Bayesian–MILP Model for Multi-Tier Supply Chains

Integrating Structural Equation Modeling and Customer Satisfaction Optimization

Dr. Daniel A. Olivares Vera

2026-02-24

Abstract

This paper proposes a Bayesian–MILP model that integrates structural equation modeling (SEM), Bayesian optimization (BO), and mixed-integer linear programming (MILP) for multi-tier supply chains oriented to customer value maximization.

Table of contents

Download PDF version

1 Introduction

Traditional supply chain optimization focuses primarily on cost minimization or efficiency maximization. However, in multi-tier supply networks, upstream process decisions may strongly influence downstream customer satisfaction through latent quality constructs.

2 Structural Equation Model Specification

The hierarchical SEM is defined as follows:

$$\eta_i = \mathbf{x}_i^\top \beta + \zeta_i \tag{1}$$

$$Y_{ik} = \lambda_k \eta_i + \varepsilon_{ik} \tag{2}$$

where:

- $\mathbf{x}_i$ is the vector of process variables,
- β are structural coefficients,
- λ_k are measurement loadings,
- $\zeta_i \sim \mathcal{N}(0, \sigma_\zeta^2)$,
- $\varepsilon_{ik} \sim \mathcal{N}(0, \sigma_{\varepsilon_k}^2)$.

As shown in Eq. Equation 1 and Eq. Equation 2,

Como se muestra en la Figura Figure 1, el modelo integra la red multi-eslabón con la capa Bayesiana y la capa de optimización.

3 Simulated SEM Results

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path

np.random.seed(42)
```

Figure 1

```
out_fig = Path("results/figures")
out_tab = Path("results/tables")
out_fig.mkdir(parents=True, exist_ok=True)
out_tab.mkdir(parents=True, exist_ok=True)

n_products = 200
n_attributes = 3
n_process_vars = 5

beta_true = np.random.normal(0, 0.8, n_process_vars)
lambda_true = np.array([0.9, 1.1, 0.8])
sigma_zeta_true = 0.5
sigma_eps_true = np.array([0.4, 0.3, 0.5])

X = np.random.normal(0, 1, (n_products, n_process_vars))
zeta = np.random.normal(0, sigma_zeta_true, n_products)
eta = X @ beta_true + zeta

Y = np.zeros((n_products, n_attributes))
for k in range(n_attributes):
    eps = np.random.normal(0, sigma_eps_true[k], n_products)
    Y[:, k] = lambda_true[k] * eta + eps

dfY = pd.DataFrame(Y, columns=["Durability", "Appearance", "Comfort"])
corr = dfY.corr()
```

```
corr.to_csv(out_tab / "corr_attributes.csv", index=True)

plt.figure()
plt.hist(eta, bins=30)
plt.title("Simulated latent quality  ")
plt.tight_layout()
plt.savefig(out_fig / "eta_hist.png", dpi=200)
plt.show()

corr.round(3)
```

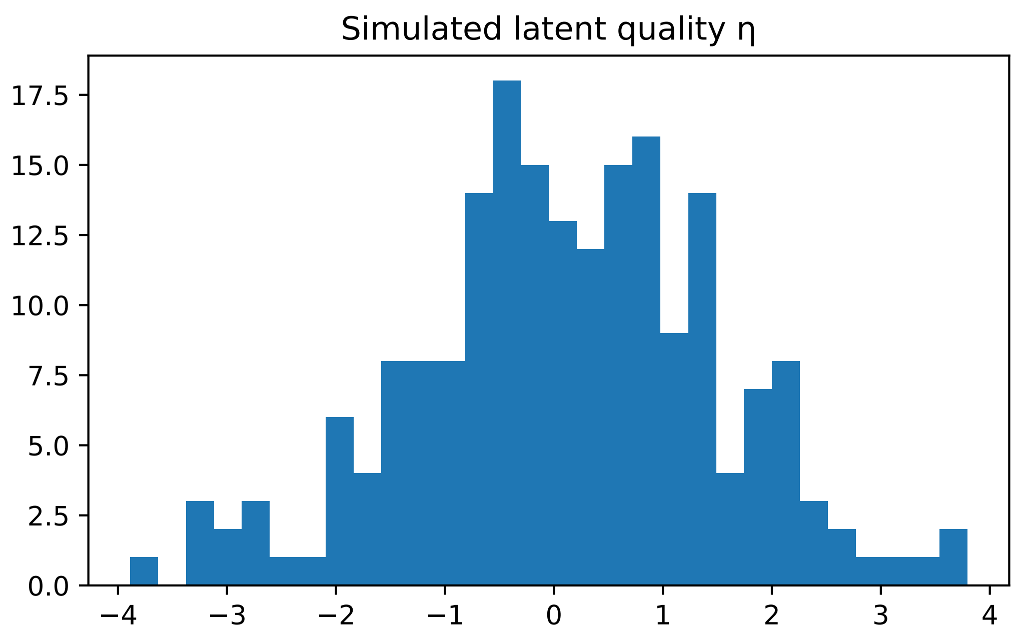


Figure 2: Distribution of simulated latent quality ( ).

	Durability	Appearance	Comfort
Durability	1.000	0.934	0.869
Appearance	0.934	1.000	0.899
Comfort	0.869	0.899	1.000

44 # 4 Bayesian SEM Estimation (PyMC)

```
import pymc as pm
import pytensor.tensor as pt
import arviz as az
```

```python
Y_obs = dfY.values
X_obs = X

n, J = X_obs.shape
K = Y_obs.shape[1]

with pm.Model() as sem_model:
    beta = pm.Normal("beta", mu=0.0, sigma=1.0, shape=J)
    sigma_zeta = pm.HalfNormal("sigma_zeta", sigma=1.0)

    lam = pm.Normal("lambda", mu=1.0, sigma=0.5, shape=K)
    sigma_eps = pm.HalfNormal("sigma_eps", sigma=1.0, shape=K)

    eta_latent = pm.Normal(
        "eta",
        mu=pt.dot(X_obs, beta),
        sigma=sigma_zeta,
        shape=n
    )

    muY = eta_latent[:, None] * lam[None, :]

    pm.Normal("Y", mu=muY, sigma=sigma_eps, observed=Y_obs)

    idata = pm.sample(
        draws=800,
        tune=800,
        chains=2,
        target_accept=0.9,
        random_seed=42,
        progressbar=False
    )

az.summary(idata, var_names=["beta", "lambda", "sigma_zeta", "sigma_eps"], round_to=3)
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_ha
beta[0]	0.485	0.099	0.306	0.666	0.025	0.007	14.622	85.950	1.098
beta[1]	-0.124	0.053	-0.220	-0.020	0.005	0.002	109.609	410.665	1.018
beta[2]	0.570	0.111	0.368	0.766	0.026	0.008	17.392	72.167	1.097
beta[3]	1.386	0.240	0.934	1.808	0.067	0.022	13.269	45.011	1.127
beta[4]	-0.165	0.055	-0.263	-0.061	0.008	0.001	46.888	464.406	1.036
lambda[0]	0.776	0.143	0.563	1.053	0.043	0.017	14.097	45.235	1.124
lambda[1]	0.968	0.178	0.694	1.301	0.053	0.021	14.190	41.852	1.127
lambda[2]	0.709	0.132	0.499	0.957	0.039	0.015	14.669	47.337	1.121

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_ha
sigma_zeta	0.627	0.117	0.418	0.828	0.031	0.010	14.858	72.299	1.113
sigma_eps[0]	0.403	0.028	0.352	0.454	0.001	0.001	695.670	1067.204	1.006
sigma_eps[1]	0.311	0.039	0.236	0.385	0.003	0.001	156.141	200.884	1.023
sigma_eps[2]	0.504	0.029	0.454	0.559	0.001	0.001	951.830	1221.667	1.002

45
5 Posterior Diagnostics

```
az.plot_trace(idata, var_names=["beta", "lambda", "sigma_zeta", "sigma_eps"])
plt.tight_layout()
plt.show()
```

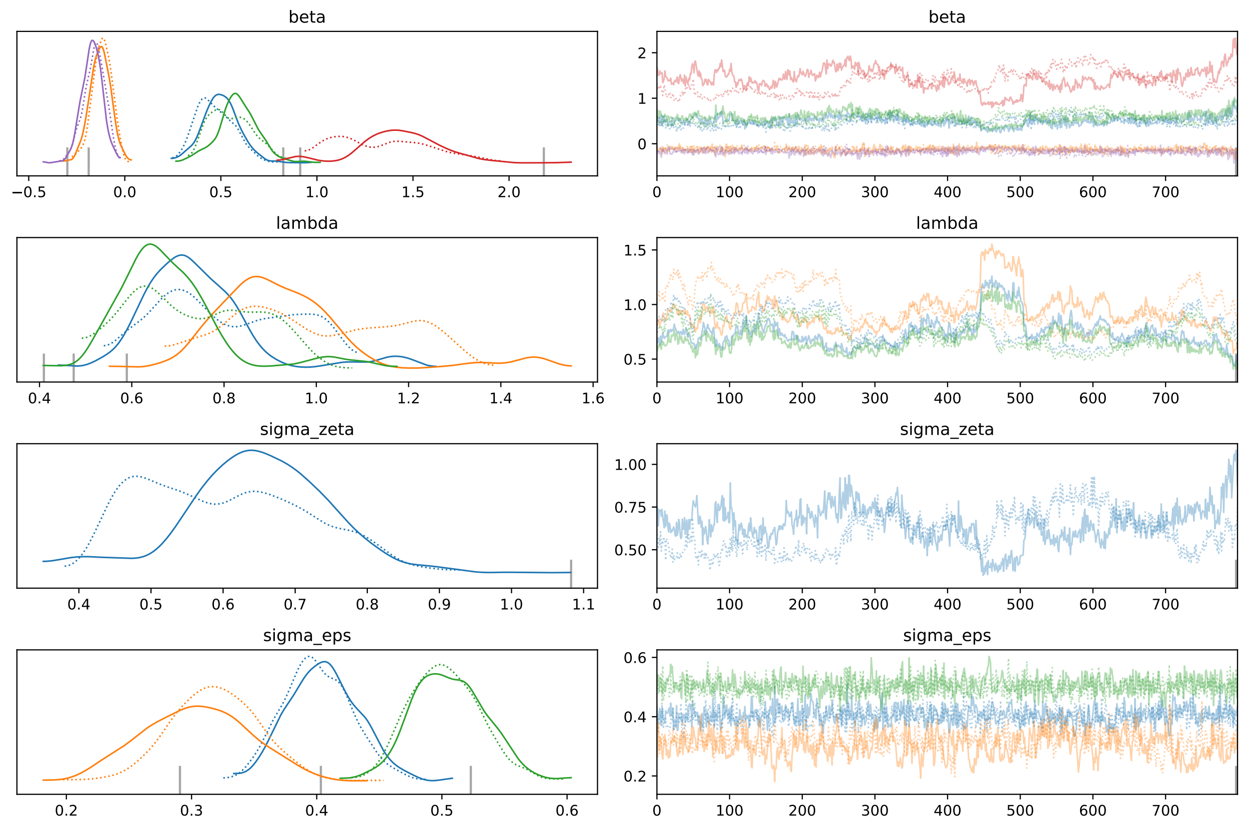


Figure 3: MCMC trace diagnostics.

46
6 Parameter Recovery Check

```
post = az.summary(idata, var_names=["beta", "lambda"], round_to=3)

post_mean_beta = post.loc[[f"beta[{j}]" for j in range(J)], "mean"].to_numpy()
```

```python
post_mean_lam = post.loc[[f"lambda[{k}]" for k in range(K)], "mean"].to_numpy()

comparison = pd.DataFrame({
    "true": np.concatenate([beta_true, lambda_true]),
    "posterior_mean": np.concatenate([post_mean_beta, post_mean_lam])
})

comparison
```

	true	posterior_mean
0	0.397371	0.485
1	-0.110611	-0.124
2	0.518151	0.570
3	1.218424	1.386
4	-0.187323	-0.165
5	0.900000	0.776
6	1.100000	0.968
7	0.800000	0.709

```python
plt.figure()
plt.scatter(comparison["true"], comparison["posterior_mean"])
plt.xlabel("True value")
plt.ylabel("Posterior mean")
plt.plot(
    [comparison["true"].min(), comparison["true"].max()],
    [comparison["true"].min(), comparison["true"].max()]
)
plt.tight_layout()
plt.show()
```

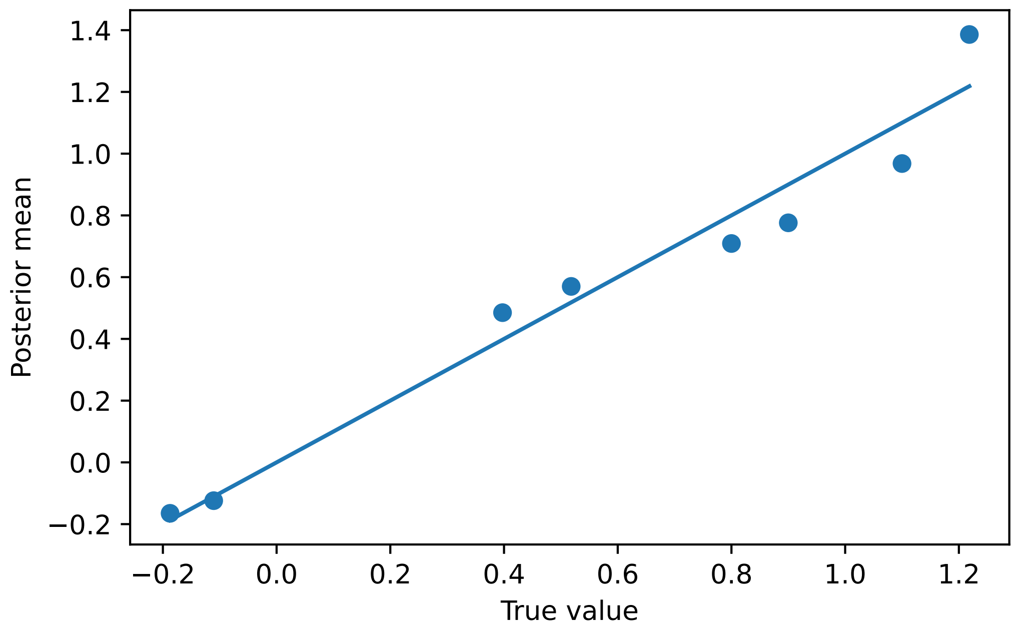
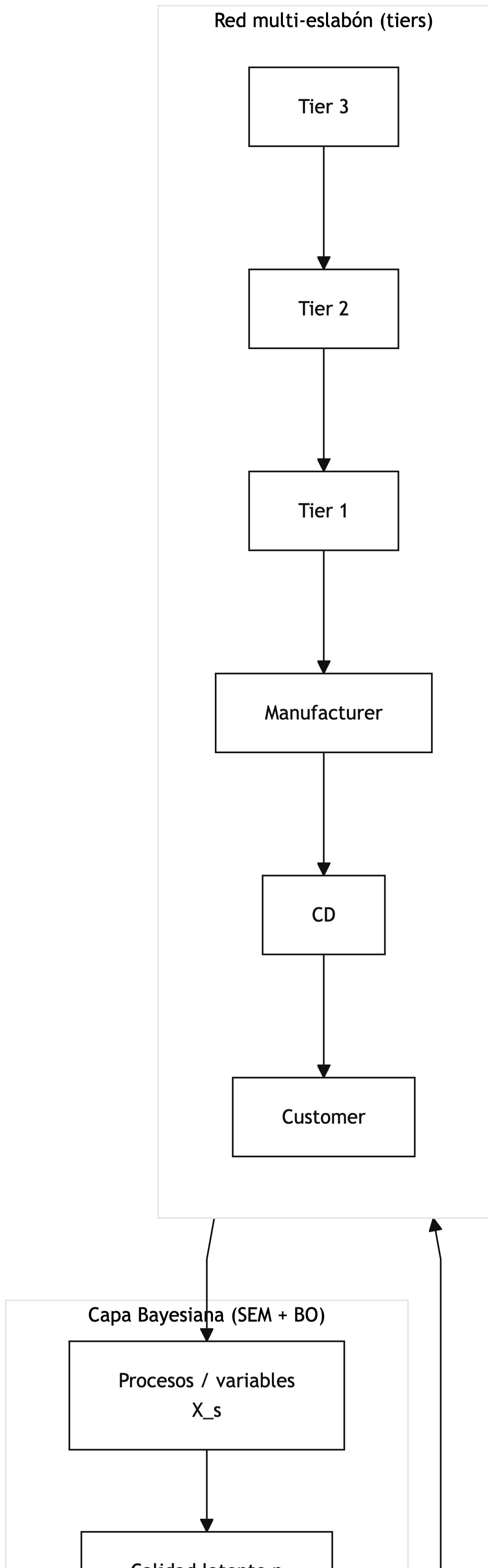


Figure 4: Posterior mean vs true parameter values.

47 7 Integrated Architecture (Conceptual Diagram)

```
%%{init: {
  "theme": "base",
  "themeVariables": {
    "background": "#ffffff",
    "primaryColor": "#ffffff",
    "primaryTextColor": "#111111",
    "primaryBorderColor": "#111111",
    "lineColor": "#111111",
    "fontSize": "12px"
  },
  "flowchart": { "curve": "linear" }
}}%%

flowchart TB

  subgraph S["Red multi-eslabón (tiers)"]
    direction TB
    T3["Tier 3"] --> T2["Tier 2"] --> T1["Tier 1"] --> M["Manufacturer"] --> D["CD"] -->
  end

  subgraph B["Capa Bayesiana (SEM + BO)"]
    direction TB
```

```
    X["Procesos / variables<br/>X_s"] --> ETA["Calidad latente  "]
    ETA --> Y["Criterios Y_{c,k}"]
    Y --> SC["Satisfacción S_c"]
    SC --> U["Utilidad económica U"]
  end

  S --> X
  U --> MILP["MILP: selección de enlaces z<br/>y flujos q"]
  MILP --> S
```


Red multi-eslabón (tiers)
Tier 3
Tier 2
Tier 1
Manufacturer
CD
Customer
Capa Bayesiana (SEM + BO)
Procesos / variables
X_s

8 Customer Satisfaction and Endogenous Demand

Customer satisfaction for product p and customer c is defined as:

$$S_{c,p} = \sum_{k=1}^{K} w_{c,k} \mathbb{E}[Y_{c,p,k}] \tag{3}$$

where:

- $w_{c,k}$ represents the importance weight of criterion k for customer c,
- $\sum_k w_{c,k} = 1$.

Demand is assumed to be endogenous and driven by satisfaction:

$$d_{c,p} = d^0_{c,p} + \alpha_{c,p} S_{c,p} \tag{4}$$

where:

- $d^0_{c,p}$ is the baseline demand,
- $\alpha_{c,p}$ measures sensitivity of demand to satisfaction.

9 Economic Utility Function

Total economic utility is defined as:

$$U = \sum_{c,p} \left(price_{c,p} \cdot q_{c,p}\right) + \gamma \sum_{c,p} S_{c,p} q_{c,p} - C \tag{5}$$

where:

- $q_{c,p}$ is the quantity delivered to customer c,
- γ represents the economic impact of satisfaction,
- C is total supply chain cost.

10 Cost Structure

Total cost is composed of:

$$C = C^{var} + C^{struct} \tag{6}$$

Variable cost:

$$C^{var} = \sum_{(i,j),p} c^{arc}_{i,j,p} q_{i,j,p} \tag{7}$$

Structural cost:

$$C^{struct} = \sum_{(i,j),p} f_{i,j} z_{i,j,p} \tag{8}$$

11 MILP Formulation

The optimization problem is:

$$\max U$$

Subject to:

Flow conservation:

$$\sum_i q_{i,j,p} = \sum_k q_{j,k,p}$$

Capacity constraints:

$$\sum_{j,p} q_{i,j,p} \leq Cap_i$$

Link activation constraints:

$$q_{i,j,p} \leq M z_{i,j,p}$$

Binary structure:

$$z_{i,j,p} \in \{0,1\}$$

12 Bayesian Optimization Layer (GP + Expected Improvement)

This section formalizes the Bayesian Optimization (BO) layer used to learn or adapt decision parameters (e.g., customer weights, process targets, or policy parameters) that affect satisfaction and the downstream MILP objective. Let $\theta \in \Theta \subset \mathbb{R}^d$ denote the vector of tunable parameters. Examples include:

- preference weights $\theta = \{w_{c,k}\}$,
- process-control targets $\theta = \{x^\star_{s,j}\}$,
- economic trade-off parameters $\theta = \{\gamma, \alpha_{c,p}\}$,
- or any calibration vector that impacts expected satisfaction and utility.

We define the (black-box) BO objective as the **expected economic utility** induced by θ:

$$f(\theta) \;=\; \mathbb{E}\Big[\,U(q^\star(\theta), z^\star(\theta);\,\theta)\,\Big], \tag{9}$$

where $(q^\star(\theta), z^\star(\theta))$ is the MILP optimal solution under parameterization θ, and the expectation is taken with respect to the Bayesian layer uncertainty (SEM posterior and any stochastic components).

12.1 Gaussian Process Surrogate

At iteration n, we have evaluated f at $\mathcal{D}_n = \{(\theta_i, y_i)\}_{i=1}^{n}$, where:

$$y_i \;=\; f(\theta_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2). \tag{10}$$

We place a Gaussian Process prior on f:

$$f(\theta) \sim \mathcal{GP}\big(m(\theta),\, k(\theta, \theta')\big), \tag{11}$$

commonly using a constant mean $m(\theta) = m_0$ and an RBF kernel:

$$k(\theta, \theta') \;=\; \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d} \frac{(\theta_\ell - \theta'_\ell)^2}{\rho_\ell^2}\right). \tag{12}$$

Given $\mathcal{D}_n$, the GP posterior at a candidate θ is Gaussian:

$$f(\theta) \mid \mathcal{D}_n \;\sim\; \mathcal{N}\big(\mu_n(\theta),\, \sigma_n^2(\theta)\big), \tag{13}$$

with standard expressions:

$$\mu_n(\theta) \;=\; m(\theta) + \mathbf{k}_n(\theta)^\top \big(\mathbf{K}_n + \sigma_\epsilon^2 \mathbf{I}\big)^{-1} (\mathbf{y} - \mathbf{m}_n), \tag{14}$$

$$\sigma_n^2(\theta) \;=\; k(\theta, \theta) - \mathbf{k}_n(\theta)^\top \big(\mathbf{K}_n + \sigma_\epsilon^2 \mathbf{I}\big)^{-1} \mathbf{k}_n(\theta), \tag{15}$$

where $\mathbf{K}_n = [k(\theta_i, \theta_j)]_{i,j}$, $\mathbf{k}_n(\theta) = [k(\theta_1, \theta), \dots, k(\theta_n, \theta)]^\top$, $\mathbf{y} = [y_1, \dots, y_n]^\top$, and $\mathbf{m}_n = [m(\theta_1), \dots, m(\theta_n)]^\top$.

12.2 Expected Improvement Acquisition

Let $f_n^{\max} = \max_{i \le n} y_i$ be the best observed value so far. The Expected Improvement (EI) acquisition for maximization is:

$$\mathrm{EI}_n(\theta) \;=\; \mathbb{E}\Big[\max\big(0,\, f(\theta) - f_n^{\max} - \xi\big) \;\big|\; \mathcal{D}_n\Big], \tag{16}$$

where $\xi \ge 0$ controls exploration. Under the GP posterior @eq-gp-post, EI has closed form. Define:

$$Z(\theta) \;=\; \frac{\mu_n(\theta) - f_n^{\max} - \xi}{\sigma_n(\theta)}, \tag{17}$$

then:

$$\mathrm{EI}_n(\theta) \;=\; \big(\mu_n(\theta) - f_n^{\max} - \xi\big)\Phi\big(Z(\theta)\big) + \sigma_n(\theta)\phi\big(Z(\theta)\big), \tag{18}$$

with $\Phi(\cdot)$ and $\phi(\cdot)$ the standard normal CDF and PDF.

The BO iteration selects the next evaluation point by:

$$\theta_{n+1} \;=\; \arg\max_{\theta \in \Theta} \; \mathrm{EI}_n(\theta). \tag{19}$$

12.3 Coupling with the Bayesian SEM and MILP

For each candidate θ evaluated during BO, the workflow is:

1. **SEM posterior propagation:** draw $(\beta, \lambda, \sigma) \sim p(\cdot \mid \text{data})$ and compute $\mathbb{E}[Y_{c,p,k} \mid X, \theta]$ (or Monte Carlo estimates).
2. **Satisfaction / demand update:** compute $S_{c,p}(\theta)$ and any derived demand parameters.
3. **MILP solve:** solve the MILP to obtain $(q^\star(\theta), z^\star(\theta))$.
4. **Utility evaluation:** compute $y = f(\theta)$ as the expected (or Monte Carlo averaged) utility.

This closes the Bayesian loop: BO learns θ that maximizes the satisfaction-driven expected utility under uncertainty.

13 Conclusion

The proposed framework integrates:

- Bayesian Structural Equation Modeling
- Multi-tier supply chain structure
- Endogenous satisfaction-driven demand

- Mixed-integer network optimization

forming a unified architecture for economic utility maximization.

::contentReferenceoaicite:0