An interactive optimization framework for incorporating a broader range of human feedback into stochastic multi-objective mixed integer linear programs

Damsara Jayarathne[a] Sandipan Mishra[a] John E. Mitchell[b] Jennifer Pazour[c]

[a] *Mechanical, Aerospace, and Nuclear Engineering Dept.,*
[b] *Mathematical Sciences Dept.,*
[c] *Industrial and Systems Engineering Dept., Rensselaer Polytechnic Institute, 110 8th street, Troy, 12180, NY, United States of America*

Abstract

Interactive optimization leverages the strengths of optimization frameworks alongside the expertise of human users. Prior research in this area tends to either ask human users for the same type of information, or when varying information is requested, users must manually modify the optimization model directly. These limitations restrict the incorporation of wider human knowledge into the problem domain and demand a higher level of optimization expertise from the user. To address these challenges, we propose a new iterative interactive optimization framework that enables human users to answer multiple types of questions, with their input being transformed to create a higher fidelity stochastic multi-objective mixed integer linear programming model (SM-MILP). The goal of the framework is to arrive at an SM-MILP that can recommend design solutions satisfactory to the human designer. Our framework can ask the human user targeted questions to extract important information while reducing the solution's performance uncertainty through a Conditional Value at Risk (CVaR) formulation. The conducted computational experiments on a supplier selection problem demonstrate that this iterative framework can reduce the reality gap while converging faster to the ground truth solution and reducing the confidence interval length.

Keywords: Stochastic mixed integer linear programs, multi-objective, interactive optimization, human-in-the-loop, CVaR

1. Introduction

Stochastic multi-objective mixed integer linear programming (SM-MILP) models are optimization models that recommend the best systematic decisions when there are combinatorially many options, parameter uncertainty, multiple decision criteria, and decision trade-offs due to systemic interactions, and constraints on finite resources. SM-MILP models are commonly used to guide a wide range of design problems. For example, in supply chain design, determining what network of suppliers and what contracts should be planned to supply and manufacture commodities is often aided by SM-MILP models (1; 2; 3). Not only are there combinatorially many different ways to design such a network, such decisions tend to be based on multiple criteria (e.g., minimizing costs, maximizing resilience, maximizing on-time performance), and the decisions are strategic in nature, which means they must be made and locked into with only an uncertain forecast of the future supplier performance and cost and future demand.

The construction of such models are subjective; they only capture the features and information that they were designed to represent. This means the optimization model itself may not capture a certain criterion or constraint that is critical to the decision-making process. On the other hand, it is often impractical to build a model that captures all the problem's features. This gap between what is represented in the model and what occurs, in reality, is known as the so-called "reality gap". At some level, there will always be a reality gap, as by definition a model is only a representation of reality. However, especially for critical design decisions, reducing the reality gap is important and one of the most common methods to reduce this reality gap is to use an interactive approach that dynamically integrates a human user (HU) into the optimization-modeling process (4).

Interactive optimization (IO) incorporates advanced optimization techniques into decision support tools with which humans interact. They have broad applications as they can help to synergistically combine the power of human knowledge and the power of advanced optimization approaches (4). Humans typically have access to important context and domain knowledge (common knowledge, rules, and experience gained through sustained practices). In contrast, humans are challenged with computing complex performance metrics and efficiently identifying designs when there are exponentially many different solutions. While there is a rich literature in IO, the majority of the techniques focus on eliciting a narrow range of data from a human, i.e., most state-of-the-art IO techniques focus on asking HU to only update their preferences (5).

In this paper, we propose a new IO framework for SM-MILPs to elicit and incorporate a broader range of information from the HU into an SM-MILP. It consists of an optimization model in the form of an SM-MILP which provides the optimal solution to the HU, given the currently modeled mathematical structure and input data. The refinement takes in the model-based solution and presents possible questions to the HU. Furthermore, the refinement layer needs to interpret the method of updating the optimization model based on the HU's responses. In the proposed framework, the refinement can update data or change the structure of the mathematical formulation of the optimization problem based on human feedback and interpretation. This is an iterative process which repeats until a terminating criterion is met.

This methodology has the potential to execute varying refinement actions by allowing the framework to ask the HU different questions, which inquire about a broad range of knowledge the HU may possess. The framework also uses information from the mathematical formulation, specifically dual variable values obtained from the LP-relaxation of the SM-MILP, to determine the most valuable information to be extracted from the user (which we term as math-informed targeted questions). A key challenge with creating a method that elicits a broader range of knowledge from an HU is how to incorporate the response from an HU and use it to update the math model. To address this challenge, we present a Monte Carlo-based framework to transform human responses into input data that can be ingested into the SM-MILP via a scenario-based optimization model formulation. Another challenge is to arrive at a solution that is satisfactory to the HU, and one key metric is whether the HU has adequate confidence in the model's output. To address this, we introduce the formulation of Conditional Value at Risk (CVaR) to minimize the risk associated with the solution produced by the optimization model. We demonstrate the framework can refine the solution to a supplier selection problem by exploiting the iterative interaction between the HU and the optimization model.

2. Related work

IO refers to a specialized area of study that empowers a HU to actively engage in the optimization process by providing input and guidance in an iterative manner (4). IO has significantly advanced over the past twenty years, with a plethora of research surveyed in the following review articles (4; 5). While the interaction between the HU and the optimization process, theoretically, can take place at any point during the process, the existing work focuses on the process where the HU provides feedback after the optimization model is solved, allowing the HU to provide feedback to shape the outcome according to their additional knowledge.

In IO, the HU can be either an adjuster, or enricher, or both (4; 5). An adjuster provides feedback on the input parameter values or expands the elements of proposed sets. Therefore, an adjuster will only be changing or refining the data associated with the model. The large majority of existing IO frameworks only allow a HU to serve as an adjuster, and include for example research in (6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20; 21). Although refining data does not change the model formulation's functional form, it can lead to new data or data with better quality. A special type of data refinement commonly used in multi-objective optimization, where the HU is asked to provide more information to update the preference weights across multiple criteria, is known as a preference update. For example, IO strategies that use reference points are presented in (22). A multi-objective optimization model is solved, starting with an approximated Pareto optimal set. Then, the HU is asked to classify objective functions, specify upper and lower bounds of the sub-objectives and weighting coefficients. A subproblem is formulated taking this information into account, of which the solution is presented to the HU. Once the HU is satisfied, the algorithm is terminated. Similarly, a reference point method have been employed in (23). In this work, the HU provides different reference points so that they can explore the Pareto frontier through several interactions. However, it is worth noting that these methods require the knowledge of a suitable reference point. Notably, the preference update is the most studied mode of refinement in the IO literature with HU as adjuster (24; 25; 26; 27; 18; 17; 10; 11; 12; 13; 28; 29). Further, all these approaches use a static query, where the query presented to the HU does not change from one iteration to the next. Therefore, there is a need for IO methods that can incorporate a broader range of human knowledge into optimization models.

The HU is deemed an enricher if the provided feedback results in changes to the structure of the optimization model. In general, these structural changes are in the form of adding or removing constraints/ objectives or changing the sets defined in the formulation of the problem. When a HU plays the role of an enricher, they provide feedback that the model structure may be incomplete as it does not capture all the aspects of the HU's constraints or objective criteria. Methodologies that enable a HU to serve as an enricher are rare in the current IO literature (5).

The following papers present IO methodologies, where the HU plays the role of an enricher. IO has been used for planning bulk deliveries in (20), in which the HU can adjust the constraints *manually* at each iteration. To modify the constraints and check for infeasible solutions, the HU must be a domain expert as well as an optimization expert. This is common in the other papers where the HU serves as an enricher. For example, a shift scheduling problem is solved using an IO framework in (30) where at each iteration, the HU is provided the solution, and the HU can manually change the soft constraints in the optimization problem. IO has been utilized to solve a layout optimization problem in (31). The method requires the HU to inspect candidate solutions at each iteration and then the HU can manually modify constraints when they want to incorporate additional information. These changes are then included in the refined optimization model. In (32), IO is used to solve a layout planning problem which is solved by a genetic algorithm that converges to a local minimum. The HU is presented with this local solution and allowed to directly change the solution according to the HU's preference and expertise. An IO method to reoptimize the components used in additive manufacturing is developed in (33). This methodology presents the HU with a candidate solution for the formulated multi-objective deterministic linear program and the HU can *manually* edit the structural components or specify new design requirements as constraints in the problem formulation.

While most IO approaches focus on deterministic models, some work has focused on decision-making under uncertainty. For example, uncertainty has been incorporated into IO frameworks applied to the manufacturing processes in (34; 35), which uses a stochastic optimization model to capture operational risk by minimizing parameter uncertainty in the objective. Furthermore, an IO scheme which comprises stochastic optimization modeling, user interaction, and policy analysis has been employed to conduct energy management (36). Moreover, (37) has developed an IO methodology that captures stochasticity in integer programming problems by leveraging chance constraints. These works highlight the importance of capturing uncertainty in applying the IO method to real-world applications. However, these methods do not query the HU for uncertain parameters in the problem formulation.

2.1. Summary of the Literature

While the IO domain has expanded, most developed methodologies only use the HU as an adjuster to extract more accurate parameter values in the model, and most methodologies are designed to extract a very narrow range of information from the HU. In fact, in most existing IO frameworks, the HU is only asked to provide preference information to arrive at a satisfactory solution in a multi-objective optimization setting. Generally, HUs have additional context that could be used to update a broader range of data inputs or to dynamically add, remove, or change constraints and objectives related to the problem at hand. We are not aware of approaches asking the HU for multiple types of updated information that are done in a guided way. While IO methods that view the HU as an enricher allow a wider range of human knowledge to be incorporated into the optimization problem, the existing enricher IO literature requires the HU to decide which aspect of the problem needs adjustments and requires the HU to make changes directly to the optimization model. This requires the HU to be a domain expert and to have significant optimization expertise. In our work, we are interested in developing a methodology where the HU is a domain expert, but not an optimization expert.

Thus, our approach contributes to the IO literature by expanding the information elicited and incorporated from the HU into a SM-MILP by (i) allowing for multiple types of questions to be answered by the HU and the type of feedback elicited from the HU to change over iterations; (ii) allowing the HU to be only a domain expert, and not requiring the HU to be an optimization expert; instead of having the HU directly make adjustments to the optimization formulation, the methodology queries domain information from the HU, and the methodology transforms this information into ways to change the optimization model's structure or data inputs; and (iii) queries the HU for uncertain parameters.

3. Problem statement

Consider a complex design problem that can be posed as a stochastic multi-objective mixed integer linear programming (SM-MILP) optimization problem. Given this is a model (i.e., a representation of the problem of interest), the problem formulation and the data will be different than the characteristics pertaining to the real-world design problem. In this work, we are motivated to create a methodology that can iteratively elicit and incorporate a broad range of information and knowledge of an experienced HU into the optimization model formulation to produce a design solution that is satisfactory to the HU. This is achieved by iteratively querying the HU. The satisfaction of the HU is threefold. 1) The solution should be realistic to the HU; 2) the expected performance of the solution should be adequate to the HU across all evaluation criteria; and 3) The certainty or confidence in the solution's performance across potentially multiple criteria should be high enough for the HU to pursue the recommended design.

The aim of our framework is to converge as quickly as possible to a satisfactory solution without overburdening the HUs. Furthermore, our framework should be computationally tractable to solve and applicable when the HU is merely a domain expert but not an optimization expert. This means the framework must be able to query the HU for additional knowledge about the design problem, but should not require the HU to directly contribute in making/ altering the structure/ constraints of the problem formulation. This requires a way to first interpret broad feedback and translate it into a format that can be fed into the optimization framework. This is particularly challenging since the HU is asked to provide feedback not just on one aspect but on multiple aspects of the problem.

4. Proposed methodology

The proposed IO framework for refining SM-MILP models by posing targeted and different queries to the HU at each iteration is given in Fig. 1. The framework starts with an initial SM-MILP solved via a scenario-based model. After solving the scenario-based SM-MILP model, the framework asks the HU to evaluate the recommended design solution, its performance, and the performance confidence intervals across multiple criteria. These values are presented to the HU, who is asked whether or not the solution and its performance is satisfactory. If not, the HU is then presented with a set of targeted refinement questions. The HU responds to these questions, and these responses are interpreted so that they can be meaningfully and tractably incorporated back into the optimization model, either as an update to the model's structure or updated data inputs. The methodology is iterative, i.e., the *modified* mathematical model is solved at each time step k and this process repeats until the HU is satisfied and recommends termination.

4.1. Scenario-based model

Two-stage SM-MILPs are a common approach to modeling uncertainty in optimization problems (38). Specifically, stage one decisions are made "now" and stage two decisions are made after uncertainty in the system has been realized. In this work, we consider a multiple objective version, where these decisions are made by optimizing a weighted sum of expected performance across the

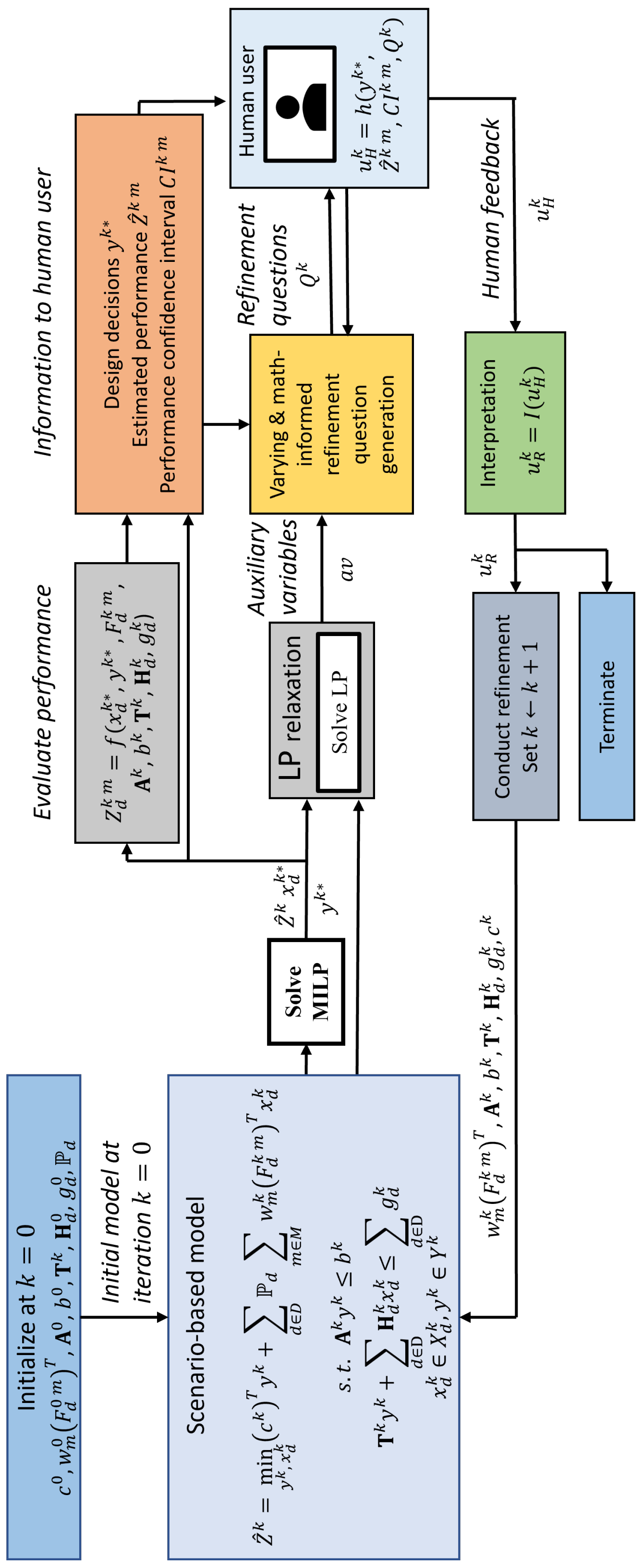
Initialize at $k=0$
$c^0, w_m^0 (F_d^{0\,m})^T, \mathbf{A}^0, b^0, \mathbf{T}^k, \mathbf{H}_d^0, g_d^0, \mathbb{P}_d$
Initial model at iteration $k=0$
Scenario-based model
$\hat{Z}^k = \min_{y^k, x_d^k} (c^k)^T y^k + \sum_{d\in D} \mathbb{P}_d \sum_{m\in M} w_m^k (F_d^{k\,m})^T x_d^k$
$s.t.\ \mathbf{A}^k y^k \le b^k$
$\mathbf{T}^k y^k + \sum_{d\in D} \mathbf{H}_d^k x_d^k \le \sum_{d\in D} g_d^k$
$x_d^k \in X_d^k, y^k \in Y^k$
Solve MILP
$\hat{Z}^k\ x_d^{k*}$
y^{k*}
LP relaxation
Solve LP
Auxiliary variables
av
Evaluate performance
$Z_d^{k\,m} = f(x_d^{k*}, y^{k*}, F_d^{k\,m}, \mathbf{A}^k, b^k, \mathbf{T}^k, \mathbf{H}_d^k, g_d^k)$
Information to human user
Design decisions y^{k*}
Estimated performance $\hat{Z}^{k\,m}$
Performance confidence interval $CI^{k\,m}$
Varying & math-informed refinement question generation
Refinement questions Q^k
Human user
$u_H^k = h(y^{k*}, \hat{Z}^{k\,m}, CI^{k\,m}, Q^k)$
Human feedback
u_H^k
Interpretation
$u_R^k = I(u_H^k)$
u_R^k
Conduct refinement
Set $k \leftarrow k+1$
Terminate
$w_m^k (F_d^{k\,m})^T, \mathbf{A}^k, b^k, \mathbf{T}^k, \mathbf{H}_d^k, g_d^k, c^k$

Figure 1: Proposed interactive optimization framework

set of design criteria $m \in O$ with a set of scenarios, denoted as $d \in D$, that represent the uncertainty in input parameter values.

Our framework solves such a model at each iteration k, and we present a general scenario-based SM-MILP at iteration k in (1). The objective function achieves the minimum expected weighted performance across the set of criteria $m \in O$. Both first-stage (y^k) decision variables in iteration k and second-stage (x_d^k) decision variables obtained in iteration k for scenario $d \in D$ can take on a mix of either continuous binary values. Let I and J be the set of all the first-stage and second-stage variables, respectively. We define $I_1 \subseteq I$ and $J_1 \subseteq J$ for binary variables in the first and second stages, respectively.

$$\begin{aligned}
&\hat{Z}^k = \min_{y^k,\, x_d^k} (c^k)^T y^k + \sum_{d \in D} \mathbb{P}_d \sum_{m \in O} \Omega_m^k (F_d^{k\ m})^T x_d^k \\
&\text{s.t.} \quad \mathbf{A}^k y^k \leq b^k \\
&\quad \mathbf{T}^k y^k + \sum_{d \in D} \mathbf{H}_d^k x_d^k \leq \sum_{d \in D} g_d^k \\
&\quad x_d^k \in X_d^k, \quad X_d^k = \{x_d^k | x_{id}^k \in \{0,1\}, \forall i \in I_1^k, x_d^k \geq 0\} \\
&\quad y^k \in Y^k, \quad Y^k = \{y^k | y_j^k \in \{0,1\}, \forall j \in J_1^k, y^k \geq 0\}
\end{aligned} \tag{1}$$

$\hat{Z}^k$ refers to the optimal objective function value at iteration k. We denote the optimal solution obtained in iteration k as y^{k*}, x_d^{k*}. Input data to this model are required in different dimensions and will change across iterations. This includes $\mathbb{P}_d$, which is the probability of scenario $d \in D$ occurring in iteration k, and Ω_m^k, the preference weight assigned for each objective $m \in O$ in iteration k. We note that the probability of scenario d occurring is equally likely. Additional input data at iteration k includes vectors: c^k which is the coefficient vector of the first-stage decision variables in the objective function, $F_d^{k\ m}$ is the coefficient vector of the second-stage decision variables in scenario $d \in D$ for objective $m \in O$ for iteration k, b^k is a vector containing the right-hand side constants for the first-stage variables, and g_d^k is a vector containing the right-hand side constants for the second-stage variables in scenario $d \in D$. $\mathbf{A}^k$ and $\mathbf{T}^k$ are input data in matrix form representing the coefficients for first-stage variables in constraints not influenced by uncertainty (captured by scenarios $d \in D$). Finally, $\mathbf{H}_d^k$ represents coefficients for the second-stage variables in constraints that are influenced by uncertainty.

Our methodology requires an initial formulation of the problem, which we denote as the model solved at iteration $k = 0$. Then at each iteration k, the model is updated based on the HU's feedback and its interpretation. The model is solved using an off-the-shelf solver at each iteration k which delivers the optimal solution for the model formulated at iteration k that comprises of the expected performance $\hat{Z}^k$ and the optimal design decisions, y^{k*}, as well as associated second stage variables needed to deal with the uncertainty present in scenario $d \in D$, x_d^{k*} .

4.2. *LP relaxation*

LP relaxation is employed to generate auxiliary information used to compose targeted refinement questions that access the HU information that has more value in the iterative design process. The optimal solution of the integer variables denoted as x_{id}^{k*} and y_j^{k*} from solving (1) at iteration k are fixed at their optimal values and then we solve the resulting linear programming formulation. This is executed in the LP relaxation block. The solved linear program provides a dual variable value for each remaining continuous constraint.

4.3. *Evaluating performance*

The scenario-based model solved at iteration k determines a single set of design solutions (i.e., y^{k*} values) based on an expected performance across all scenarios. Yet, the performance of the recommended solution changes from one scenario to another as well as from one objective to another. Therefore, at iteration k we evaluate the performance of each scenario $d \in D$ for each sub-objective $m \in M$ as $Z_d^{k\ m}$ as

$$Z_d^{k\ m} = (c^k)^T y^k + w_m^k (F_d^{k\ m})^T x_d^k. \tag{2}$$

Then, using these $Z_d^{k\ m}$ values, we calculate performance confidence intervals, $CI^{k\ m}$ for each sub-objective m at iteration k.

4.4. *Information displayed to the HU*

The HU is provided with information that facilitates the refinement actions. This information consists of the design decision values y^{k*} from solving the model at iteration k, its estimated performance in each sub-objective $Z^{k\ m}$, and the performance confidence interval in each sub-objective $CI^{k\ m}$. This information is displayed to the HU, along with a set of refinement questions detailed in Section 5.3.

4.5. Refinement questions

In our framework, the HU is queried different questions which enables the HU to provide a wide range of knowledge. The set of refinement questions at iteration k, Q^k, are presented to the user along with $\hat{Z}^{k\ m}$, y^{*k}, and $CI^{k\ m}$ so that the HU provides feedback u_H^k so that the iterative process arrives at a solution that is satisfactory to the HU. These different refinement questions ask the HU different inquiries adapted to the current state of the model and the outputs at each iteration k. These questions are personalized to the problem at hand, but in a general sense they ask for better-quality information, since the HU might not have provided all the necessary information at the very beginning of the process. For example, in cases where the scenario-dependent input data is not provided directly for all the scenarios, the algorithm can ask for additional information, and then use the HU's feedback to generate appropriate data for all the scenarios. Math-informed refinement questions utilize the information from the mathematical formulation specifically dual variables (Sec. 4.2) to ask targeted questions that focus on extracting the most valuable information from the user (meaning information that could update the binding constraints of the LP relaxation, which we determine using the dual variable values). This allows us to personalize the questions displayed at each iteration.

4.6. Human feedback

The human feedback denoted as u_H^k, corresponds to feedback provided by the HU in response to the refinement questions and the information displayed to the user at iteration k. In general, the HU feedback is typically in response to the design decisions y^{k*}, the estimated performance values $\hat{Z}^{k\ m}$, the performance confidence intervals $CI^{k\ m}$, and the refinement questions Q^k. Thus, we denote it as a function of these inputs by $u_H^k = h(y^{k*}, \hat{Z}^{k\ m}, CI^{k\ m}, Q^k)$. As the framework's goal is to query a wide range of data from the HU and our framework is capable of asking different questions, the human feedback can also be in different forms (an illustration of this is given in Section 6.2). This leads to the need to interpret the feedback and translate it into a format that can be inputted into the optimization model, which we explain in the next section.

4.7. Interpretation and refinement actions

The interpretation block interprets the human feedback in such a way that it can be used in a meaningful way to increase the modeling fidelity of our scenario-based model. As shown in Fig.

1, it outputs the refinement at iteration k as $u_R^k = I(u_H^k)$ where function I takes human feedback u_H^k as arguments. Thus, the refinement actions u_R^k can be classified into three types: 1) update input data, 2) update the mathematical structure of the stochastic model, or 3) if further improvement is not required, terminate. For all of these refinement actions, we provide illustrative examples in Section 6.2.

4.7.1. Update input data

The update input data refinement action uses data provided by the HU to update input data, either *directly* or *indirectly*. Direct data is directly applied in the model via updates to input parameters, w_m^k, $A^k, b^k, \mathbf{T}^k$, $\mathbf{H}_d^k$, or g_d^k. On the other hand, indirect data must go through an additional *interpretation* procedure before it can be incorporated into the model.

Interpretation: The interpretation function I is chosen based on the relevant application. For instance, in some cases, the *Monte Carlo* (39) method can be a useful method of interpretation. In such a case, the human input is interpreted as constraints and relationships captured in the Monte Carlo simulation, which then translates human input data into scenario-based data capturing uncertainties in some of the input data such as $\mathbf{H}_d^k$, or g_d^k. Thus, even though the scenario data is randomly generated at each iteration, questions are asked of the HU in such a way as to incorporate more structure into the Monte Carlo simulation. This results in the scenario-based data being generated in ways that have higher fidelity and that capture this human input. An example explaining the details of the implementation of the Monte Carlo method for the supplier selection problem is given in Section 6.2.

4.7.2. Designing a solution with reduced risk via structural modifications

To accommodate the need for our methodology to produce a design solution where the certainty or confidence in the solution's performance across potentially multiple criteria should be high enough for the HU to pursue the recommended design, we design a refinement mechanism able to produce solutions that have lower risk (i.e., have smaller estimated confidence interval widths across a given performance criteria). Specifically, we employ a CVaR formulation (40) which quantifies the average loss over unlikely scenarios beyond a confidence interval. Furthermore, it ensures that the reliability of the network is improved by minimizing the extreme losses in the tail of a distribution of interest. Notably, this risk aversion refinement is a structural modification that interprets the human feedback and then uses it to modify the model's mathematical structure by adding new constraints and an additional sub-objective, with details described in Section 6.2. This means that our methodology enables HU to serve as enrichers, not just adjusters.

Interpretation: In the case of risk aversion, the interpretation would be to assign the relevant variable specified by the HU in the optimization model.

5. Demonstration problem: A supplier selection problem

The goal of this section is to introduce a problem that we use in Section 6 to demonstrate our framework on. In general, supply chain design problems are commonly guided through the use of SM-MILP (41; 42). We deploy the proposed methodology to a multi-echelon capacitated supplier selection problem.

This problem takes as input a given bill of materials, final product demand forecast for a set of customer locations, a set of potential suppliers (and their associated resources and attributes), and determines which suppliers to source which materials, physical components, and services from, as well as how much to source, and what is the resulting supplier network structure.

5.1. Scenario-based model of the supplier selection problem

In this section, we use set notation to mathematically formulate the supplier selection problem as a scenario-based, two-stage SM-MILP. The first-stage decision variables are binary variables deciding whether to establish a contract with a supplier or not. The second-stage decision variables capture for a given scenario, the shipment flows, unmet demand, and on-time performance (which occur after the stochastic values in demand and other supply chain resources have materialized). We refer the reader to Appendix C for the structure of the network. The following sets defines the problem formulation.

Sets

S	set of all suppliers	P_e	set of end commodities
V_0	set of customer nodes	P_r	set of raw commodities
V	set of suppliers and customers	S_1	set of echelon 1 suppliers
RS	set of resources	S_2	set of echelon 2 suppliers
P	set of commodities	D	set of scenarios
M	set of sub-objectives		

Note that echelon 2 suppliers supply raw commodities which are processed at echelon 1 suppliers to produce end commodities. These sets have the following relationships with each other: $S_1 \subset S, S_2 \subset S, V = S \cup V_0, P = P_r \cup P_e$.

Input Parameters

$$I_{ipj} = \begin{cases} 1, \text{if commodity } p \in P \text{ sent from supplier } i \in S \text{ to } j \in V \\ 0, \text{otherwise} \end{cases}$$

α_{ip}^{r}	Estimated rate of resource $r \in RS$ consumed by supplier $i \in S$ for commodity $p \in P$
u_i^r	Capacity of supplier $i \in S$ to handle resources $r \in RS$
f_i	Fixed cost of establishing a contract with supplier $i \in S$
E_{jp}^{d}	Customer demand for end product $p \in P_e$ at customer node $j \in V_0$ for scenario $d \in D$
c_{ipj}^{d}	Cost to make one unit of commodity $p \in P$ and ship from supplier $i \in S$ to node $j \in V$ in scenarios $d \in D$
q_{ip}	Expected number of units requiring rework of commodity $p \in P$ at supplier $i \in S$
θ_{pl}	Number of units required of commodity $p \in P$ to make one unit of $l \in P$
τ_{ipj}	Time to ship commodity $p \in P$ from supplier $i \in S$ to node $j \in V$
$\overline{M}$	Big M constants related to choosing the maximum time to process
M^{λ}	Big M constants related to choosing ϕ_{ij} $i \in S,\ j \in V$
T_c^{Lead}	Required lead time at customer node $c \in V_0$
M^{lambda}	Big M constants related to customer node $c \in V_0$
$\mathbb{P}^d$	Probability of scenario $d \in D$ occurring
$\Delta_{1,2,3}$	Coefficient
$\Omega_{1,2,3}$	Weight

Decision Variables

$$y_i = \begin{cases} 1, \text{if contract is established with supplier } i \in S \\ 0, \text{ otherwise} \end{cases}$$

w_{ipj}^d	Number of units of commodity $p \in P$ shipped from $i \in S$ to $j \in V$ for scenario $d \in D$
x_{jp}^d	Percentage of unmet demand at customer $j \in V_0$ of end product $p \in P_e$ for scenarios $d \in D$
$r_{i,IN}^{p,l,d}$	Total inflow of commodity $p \in P$ at supplier $i \in S_1$ to be converted to $l \in P$ for scenario $d \in D$
$r_{i,OUT}^{p,d}$	Total outflow of commodity $p \in P$ at supplier $i \in S_1$ for scenario $d \in D$

$$z_{ipj} = \begin{cases} 1, \text{if commodity } p \in P \text{ sent from supplier } i \in S \text{ to } j \in V \\ 0, \text{otherwise} \end{cases}$$

$$zz_{ij} = \begin{cases} 1, \text{if any commodity is sent from supplier } i \in S \text{ to } j \in V \\ 0, \text{otherwise} \end{cases}$$

$$\phi_{ij} = \begin{cases} T_i, if zz_{ij} > 0 \text{ for } i \in S \text{ to } j \in V \\ 0, \text{otherwise} \end{cases}$$

λ_j Time of the last delivery to node $j \in V$ (starting from echelon 1)

$$h_{ipj} = \begin{cases} 1, \text{if sent commodity } p \in P \text{ from supplier } i \in S \text{ to } j \in V \\ 0, \text{otherwise} \end{cases}$$

T_j Maximum time to receive commodities at node $j \in V$ from the previous echelon

$$v_c = \begin{cases} 1, \text{if } \lambda_c - T_c^{Lead} > 0 \text{ for customers } c \in V_0 \\ 0, \text{otherwise} \end{cases}$$

$$\psi_c = \begin{cases} (\lambda_c - T_c^{Lead}), \text{if } v_c > 0 \text{ for customers } c \in V_0 \\ 0, \text{otherwise} \end{cases}$$

Objective

$$\min \quad \Omega_1\Delta_1\left(\sum_{i\in S} f_i y_i + \sum_{d\in D}\mathbb{P}^d \sum_{i\in S}\sum_{p\in P}\sum_{j\in V} c^d_{ipj} w^d_{ipj}\right) + \Omega_2\Delta_2 \sum_{d\in D}\mathbb{P}^d \sum_{j\in V_0}\sum_{p\in P_e} x^d_{jp} E^d_{jp} + \Omega_3\Delta_3 \sum_{c\in V_0}\psi_c \quad (3)$$

Constraints

$$\sum_{i\in S} w^d_{ipj} \geq E^d_{jp}(1-x^d_{jp}) \;\; \forall j\in V_0,\;\forall p\in P_e,\;\forall d\in D \quad (4)$$

$$\sum_{u\in S_2} w^d_{upi} = \sum_{l\in P} r^{p,l,d}_{i,IN} \;\; \forall i\in S_1,\;\forall p\in P_r,\;\forall d\in D \quad (5)$$

$$\sum_{l\in P} r^{p,d}_{i,OUT} = \sum_{g\in V} w^d_{ipg} \;\; \forall i\in S_1,\;\forall p\in P_e,\;\forall d\in D \quad (6)$$

$$r^{lp,d}_{i,IN} = \theta_{lp} r^{p,d}_{i,OUT} \;\; \forall i\in S_1,\;\forall p\in P(i),\;\forall l\in P,\;\forall d\in D \quad (7)$$

$$\sum_{j\in V}\sum_{p\in P}\alpha^r_{ip}\Big(\sum_{d\in D}\mathbb{P}^d w^d_{ipj} + q_{ip}y_i\Big) \leq u^r_i y_i \;\; \forall i\in S,\;\forall r\in RS \quad (8)$$

$$w^d_{ipj} \geq 0 \;\; \forall i\in S,\;\forall p\in P,\;\forall j\in V,\;\forall d\in D \quad (9)$$

$$w^d_{ipj} \leq \overline{M} I_{ipj} \;\; \forall i\in S,\;\forall p\in P,\;\forall j\in V,\;\forall d\in D \quad (10)$$

$$0 \leq x^{p,d}_j \leq 1 \;\; \forall j\in V_0,\;\forall d\in D,\;\forall p\in P_e \quad (11)$$

$$\sum_{d\in D} w^d_{ipj} \leq \overline{M} z_{ipj} \;\; \forall i\in S,\;\forall p\in P,\;\forall j\in V \quad (12)$$

$$T_j \geq \tau_{ipj} z_{ipj} \;\; \forall i\in S,\;\forall p\in P,\;\forall j\in V \quad (13)$$

$$T_j \leq \tau_{ipj} z_{ipj} + \overline{M}(1-h_{ipj}) \;\; \forall i\in S,\;\forall p\in P,\;\forall j\in V \quad (14)$$

$$\sum_{i\in S}\sum_{p\in P} h_{ipj} = 1 \;\; \forall j\in V \quad (15)$$

$$\sum_{p\in P} z_{ipj} \leq \overline{M} zz_{ij} \;\; \forall i\in S, \forall j\in V \quad (16)$$

$$\lambda_i = 0 \;\; \forall i\in S_2 \;\; (initialization) \quad (17)$$

$$\phi_{ij} \leq M^{\lambda} zz_{ij} \;\; \forall i\in S,\;\forall j\in V \quad (18)$$

$$\phi_{ij} \leq T_j \;\; \forall i\in S,\;\forall j\in V \quad (19)$$

$$\phi_{ij} \geq \lambda_i - M^{\lambda}(1-zz_{ij}) \;\; \forall i\in S,\;\forall j\in V \quad (20)$$

$$\lambda_j \geq \lambda_i + \phi_{ij} \;\; \forall i \in S, \;\; \forall j \in V \tag{21}$$

$$\lambda_c - T_c^{lead} \leq M^{lead} v_c \;\; \forall c \in V_0 \tag{22}$$

$$\psi_c \leq \overline{M} v_c \;\; \forall c \in V_0 \tag{23}$$

$$\psi_c \leq (\lambda_c - T_c^{Lead}) \;\; \forall c \in V_0 \tag{24}$$

$$\psi_c \geq (\lambda_c - T_c^{Lead}) - \overline{M}(1 - v_c) \;\; \forall c \in V_0 \tag{25}$$

The objective function in (3) is to minimize the total weighted expected supply chain costs across three sub-objectives of costs, unmet demand, and lateness. The costs are captured as fixed costs of awarding a contract plus the expected variable transportation cost which consists of the cost incurred to transport a commodity p from i^{th} node to j^{th} node. The expected unmet demand cost is a penalty for not being able to fulfill the demand requirements in a given scenario, which is captured via constraint (4). Constraints (5), (6), (7) are balance constraints that ensure the commodities coming into each node are sent out in appropriate ratios (after combining them in necessary ratios using θ_{pl}). Constraint (8) relates the amount of resource consumption and the resource capacities possessed by each supplier. Constraints (9) defines the upper bound of the number of commodities shipped whereas (11) defines the lower and upper bound on the percentage of unmet demand. In order to minimize the *makespan*, we employ a series of constraints from (12) to (25). z_{ipj} in (12) captures the paths that the commodities are supplied. (13) and (14) along with (15) computes the maximum time to receive commodities at node $j \in V$. (16) keeps track of any path that supplies any product and (17) initializes the time of last delivery at S_2 suppliers. (18)-(21) computes the time of last delivery to node $j \in V$. If the lead time prescribed at a particular customer node (T_c^{lead} for $c \in V_0$) is less than the computed time λ_j $j \in V$, then ψ_c $c \in V_0$ captures this phenomenon and it is penalized in the objective. This constraint is captured by constraints (22) - (25). Note that we construct the equations (12) to (25) in such a way that they do not depend on the scenarios so that we avoid the excess computational time caused by these constraints.

5.2. Different variations of the mathematical formulation

We introduce two variations of the mathematical model, which have increasing model fidelity, and are denoted as Model 1 and Model 2. Model 1 provides the basis for the initiation of the refinement framework (i.e., used at iteration $k = 0$). These models serve the purpose of updating the model structure when CVaR is utilized.

5.2.1. Model 1

In iteration $k = 0$, Model 1 minimizes (3) subject to constraints (4) - (25), and uses single point estimates for all stochastic input parameters by considering the cardinality of the scenario sets to be 1 (i.e., $|D| = 1$) and the probability of this single scenario to be 1, i.e., $\mathbb{P}^d = 1 \quad \forall d \in D$. In iteration $k > 0$, the cardinality of the scenario is set to predetermined values (i.e., $|D| > 1$).

5.2.2. Model 2

Model 2 minimizes (26) subject to constraints (4) - (25) and constraints (27)-(29). This formulation adjusts Model 1 to also include the conditional value at risk (40). Ω_4, Δ_4, and β are the weight, the normalizing coefficient, and the threshold of the worst outcome (required level of confidence), respectively, which are input parameters. μ is a continuous variable that represents the value at risk and R^d is a continuous variable that captures the conditional value at risk. In addition to what is captured in Model 1, Model 2 captures the risk of not meeting demand and optimizes for the expected worst outcome.

$$\begin{aligned} \min \quad & \Omega_1\Delta_1 \left(\sum_{d\in D}\sum_{i\in S} f_i y_i + \sum_{d\in D} \mathbb{P}^d \sum_{i\in S}\sum_{p\in P}\sum_{j\in V} c^d_{ipj} w^d_{ipj} \right) \\ & + \Omega_2\Delta_2 \sum_{d\in D} \mathbb{P}^d \sum_{j\in V_0}\sum_{p\in P_e} x^d_{jp} E^d_{jp} + \Omega_3\Delta_3 \sum_{c\in V_0} \psi_c + \Omega_4\Delta_4 \left(\mu + \frac{1}{|D|(1-\beta)} \sum_{d\in D} R^d \right) \end{aligned} \tag{26}$$

$$R^d \geq \sum_{j\in V_0}\sum_{p\in P_e} x^d_{jp} E^d_{jp} - \mu \ \ \forall d \in D \tag{27}$$

$$R^d \geq 0 \ \ \forall d \in D \tag{28}$$

$$\mu \geq 0. \tag{29}$$

Model 1 captures the unmet demand through only the expected value of unmet demand. In comparison, Model 2 captures the unmet demand in terms of the distribution of the unmet demand x^d_{jp}. Specifically, it minimizes the expected uncertainty of the unmet demand by taking the possible worst outcome of the unmet demand distribution into account.

5.3. Information and refinement questions presented to the HU

At each iteration k, the framework presents the recommended supply chain design, which for our demonstration problem is the set of selected suppliers (i.e., values of y_i), and such a design's expected performance estimates to the HU. This includes the optimal objective function value calculated using (3) or (26). In addition, we also present to the HU, the unmet demand (UD) calculated using 30.

$$\mathrm{UD} = \sum_{d \in D} \mathbb{P}^d \sum_{j \in V_0} \sum_{p \in P_e} x^d_{jp} E^d_{jp} \tag{30}$$

The 95% confidence interval(CI) width of UD is calculated as

$$CI\ width = 2\frac{1.96}{|D|}\sqrt{\sum_{d \in D} \mathbb{P}^d (\sum_{d \in D} \mathbb{P}^d \sum_{j \in V_0} \sum_{p \in P_e} x^d_{jp} E^d_{jp} - \sum_{j \in V_0} \sum_{p \in P_e} x^d_{jp} E^d_{jp})^2}. \tag{31}$$

The local objective function is calculated using (3), if Model 1 is employed. If *model 2* is employed, the objective defined in (26) is employed.

The cost of production is defined as

$$\sum_{d \in D} \sum_{i \in S} f_i y_i + \sum_{d \in D} \mathbb{P}^d \sum_{i \in S} \sum_{p \in P} \sum_{j \in V} c^d_{ipj} w^d_{ipj}. \tag{32}$$

After each iteration k, the framework first asks “What do you not like about the recommended solution or its performance estimates?” and then presents to the HU a set of options that the HU can choose from. These options are 1) the solution performance is deemed unrealistic (i.e. the solution does not contain a confidence interval value) 2) the percentage unmet demand is too high, 3) the confidence interval width of unmet demand is too wide 4) there is unrealistic production cost, 5) satisfied with the solution and its performance. Based on which option is selected, then a follow-up refinement question is asked of the HU. Specifically, if they select option 1 then “What is the percentage variation in demand?” is asked; if option 2 is selected “Do you have additional suppliers that can provide a targeted product?” is asked; if option 3 is selected “What is the required level of confidence in the solution’s unmet demand that is deemed necessary?”; and if option 4 is selected “Do you have additional data on cost?” is asked. In response, the human provides information that the algorithm interprets which is described in more detail in Section 6. Option 5 results in the termination of asking questions. In our demonstration, we assume that we have the required information to accurately capture the lateness (T^{λ}_c, required lead time at customer nodes and τ_{ipj}, time to ship). Therefore, for this demonstration, the refinement questions do not inquire about the lateness related input data.

6. Demonstration of framework via computational experiments

In this section, we demonstrate our refinement procedures, i.e., what refinement questions are asked, how the feedback from the HU is interpreted, and how the update is carried out. Furthermore, in this section, we design a set of computational experiments to quantify the benefits of the key features of our approach, which includes the ability of our method (1) to ask multiple, varied questions to elicit a broader range of information from a HU, (2) to ask targeted questions informed by mathematical properties of the current model, and (3) to make structural modifications to the optimization model in response to human feedback about confidence interval widths.

We conduct these experiments using the supplier selection problem described in Section 5. The details of these experiments and our data generation process are given in Appendix A. The iterative framework is implemented using Python, and the optimization models are solved using GUROBI solver. All computations are conducted on a single computer with an Intel core i7 3.8GHz with 16GB RAM.

6.1. Benchmark performance of the iterative method

We create a ground truth model that represents the model that would result if the HU were asked an unlimited number of questions. The ground truth model used for all of our computational experiments consists of *model 2* defined in Section 5.2.2 and uses *ground truth data* described in Appendix B.1, Appendix B.2, Appendix B.3, Appendix B.4, and Appendix C.1. We solve the ground truth model to optimality, which results in the objective function value denoted as $GTO(\mathbf{y}^*)$, where $\mathbf{y}^*$ are the optimal first stage decision variables, i.e., the optimal set of suppliers selected. In all computational experiments that follow, we solve different optimization models with different data inputs at each iteration. We denote the optimal first-stage solutions (set of suppliers selected) obtained in iteration k as $\mathbf{y}^k$. We are interested in understanding how close such model outputs are to the ground truth solution. To do so, we evaluate the first-stage solutions $\mathbf{y}^k$ obtained in iteration k in the *ground truth model*. That is, we fix the values for $\mathbf{y}^k$ and solve the ground truth optimization model for the remaining decision variable values. This allows us to obtain the evaluation of the performance of the first-stage solution at iteration k, $\mathbf{y}^k$ in the ground truth model, and we denote the objective value as $GTO(\mathbf{y}^k)$. A key performance evaluation in the subsequent sections is the objective function value's percentage distance to the ground truth at iteration k, which is defined in (33).

$$\text{Percentage distance to ground truth at iteration } k = \frac{|GTO(\mathbf{y}^*) - GTO(\mathbf{y}^k)|}{GTO(\mathbf{y}^*)} \cdot 100 \tag{33}$$

6.2. *Iterative refinement eliciting a wide range of information from the HU*

In this section, we demonstrate the ability of our methodology to have a HU respond to different types of questions so that the method can extract a wider range of information iteratively. We refer the reader to Appendix A for the overview of the computational experiment.

Table 1 provides detailed information for 18 iterations of our methodology, which includes (i) the iteration, (ii) the percent distance to the ground truth, (iii) the local objective value which is the objective function value solved using the relevant model (refer Section 5.2 and data inputs), (iv) local cost of production ((32)), (v) 95% confidence interval length of UD ((31)), (vi) UD ((30)), and (vii) the refinement to be executed (see Section 5.3 for a mapping). Each of these are obtained from solving the local model in each iteration k. Further, the last column (refinement) in Table 1 provides how the HU reacts to the information shown to the HU at each iteration, by specifically identifying which option (out of the options described in Section 5.3) they found were unsatisfactory. Then, based on this response, the follow-up question, and the HU feedback to that question, as well as the interpretation action are presented (and we describe these actions in more details below).

As described in Method 2 in the Appendix A, in iteration $k = 0$, we solve the initial problem formulation with our initial data and obtain a solution. Inspecting the information provided to the HU (described in Section 5.3), the HU responds saying the solution performance is deemed unrealistic. They do not like that the unmet demand confidence interval is not defined which implies that the unmet demand is defined as as a single-point estimate in the problem formulation. Demand is associated with demand uncertainty, and therefore, by representing demand using a single-point estimate, the model estimates unrealistic performance. Therefore, the methodology translates the human feedback that the input data for demand uncertainty should be better represented by not a single-point estimate but should accommodate variations in demand. Thus, the HU is asked to provide the expected variation in demand, which we denote as ΔE. For this example, the human responds back that this is specified as 10% current demand E_{jp}. Then the framework interprets

Table 1: Detailed iteration-by-iteration demonstration on the supplier selection problem

Iteration k	**% distance to ground truth**	**Local objective**	**Local cost of production ($)** 000'	**Local unmet demand (UD)** (units)	**Local CI width of UD** (units)	**Refinement Option Type (see Section 5.3)**
0	95.9	1.97	90.7	763.8	N/A	1-input demand variation
1	95.8	1.62	92.7	626.0	63.8	2-add new suppliers
2	73.91	1.42	527.0	436.8	62.6	3-increase β
3	73.91	1.95	527.0	436.8	62.6	3-increase β
4	73.91	2.55	527.0	436.8	62.6	2-add new suppliers
5	45.1	2.13	1254.8	250.9	61.8	4-update cost data
6	45.1	2.36	1622.7	250.9	61.8	4-update cost data
7	45.1	2.59	1997.8	250.9	61.8	4-update cost data
8	45.1	2.62	2051.8	250.9	61.8	4-update cost data
9	45.1	2.68	2143.8	250.9	61.8	4-update cost data
10	45.1	2.71	2180.4	250.9	61.8	2-add new suppliers
11	61.6	2.28	1493.0	250.9	61.8	2-add new suppliers
12	61.6	2.28	1493.0	250.9	61.8	3-increase β
13	29.7	2.75	2798.2	121.7	48.1	4-update cost data
14	29.7	2.78	2844.4	121.7	48.1	4-update cost data
15	29.7	2.81	2890.3	121.7	48.1	4-update cost data
16	29.7	2.84	2936.3	121.7	48.1	4-update cost data
17	29.7	2.84	2935.3	121.7	48.1	3-increase β
18	0.01	3.12	4175.8	25.1	24.4	5-terminate

this additional feedback and generates scenarios assuming demand E^{d}_{jp} at customer node $j \in V_0$ for product $p \in P_e$ varies for each scenario $d \in D$, for example, by employing the Monte Carlo method and sampling from the uniform distribution $U\left[E_{jp}(1-\Delta E), E_{jp}(1+\Delta E)\right]$ where E_{jp} is the initial demand at customer node $j \in V_0$ for product $p \in P_e$. After incorporating the new input data, the resulting SM-MILP in Model 1 is solved. The information shown at iteration $k = 1$ indicates that the HU can now be presented not with just a single point estimate, but a more realistic confidence interval of the design solution's resulting unmet demand.

In iteration $k = 1$, we illustrate how mathematical information embedded in the mathematical formulation can be utilized to generate targeted questions that can extract information from the HU that is most beneficial to the optimization model. After being presented with the current design solution and its estimated performance, and a set of refinement questions, the human identifies that the current design creates excess average unmet demand. The dual variable associated with a particular constraint in a linear program is employed in generating specific questions to elicit more information from the HU.

The unmet demand can be reduced if there is more capacity in the supply chain network. Therefore, the MILP is converted to an LP (by fixing the supplier selection variables and other integer variables) so that the dual variable of each of the supplier-resource *capacity constraints* (8) can be calculated. As the goal is to minimize the objective, a negative dual variable associated with a particular supplier-resource pair suggests that if a *new* supplier with the same capabilities could be added, there is a potential opportunity for improvement in the objective function value (decrease in objective). However, the addition needs to ensure that the newly added supplier is capable of performing the tasks performed by the supplier who is limiting the performance. It is identified that supplier *S22* has the highest negative dual variable value. This can be interpreted as supplier S22, which is in echelon 2 and makes end product 1, creating a bottleneck. Therefore, a targeted question is generated, "Do you know of an additional echelon 2 supplier that could make part 10?". This methodology is explained in Method 1 in Section 6.3. In response, the HU provides information about a new supplier which is used to expand the set of suppliers to the set where $|S|_{k+1} = |S|_k + 1$ (refer Section 6.3). Note that the objective value decreased from 1.62 to 1.42 and the UD reduced from 626 to 436.

After presenting the solution at iteration $k = 2$ and the 95% confidence interval of the unmet demand, the HU responds that the confidence interval of unmet demand is too wide which implies that the solution is too risky as it has a higher level of uncertainty. Then, the algorithm asks for the value of the minimum required level of confidence (β) that should be satisfied in calculating the unmet demand. Then, in the interpretation step, the algorithm changes the model of interest to model 2 and sets the value β of equation (26). After conducting the refinement, we observed that results in iteration $k = 3$) the confidence interval width of unmet demand remained unchanged. This is because the network does not have sufficient amount of potential suppliers in the network which is also shown in iteration $k = 3$. Notably, the objective increased from 1.46 to 1.97 because the CVaR is included in the objective. We note here that the refinement executed in each refinement is not individually discussed to avoid repetition.

In iteration $k = 5$, the HU thinks that the representation of the cost of production is inaccurate. Therefore, the algorithm asks "Do you have additional information on the cost of product 1?" which is denoted as $c^d_{i,p=1,j}$. In response, the HU provides additional information in the form of $C_p\ p \in P$ which is then converted to $C^d_p\ p \in P\ d \in D$ using the Monte Carlo method to represent the scenario information that is used to solve the SM-MILP (refer line 14 in Method 2). Here, we assume that the human has access to the ground truth information of $C_p\ p \in P$. The HU provides the framework for this information to be used in the optimization. After solving the refinement problem formulation, in iteration $k = 6$, we observe that the local objective has increased.

After presenting the solution at iteration $k = 12$ and the 95% confidence interval of the unmet demand, the HU responds that the confidence interval of unmet demand, which is 61.8, is too wide which implies that the solution is too risky as it has a higher level of uncertainty. Then, the algorithm asks for another value of the minimum required level of confidence (β) that should be satisfied in calculating the unmet demand. In iteration $k = 13$, we observed that the confidence interval width of unmet demand is reduced to 48. Notably, the objective increased from 2.28 to 2.75. Selecting more suppliers increases the capacity of the network and reduces the risk of facing unmet demand. Unlike in iteration $k = 2$ to $k = 3$, the network has sufficient amount of potential suppliers for selection which reduces the CI. In iteration $k = 18$, the HU is satisfied with the solution and its performance. Therefore, the iterative refinement is terminated.

Figure 2 displays the evolution of the local objective value and the percentage distance to ground truth as a function of the human iterations. As refinement is carried out, the local objective value increases which reaches the objective value of the ground truth. The solution generated after 18 refinement actions approaches the ground truth solution. This illustrates how the framework, which is able to ask the HU for a wide range of information, is able to generate an optimization model representing the HU's ground truth model.

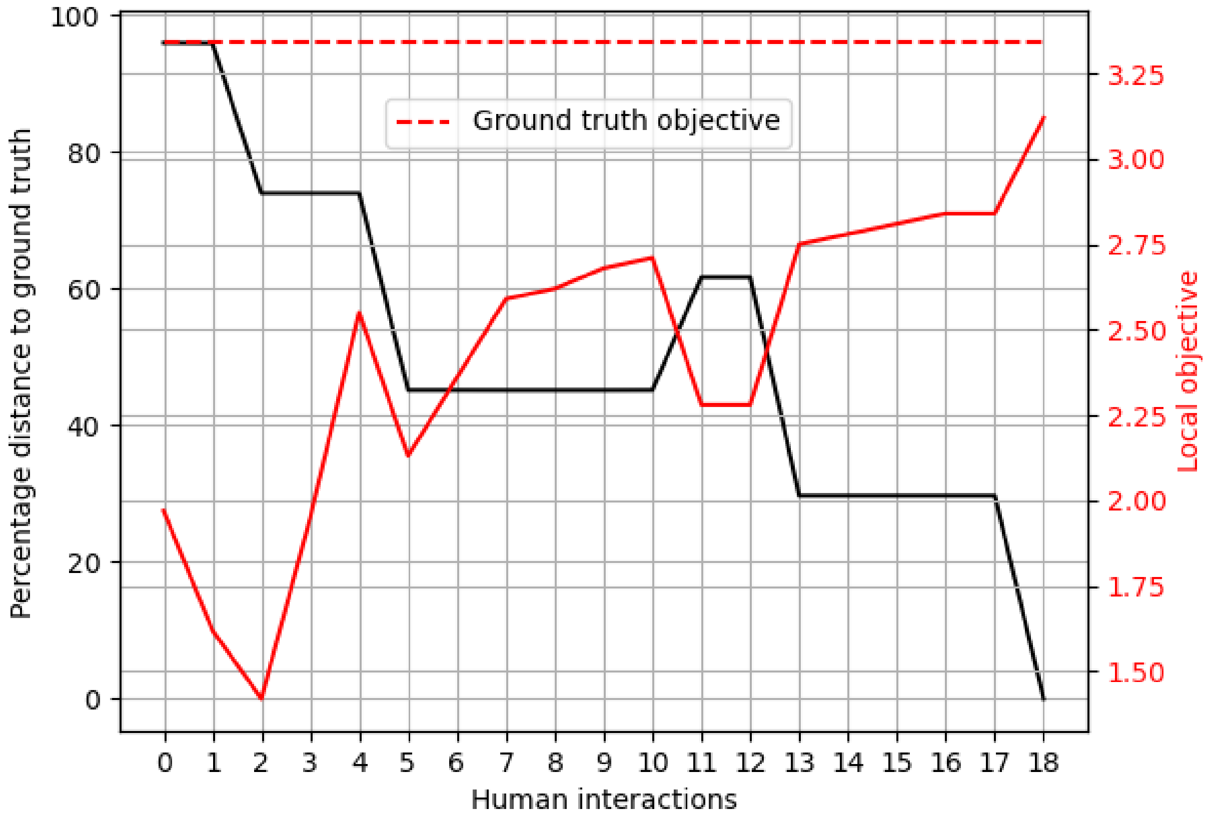

Figure 2: Evolution of local objective function value and percentage distance to ground truth with human interactions

6.3. Refinement by math-informed targeted questions

In this section, we design a computational experiment to quantify the impact of our methodology's ability to query the human for targeted information informed by the mathematical properties of the optimization model. The overview of the computational experiment is described in Appendix A.

Computational experiment: We design this experiment to consider a refinement, where the HU is queried iteratively for new potential suppliers. We consider two settings. One where the HU is asked targeted questions that ask not just for any supplier, but asks for a supplier with a well-defined capability. As described in Method 1, first we convert the SM-MILP to SM-LP by substituting the integer solution $\mathbf{y}^k$ (set of selected suppliers) and solving it for the dual variables. Then, employing the minimal dual variable and corresponding supplier, the targeted question is created. In particular, the targeted questions specify the capabilities that the new suppliers need to reduce the unmet demand.

Method 1 Generating targeted questions at iteration k

```
Require SM-MILP, $\mathbf{y}^k$
1: Substitute $\mathbf{y}^k$ to SM-MILP and convert it to SM-LP as in Section 4.2
2: Solve SM-LP
3: Calculate the dual of constraint (8)
4: Find minimum dual
5: Find the corresponding supplier $y_j$ $j \in S$ to that minimum dual
6: Find corresponding product $p_j \in P$ that supplier $y_j$ supplies using the set of commodities $I_{jpv}$
   for $j \in S$ for any $v \in V$
7: Create question “Do you have suppliers that can provide product $p_j$?”
```

Alternative method: To compare the performance of asking targeted questions, we employ an alternative method that instead of asking targeted questions, a generic question, “Do you have an additional supplier?” is asked. In response, the HU provides input data of a new supplier which is chosen at random. Note that compared to the case of targeted questions, the question does not direct the HU to provide information on a supplier with a specific capability.

We employ model 2 defined in Section 5.2.1 in conducting this experiment. Note that the experiment starts with an initial set of 20 suppliers as defined in Appendix B.5 and with 50 scenarios ($|D| = 50$). Then, depending on the unmet demand in each iteration, the HU is asked to add additional potential suppliers of which the capabilities are specified in the targeted question presented to the HU. As the refinement actions occur, the set and size of potential suppliers grows $|S|_k = |S|_k + 1$. The newly added supplier’s data is extracted from the *ground truth data* as described in Appendix A. We run 10 replications using different sets of input data and the plots that follow display the average over these 10 replications at each iteration k. We define the key performance indicators for this experiment as: 1) percentage distance to the ground truth, and 2) computational time.

When additional potential suppliers are identified by the HU (either through a targeted question or through the alternative approach), Figure 3 displays the percentage difference to the ground truth as a function of human interactions. Initially the solution lies away from the ground truth. However, as the refinements are executed, and more information is gathered from the HU, the percentage to the ground truth decreases with both approaches. Yet, the method of asking targeted questions results in a sharper decrease.

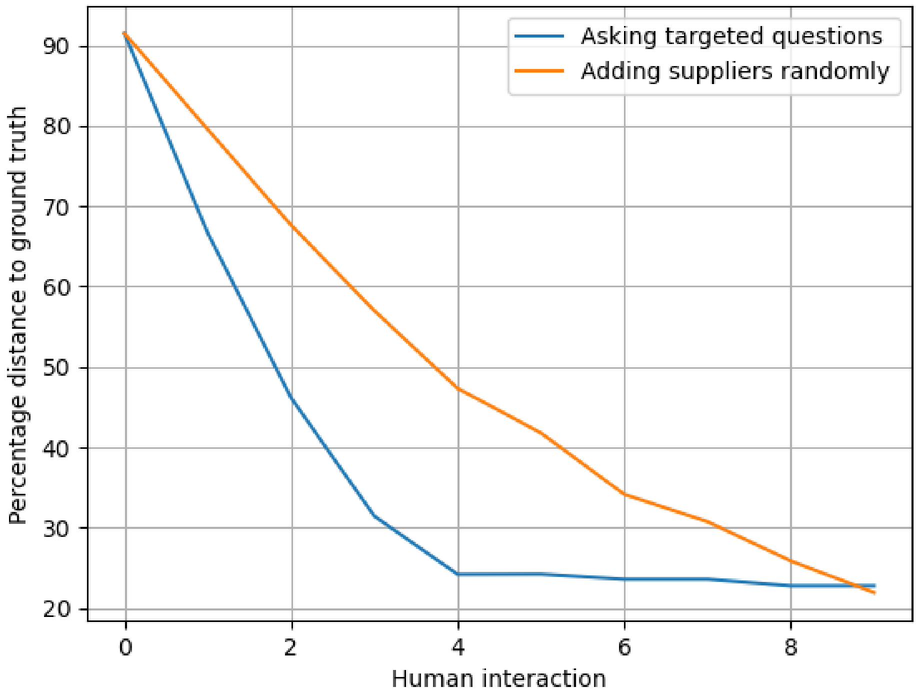

Figure 3: The percentage distance to ground truth with new potential suppliers being added to the network using the math-informed targeted question approach versus an alternative approach that asks for a generic supplier as a function of human iterations.

Furthermore, the average computational time to execute one iteration employing targeted questions is 22s whereas the average computational time to execute one iteration of the alternative method is 14s. We observe a 57% increase in computational time when the targeted questions are employed. The LP relaxation (as shown in Figure 1) step needs to be performed to ask targeted questions which explains the reason for this increase in computational time.

6.4. Demonstrating the ability to reduce the confidence interval widths in response to a HU's risk aversion

Due to uncertainty in the input data, how the solutions generated by SM-MILP will perform is uncertain. Generally, the level of uncertainty in the solution is shown through the confidence interval width of a particular variable of interest. As described in Section 4.7.2, in response to a HU's feedback that the confidence interval width of a particular sub-objective is too wide, our framework implements a risk aversion method in the form of CVaR. This approach requires the underlying mathematical model to structurally be adjusted to Model 2 in Section 5.2.2.

In this experiment, we demonstrate how our framework can reduce the uncertainty of the solution's performance by adjusting the formulation in response to the HU's feedback to optimize not only for expected performance but also for conditional value at risk. If the HU thinks that the confidence interval width of unmet demand is too wide, then the algorithm asks the HU to specify the required level of confidence β, in response. In each iteration, the HU increases the worst threshold of the outcome (level of required confidence) iteratively, which forces the optimization model to choose a more reliable solution. The overview of the computational experiment is described in Appendix A where the maximum number of replications is set to 10 and the maximum number of refinements is set to 7 (where β is changed from 0 to 0.99). In this experiment, we assume that we have perfect knowledge of all the suppliers in the network at the initialization, which defines the magnitudes of the sets $|S_1|_{k=0}$, $|S_2|_{k=0}$. At iteration $k = 0$, the optimization model is initialized using Model 1, then at iteration $k = 1$, it is changed to Model 2. The key performance indicators for this experiment are three-fold; 1) confidence interval width of unmet demand (UD), 2) local objective function and 3) percentage distance to the ground truth.

Figure 4 depicts the change in the confidence interval width of UD as the required level of confidence (β given in equation 26) varies from 0% confidence to 99%. When the conditional value at risk is introduced, there is a decrease in the confidence interval width. As the level of confidence increases the confidence width continues to decrease. Figure 5 depicts the evolution of the local objective as the worst threshold of the outcome (β given in equation 26) is changed from 0% confidence to 99%. When the conditional value at risk is introduced, there is an increase in the Model 1's objective value (which is interested in minimizing expected performance). This is expected, as the level of confidence increases, the number of selected suppliers increases, which results in a solution with higher expected costs.

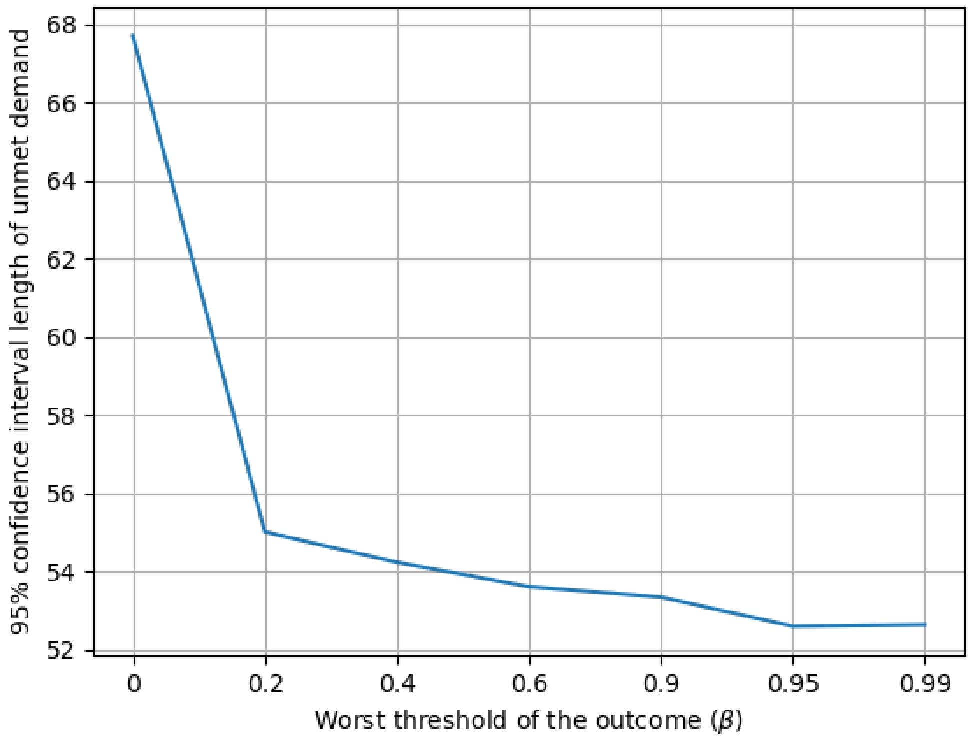

Figure 4: Evolution of confidence interval width of the unmet demand with different levels of confidences β

Thus, implementing a method that optimizes for the conditional value at risk enables the reduction of the confidence interval of the UD. Overall, the computational experiments in this section demonstrate the proposed refinement framework is capable of extracting a broader range of human knowledge to drive solution out of an SM-MILP closer to the ground truth.

7. Conclusion and future works

In this paper, we propose an interactive optimization framework to elicit a broader range of information from a HU to increase the modeling fidelity of a stochastic multi-objective mixed integer linear programming (SM-MILP) model. The IO framework is designed with a goal for the SM-MILP to produce solutions satisfactory to the human designer, measured based on whether the human designer views the recommended solution to be realistic, the expected performance of the solution is adequate across multiple evaluation criteria, and the confidence in the solution's

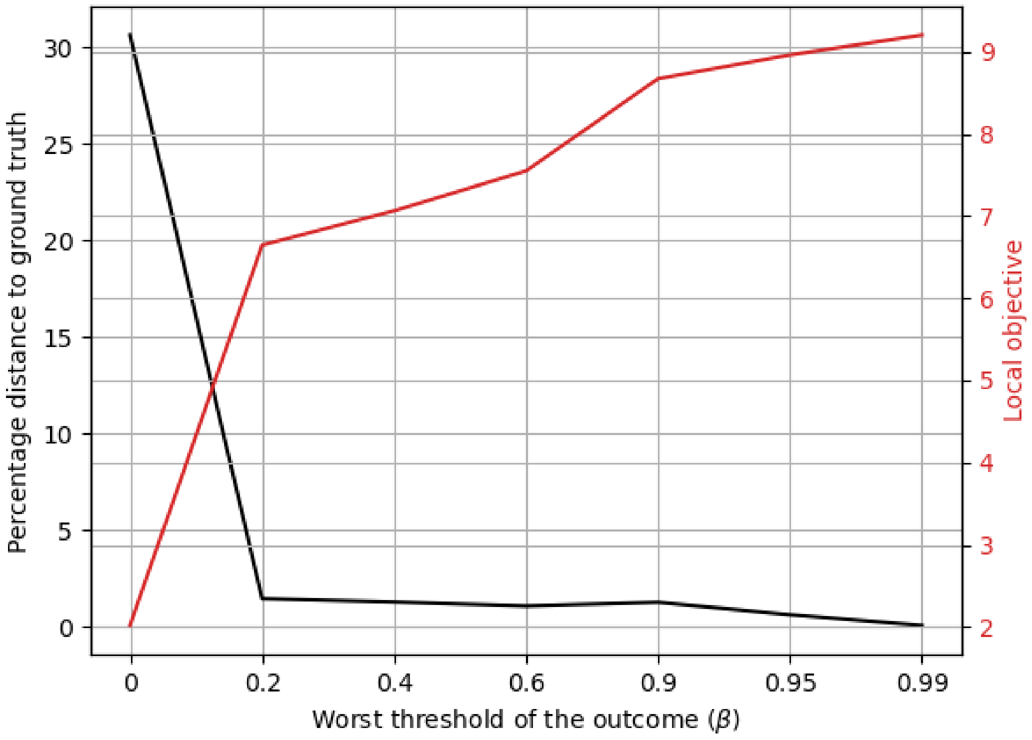

Figure 5: Evolution of confidence interval width of the unmet demand with different levels of confidences β

performance across multiple criteria is high enough for the recommended design to be pursued. We demonstrate through a supplier selection problem that our approach can lead to higher-fidelity models by allowing the HU to provide feedback based on multiple question types. The results obtained through dual-assisted targeted question generation highlight that exploiting dual variable information in an interactive methodology can lead to quicker convergence to a satisfactory solution. Further, we demonstrate that through the use of the CVaR formulation, updated recommended solutions can be generated that provide narrower confidence intervals when the HU deems the current solution too uncertain.

However, one of the limitations of the framework is that the current refinement questions are specialized to the specific optimization model of interests. Thus, future research is needed to further generalize the framework. Further, the proposed methodology requires the HU to select from a set of questions at each iteration. Future research can consider a new mechanism that identifies which question to ask the HU based on the solution of the previous iteration.

8. Acknowledgments

This research was supported by the Office of Naval Research under grant number N00014-22-1-2542.

9. Declaration of interest

The authors report there are no competing interests to declare.

Appendix A. Overview of the computational experiment

We demonstrate how different types of refinements perform under different conditions. We simulate different conditions by generating different replications. The process that we follow in the computational experiments is described as follows in Method 2.

Initially, we define the sets that define the essential requirements of the networks such as the number of suppliers, commodities, customers, etc and replication-independent data. Then, for each replication, we generate the *ground truth data* which includes the replication-dependent data, arcs of the network, and data specific to the ground truth model. Then, we generate *local data* for the replication which includes the replication-independent and dependent data, and local set initialization. In each refinement step k, we solve the relevant SM-MILP model which provides a local solution. The obtained information is recorded and presented to the HU. Then, the refinement is carried out which results in an update in data and/ or model. Note that the data that is utilized to update the *local data* is extracted from the *ground truth data* as described in line 14 in Method 2. In this way, we create a relationship between the ground truth data and local data. This iterative process is executed until we reach *max refinements* and *max replications*.

For the experiment in Section 6.2, we set the maximum replications *max replications* to 1 and the maximum refinement steps *max refinements* to 18. For the experiment in Section 6.3, the maximum number of replications is set to 10 and the maximum number of refinements is set to 9.

Appendix B. Data generation

Appendix B.1. Set generation

To conduct computational experiments we generate the sets as given below.

Method 2 Overview of the computational experiment

```
1: Generate sets as given in Appendix B.1
2: Generate replication-independent data as given in Appendix B.2
3: Set $replication \leftarrow 0$
4: while $replication < max\ replications$ do
5:     Generate ground truth data (denoted as $GTD$) by setting the sets defined in line 1 and data in 2 and replication-dependent data as in Appendix B.3, the arcs of the network (replication-dependent) as in Appendix C.1 and data specific to ground truth as in Appendix B.4
6:     Solve model 2 in Section 5.2.2 using GTD
7:     Generate local data (denoted as $LD(i,d)$ where $i \in S^k$ $d \in D$) by setting the initialization as in Appendix B.5 and extracting relevant replication-independent and dependent supplier node data generated in lines 1,2,5
8:     Set $k \leftarrow 0$
9:     Solve SM-MILP with local data $LD(i,d)$ $i \in S^k$ $d \in D^k$ to get $\mathbf{y}^k$
10:    while $k < max\ refinements$ do
11:        Record (local) information as in Section 5.3 to present to HU
12:        Query the HU
13:        Do refinement
14:        Update model using model1/ 2 in Section 5.2 and/ or local data $LD(i,d)$ $i \in S^k$ $d \in D^k \leftarrow GroundTruthData(GTD)$
15:        Solve SM-MILP with local data $LD(i,d)$ $i \in S^k$ $d \in D^k$ to get $\mathbf{y}^k$
16:        $k \leftarrow k+1$
17:    end while
18:    $replication \leftarrow replication + 1$
19: end while
```

Number of suppliers in the ground truth model - echelon 1: $|S_1| = 20$, echelon 2: $|S_2| = 20$. Total number of customers: $|V_0| = 6$, total suppliers and customers: $|V| = |S| + |V_0|$, total resources: $|RS| = 2$, total commodities: $|P| = 12$, raw commodities: $|P_r| = 10$, end commodities: $|P_e| = 2$, number of scenarios $|D| = 50$.

Appendix B.2. Generation of replication-independent

We generate the fixed cost of suppliers as follows: $f_i = 2000\ for\ i = 1:5 \in S_2$, $f_i = 300000\ for\ i = 5:10 \in S_2$, $f_i = 2000\ for\ i = 11:20 \in S_2$, $f_i = 300000\ for\ i = 1:20 \in S_1$.

We generate θ_{lp} which is the ratio between the commodities assembled as,
$\theta_{lp} = 4,2,1,3,1,1,1,1,1,1,1,1$ for $p = 1.\ldots.10, p \in P_r$ and $l = 11, l \in P_e$.
$\theta_{lp} = 4,1,2,2,1,1,1,1,1,1,1,1$ for $p = 1.\ldots.10, p \in P$ and $l = 12, l \in P_e$, else, $\theta_{lp} = 0$.
Resource consumption: $\alpha^r_{ip} = 0.3\ \forall i \in S, p \in P, r \in RS$
Capacity of suppliers: $u^r_i = 620\ \forall i \in S_1, \forall r \in RS$ and $u^r_i = 700\ \forall i \in S_2, \forall r \in RS$
The remaining parameters are generated as follows: $M_0 = 1000$, $M_\lambda = 100$, $M_c = 500$, $M_{lead} = 100$, $\overline{M} = 1000$, $\Omega_m = 1/|M|\ \forall\ m\ \in M$ and $\Delta_m = [2.5 \times 10^{-6}, 0.1, 0.02, 0.1]$

Appendix B.3. Generation of replication-dependent data

We generate replication-dependent input data for demand based on the following distribution as: $E^d_{jp}\ \sim U[5,120] \forall j \in V_0\ \forall p \in P_e \forall d \in D$.
The time to ship commodities between suppliers and/or customers are given by: $\tau_{ipj}\ \sim U[2,4]\ \forall i \in S, j \in V, p \in P$, $T^{lead}_c \sim U[4,6]\ \forall c \in V_0$.
Returned commodities are defined by: $q_{jp} \sim U[q_{LB}, 2q_{LB}]\ \forall j \in V_0$ and $p \in P$ where $q_{LB} = \sum_{d \in D} \dfrac{P^d E^d_{jp}}{|D|}$.

Appendix B.4. Ground truth specific data

Transportation cost: $c^d_{ipj} = D_{i,j} C^d_p$ where $D_{i,j} \sim U[0.5,1]\ \forall i \in S, j \in V_0$ is the distance matrix and $C^d_p \sim U[50,200]\ \forall p \in P, \forall d \in D$ is the transportation cost per unit per unit distance. The threshold of the worst outcome (β) is set to 0.99.

Appendix B.5. Local model initialization at iteration $k = 0$

Number of potential suppliers: $|S_1|_{k=0} = 10$, $|S_2|_{k=0} = 10$ as $S_1 = 1,\ldots,10$ and $S_2 = 1,\ldots,10$.
Number of scenarios: $|D|_{k=0} = 1$. Transportation cost: $c^d_{ipj} = D_{i,j} C^d_p$ where $D_{i,j} \sim U[0.5,1]\ \forall i \in S, j \in V_0$ is the distance matrix and $C^d_p \sim U[50,55]\ \forall p \in P, \forall d \in D$ is the transportation cost per unit per unit distance.

Appendix C. Network structure

This appendix contains the network structure for the computational experiments described in Section 6. In order to carry out the computational study, we employ a 2 two-echelon supply chain network as shown in Figure C.6 which is a modified version of the 2-tier networks based on empirical data from (43). There are 20 Echelon-2 potential suppliers (n_1) that can provide the commodities $1, \ldots, 10$. The commodities provided by Echelon-2 suppliers are assembled by Echelon-1 suppliers (consisting of $n_2 = 20$ suppliers) that output the final commodities 11 and 12. Then those final commodities are shipped to customer nodes ($n_3 = 6$). The arcs which establish the connections between each supplier is defined as I_{ipj} for $i \in S$, $p \in P$, $j \in V$, which is given below.

Appendix C.1. Set of arcs in the network

We generate the arcs, I_{ipj}, between echelon 2 and 1 such that all arcs are equally likely and at least half of the echelon 2 suppliers are connected to all the suppliers in echelon 1. Furthermore, we generate the arcs, I_{ipj}, between echelon 1 and customers such that all arcs are equally likely and all the customers receive commodities from echelon 1 suppliers.

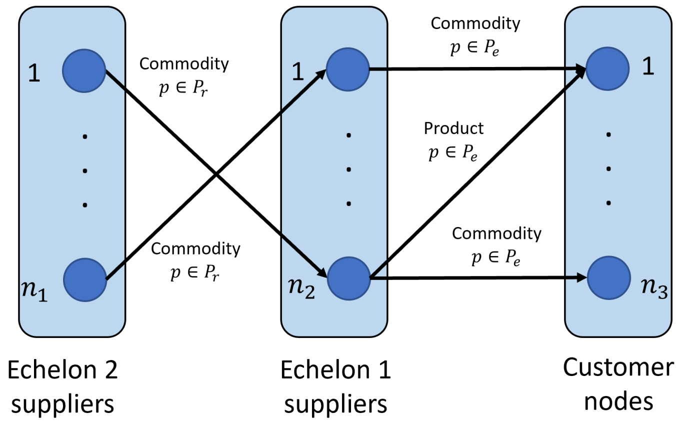

Figure C.6: Two-echelon supply chain network

References

[1] N. Ntube, H. Li, Stochastic multi-objective optimal sizing of battery energy storage system for a residential home, Journal of Energy Storage 59 (2023) 106403. pages 2

[2] A. Jamali, A. Ranjbar, J. Heydari, S. Nayeri, A multi-objective stochastic programming model to configure a sustainable humanitarian logistics considering deprivation cost and patient severity, Annals of Operations Research (2022) 1–36. pages 2

[3] H. B. Yamchi, A. Safari, J. M. Guerrero, A multi-objective mixed integer linear programming model for integrated electricity-gas network expansion planning considering the impact of photovoltaic generation, Energy 222 (2021) 119933. pages 2

[4] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, N. Gaud, A review and taxonomy of interactive optimization methods in operations research, ACM Transactions on Interactive Intelligent Systems (TiiS) 5 (3) (2015) 1–43. pages 2, 3, 4

[5] M. Zhang, J. Pazour, S. Mishra, J. E. Mitchell, D. Jayarathne, Classification of human enrichment and refinement in interactive optimization, in: Proceedings of the IISE Annual Conference Expo 2023, IISE, 2023. pages 2, 3, 4

[6] K. Deb, J. Sundar, Reference point based multi-objective optimization using evolutionary algorithms, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 635–642. pages 4

[7] Q. Ye, F. Li, Y. Hu, et al., Interactive portfolio optimization with cognition-limited human decision making assisted by auxiliary factors (2022). pages 4

[8] A. Jaszkiewicz, R. Słowiński, The 'light beam search'approach–an overview of methodology applications, European Journal of Operational Research 113 (2) (1999) 300–314. pages 4

[9] K. Miettinen, M. M. Mäkelä, Interactive multiobjective optimization system www-nimbus on the internet, Computers & Operations Research 27 (7-8) (2000) 709–723. pages 4

[10] S. Greco, B. Matarazzo, R. Słowiński, Dominance-based rough set approach to interactive multiobjective optimization, in: Multiobjective optimization: Interactive and evolutionary approaches, Springer, 2008, pp. 121–155. pages 4

[11] K. Miettinen, P. Eskelinen, F. Ruiz, M. Luque, Nautilus method: An interactive technique in multiobjective optimization based on the nadir point, European Journal of Operational Research 206 (2) (2010) 426–434. pages 4

[12] T. Laukkanen, T.-M. Tveit, V. Ojalehto, K. Miettinen, C.-J. Fogelholm, Bilevel heat exchanger network synthesis with an interactive multi-objective optimization method, Applied Thermal Engineering 48 (2012) 301–316. pages 4

[13] K. Miettinen, Using interactive multiobjective optimization in continuous casting of steel, Materials and Manufacturing Processes 22 (5) (2007) 585–593. pages 4

[14] S. Hu, D. Li, J. Jia, Y. Liu, A self-learning based preference model for portfolio optimization, Mathematics 9 (20) (2021) 2621. pages 4

[15] T. Jatschka, G. R. Raidl, T. Rodemann, A general cooperative optimization approach for distributing service points in mobility applications, Algorithms 14 (8) (2021) 232. pages 4

[16] Y. Hanine, Y. Lamrani Alaoui, M. Tkiouat, Y. Lahrichi, Socially responsible portfolio selection: an interactive intuitionistic fuzzy approach, Mathematics 9 (23) (2021) 3023. pages 4

[17] T. Jatschka, T. Rodemann, G. R. Raidl, A cooperative optimization approach for distributing service points in mobility applications, in: Evolutionary Computation in Combinatorial Optimization: 19th European Conference, EvoCOP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24–26, 2019, Proceedings 19, Springer, 2019, pp. 1–16. pages 4

[18] T. Jatschka, T. Rodemann, G. R. Raidl, Exploiting similar behavior of users in a cooperative optimization approach for distributing service points in mobility applications, in: Machine Learning, Optimization, and Data Science: 5th International Conference, LOD 2019, Siena, Italy, September 10–13, 2019, Proceedings 5, Springer, 2019, pp. 738–750. pages 4

[19] T. Jatschka, T. Rodemann, G. R. Raidl, A large neighborhood search for a cooperative optimization approach to distribute service points in mobility applications, in: International Conference on Metaheuristics and Nature Inspired Computing, Springer, 2021, pp. 3–17. pages 4

[20] A. van Vliet, C. G. E. Boender, A. H. Rinnooy Kan, Interactive optimization of bulk sugar deliveries, Interfaces 22 (3) (1992) 4–14. pages 4

[21] N. Ahani, T. Andersson, A. Martinello, A. Teytelboym, A. C. Trapp, Placement optimization in refugee resettlement, Operations Research 69 (5) (2021) 1468–1486. pages 4

[22] K. Miettinen, M. M. Mäkelä, Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation, Journal of the Operational Research Society 50 (9) (1999) 949–959. pages 4

[23] M. Tabatabaei, M. Hartikainen, K. Sindhya, J. Hakanen, K. Miettinen, An interactive surrogate-based method for computationally expensive multiobjective optimisation, Journal of the Operational Research Society 70 (6) (2019) 898–914. pages 4

[24] B. Afsar, J. Silvennoinen, G. Misitano, F. Ruiz, A. B. Ruiz, K. Miettinen, Designing empirical experiments to compare interactive multiobjective optimization methods, Journal of the Operational Research Society 74 (11) (2023) 2327–2338. pages 4

[25] G. Rivera, L. Cruz-Reyes, E. Fernandez, C. Gomez-Santillan, N. Rangel-Valdez, An interactive aco enriched with an eclectic multi-criteria ordinal classifier to address many-objective optimisation problems, Expert Systems with Applications 232 (2023) 120813. pages 4

[26] D. Trachanatzi, M. Rigakis, M. Marinaki, Y. Marinakis, An interactive preference-guided firefly algorithm for personalized tourist itineraries, Expert Systems with Applications 159 (2020) 113563. pages 4

[27] A. B. Ruiz, F. Ruiz, K. Miettinen, L. Delgado-Antequera, V. Ojalehto, Nautilus navigator: free search interactive multiobjective optimization without trading-off, Journal of Global Optimization 74 (2) (2019) 213–231. pages 4

[28] A. D. Piemonti, M. Babbar-Sebens, S. Mukhopadhyay, A. Kleinberg, Interactive genetic algorithm for user-centered design of distributed conservation practices in a watershed: An examination of user preferences in objective space and user behavior, Water Resources Research 53 (5) (2017) 4303–4326. pages 4

[29] K. Sindhya, V. Ojalehto, J. Savolainen, H. Niemistö, J. Hakanen, K. Miettinen, Coupling dynamic simulation and interactive multiobjective optimization for complex problems: An apros-nimbus case study, Expert systems with applications 41 (5) (2014) 2546–2558. pages 4

[30] D. Meignan, An experimental investigation of reoptimization for shift scheduling, in: Proceedings of the 11th Metaheuristics International Conference, 2015, pp. 1–10. pages 5

[31] J. Bénabès, E. Poirson, F. Bennis, Integrated and interactive method for solving layout optimization problems, Expert Systems with Applications 40 (15) (2013) 5796–5803. pages 5

[32] S. Hamel, J. Gaudreault, C.-G. Quimper, M. Bouchard, P. Marier, Human-machine interaction for real-time linear optimization, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2012, pp. 673–680. pages 5

[33] L. He, M. Gilbert, T. Johnson, T. Pritchard, Conceptual design of am components using layout and geometry optimization, Computers & Mathematics with Applications 78 (7) (2019) 2308–2324. pages 5

[34] M. Vallerio, J. Hufkens, J. Van Impe, F. Logist, An interactive decision-support system for multi-objective optimization of nonlinear dynamic processes with uncertainty, Expert Systems with Applications 42 (21) (2015) 7710–7731. pages 5

[35] P. Nimmegeers, M. Vallerio, D. Telen, J. Van Impe, F. Logist, Interactive multi-objective dynamic optimization of bioreactors under parametric uncertainty, Chemie Ingenieur Technik 91 (3) (2019) 349–362. pages 5

[36] Y. Cai, G. H. Huang, Q. Lin, X. Nie, Q. Tan, An optimization-model-based interactive decision support system for regional energy management systems planning under uncertainty, Expert Systems with Applications 36 (2) (2009) 3470–3482. pages 5

[37] K. Kato, M. Sakawa, H. Katagiri, C. Perkgoz, An interactive fuzzy satisficing method based on fractile criterion optimization for multiobjective stochastic integer programming problems, Expert Systems with Applications 37 (8). pages 5

[38] C. Li, I. E. Grossmann, A review of stochastic programming methods for optimization of process systems under uncertainty, Frontiers in Chemical Engineering 2 (2021) 34. pages 7

[39] N. Metropolis, S. Ulam, The monte carlo method, Journal of the American Statistical Association 44 (247) (1949) 335–341. pages 12

[40] R. T. Rockafellar, S. Uryasev, et al., Optimization of conditional value-at-risk, Journal of risk 2 (2000) 21–42. pages 12, 18

[41] J. Chai, E. W. Ngai, Decision-making techniques in supplier selection: Recent accomplishments and what lies ahead, Expert Systems with Applications 140 (2020) 112903. pages 13

[42] S. Aouadni, I. Aouadni, A. Rebaï, A systematic review on supplier selection and order allocation problems, Journal of Industrial Engineering International 15 (2019) 267–289. pages 13

[43] S. P. Willems, Data set—real-world multiechelon supply chains used for inventory optimization, Manufacturing & service Operations Management 10 (1) (2008) 19–23. pages 34