10.025	N	0.00	168.8
10.0	2	SAA 1k	190.99	10.025	N	0.00	4.2
10.0	2	SAA 10k	188.99	8.631	Y	0.00	101.5
10.0	2	SAA 60k	188.99	8.631	Y	0.00	899.5
10.0	2	SAA 200k	188.99	8.631	Y	6.73	1,801.0
10.0	2	**TPM**	187.02	9.776	Y	0.00	196.8
10.0	3	SAA 1k	188.99	8.631	Y	0.00	4.5
10.0	3	SAA 10k	188.99	8.631	Y	0.00	83.4
10.0	3	SAA 60k	188.99	8.631	Y	0.00	987.4
10.0	3	SAA 200k	188.99	8.631	Y	5.08	1,800.9
10.0	3	**TPM**	188.12	9.493	Y	0.00	236.7

The unsafe knapsack returns are admitted by the native model before compilation. At 1%, native risks are 0.723, 0.648 and 0.731% across datasets 0–2; encoded risks are 0.596, 0.430 and 0.530%, against 1.952% population risk. At 20%, datasets 0 and 1 return a decision with 20.422% population risk; native risks are 18.789 and 18.531%, and encoded risks are 18.688 and 18.513%.

For newsvendor, the unsafe returns at 2.5% have encoded risks 2.022 and 2.482% in replications 002 and 003, against 2.926% population risk. The unsafe 10% return in replication 001 has encoded risk 9.430% and population risk 10.025%. Native probabilities at these returned orders were not saved, so their errors cannot be assigned specifically to fitting or encoding.

Risk-tolerance records. These tables retain all thresholds and per-run outcomes discussed in Appendix F.5.7, including the additional knapsack risk limits omitted from the compact main-text table.

The additional knapsack thresholds and records are in Tables 19 and 20.

F.5.8 Formulation dimensions and cost records

These tables extend the compact main-text comparison. Stage boundaries, allocated costs and the distinction between medians of totals and sums of medians are specified in Appendix C.3.

Table 32 reports submitted user-model sizes before presolve, solver SOS2 reformulation, search cuts, or newsvendor rebuilding. Knapsack TPM PWL-33 has 102,063 variables, 2,058 explicit binaries, 161,987 linear rows, and 2,050 SOS2 sets: more user-model variables and rows than SAA-60k. Its explicit binary count is smaller, but that count excludes any discrete structure introduced internally to process SOS2 sets and is therefore not a like-for-like measure of integrality. Binary

Table 31: Stage medians and medians of per-run accounted totals (seconds). Totals need not equal the sum of the stage medians. Newsvendor totals include allocated shares of preparation and experimental-design costs; accounting details are in subsection C.3. Search includes initialization; dashes indicate missing measurements.

Domain	Method	Data	Fit	Compile/check	Search	Total
Knapsack	SAA 1k	4.9	0.0	0.3	2.6	7.8
	SAA 10k	4.9	0.0	0.5	554.8	560.3
	SAA 60k	4.9	0.0	2.0	1,800.4	1,807.3
	SAA 200k	4.9	0.0	6.1	1,800.2	1,811.3
	TPM	4.9	697.0	236.3	1,065.9	2,128.6
Newsvendor	SAA 1k	20.2	0.0	0.2	2.4	23.2
	SAA 10k	20.2	0.0	0.7	37.3	58.7
	SAA 60k	20.2	0.0	1.9	351.9	371.9
	SAA 200k	20.2	0.0	2.8	1,101.6	1,125.2
	TPM	20.2	2,310.6	30.1	313.8	2,641.2

Table 32: Median dimensions of submitted user models before presolve, SOS2 reformulation, search cuts, or rebuilding. Newsvendor uses the initial full formulation, before incumbent-supported rebuilding. Binary counts include explicit user-model binaries only and exclude discrete variables that the solver may introduce internally to process SOS2 sets; SOS2 sets are reported separately. Binary variables are included in total variables. These dimensions do not characterize relaxation strength or the final presolved model.

Domain	Method	Variables	Binary	Linear rows	SOS2
Knapsack	SAA 1k	1,010	1,010	1,001	0
	SAA 10k	10,010	10,010	10,001	0
	SAA 60k	60,010	60,010	60,001	0
	SAA 200k	200,010	200,010	200,001	0
	TPM	102,063	2,058	161,987	2,050
Newsvendor	SAA 1k	5,608	1,206	5,294	0
	SAA 10k	30,202	5,304	29,888	0
	SAA 60k	76,921	13,091	76,607	0
	SAA 200k	126,340	21,328	126,026	0
	TPM	28,143	2,374	77,933	2,050

strengthening alone adds 20,480 continuous decision copies across the two queries (Equation 34). Newsvendor uses categorical decisions without those copies; all its initial models share 321 category and objective variables. Fitted bounds can change the retained PWL segments, but additional training rows do not enter either TPM formulation one by one.

The study reports submitted dimensions and final search outcomes but not transformed-model dimensions or root-relaxation gaps; it therefore does not establish that one formulation has a stronger relaxation. The optional-cut comparison is in Appendix F.5.6. Table 31 reports fitting, compilation and search separately. Its total column gives medians of per-run accounted totals, which need not equal sums of stage medians; Appendix C.3 explains shared-cost allocation and missing stage measurements.

F.5.9 Development study: challenges with data filtering

During development, we explored filtering training observations as a way to reduce the size of the compiled optimization formulation. Filtering changes the distribution represented by the retained data, so recovering the original violation probability requires accounting for the selection rule. This exploratory study is separate from the main method, which learns from both violations and nonviolations.

For success filtering, let the original decision-sampling mass be $p_X(x) > 0$ and let $\mathcal{S} = \{Y = 0\}$. Bayes' rule gives

$$\mathbb{P}(\mathcal{S} \mid X = x) = \frac{p_{X|\mathcal{S}}(x)\mathbb{P}(\mathcal{S})}{p_X(x)}. \tag{46}$$

A filtered model of $(X, \Xi) \mid \mathcal{S}$ must therefore recover the success-conditioned decision marginal, the retention probability and the original sampling law. Replacing these quantities by fitted or empirical estimates introduces error; the population identity does not certify the resulting query. A uniform decision design simplifies $p_X(x)$, but a stratified design requires its actual masses. A fitted correction can even require explicit probability-range restrictions.

Both historical examples retained only $Y = 0$ rows from uniform decision designs, then fitted the joint distribution of decision and uncertainty conditional on success. Knapsack used 3,600 uncertainty draws per binary decision; its success estimate multiplies the fitted decision marginal by the retained fraction $\alpha = 0.503355$ and 1,024 decisions, and is restricted to $[0.9, 1]$. Newsvendor used five draws per legal order; the corresponding factors are $\alpha = 0.051965$ and 6,039,420 legal orders, with success restricted to $[0.95, 1]$. Each α is estimated from the unfiltered training bank. The later primary newsvendor allocation is stratified, so its decision frequencies cannot be substituted into either uniform-design correction.

The filtered-knapsack development study obtained encoded certificates with profits 140, 138 and 136 across three seeds, while the corresponding population risks were 9.527%, 7.049% and 11.605%. Certification therefore did not remove model error. In a separate five-product transfer, a strengthened one-hour solve returned an order with native risk 9.356%, encoded risk 3.501% and empirical risk 31.647% on 100,000 fresh demands (pointwise 95% Wilson interval 31.359–31.936%). Its remaining own-model gap was 0.0473%. This example exhibits both fitted-query and encoding discrepancies, despite a small search gap.

These historical experiments changed more than filtering alone and cannot identify filtering as the sole cause. They motivated retaining both outcomes and the fitted conditional denominator in the primary construction. Their seeds, formulations, budgets and empirical assessments remain separate from the paired-campaign replication counts.