Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: www.tandfonline.com/journals/tjor20

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

Damsara Jayarathne , Sandipan Mishra , John E. Mitchell & Jennifer Pazour

To cite this article: Damsara Jayarathne , Sandipan Mishra , John E. Mitchell & Jennifer Pazour (02 Jun 2026): An interactive optimisation framework for incorporating a broader range of human feedback into stochastic multi-objective mixed integer linear programs, Journal of the Operational Research Society, DOI: 10.1080/01605682.2026.2677605

To link to this article: https://doi.org/10.1080/01605682.2026.2677605

Published online: 02 Jun 2026.

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY
https://doi.org/10.1080/01605682.2026.2677605

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

Damsara Jayarathne, Sandipan Mishra, John E. Mitchell and Jennifer Pazour

Rensselaer Polytechnic Institute, Troy, NY, USA

ABSTRACT

Interactive optimisation (IO) combines the analytical power of optimisation frameworks with human's contextual expertise. However, prior IO approaches require human users to repeatedly provide the same type of input or directly modify the model to incorporate different information. As a result, IO frameworks elicit a narrow range of human knowledge or require substantial optimisation expertise from users. To address these limitations, an IO framework is proposed that allows human users to respond to multiple types of queries. The framework aims to produce higher-fidelity stochastic multi-objective mixed-integer linear programming models. It employs targeted questions to elicit specific information from users, a Monte Carlo-based framework to transform human responses into input data for a scenario-based optimisation model formulation, and uses a Conditional Value at Risk (CVaR) formulation to balance expected performance with risk tolerances. Computational experiments on a supplier selection problem demonstrate that this framework can narrow the reality gap and converge towards the ground-truth solution. Moreover, it dynamically adapts to user feedback, and when the human expresses insufficient confidence in the solution's performance, it can recommend solutions with narrower performance confidence intervals.

PRACTITIONER SUMMARY

This paper presents a new interactive optimisation (IO) framework designed to improve how complex decision models, particularly stochastic multi-objective mixed-integer linear programs (MILPs), are created. For practitioners, this approach provides a structured way to integrate a broad range of human domain expertise into advanced optimisation models without requiring deep optimisation expertise, leading to more reliable and trusted optimisation-based decisions under uncertainty.

In many real-world design problems, such as supplier selection and supply chain network design, decisions must be made under uncertainty, with multiple objectives (e.g., cost, resilience, service), and with long-term strategic consequences. While stochastic MILP models are powerful tools, they inevitably simplify reality. Important constraints, preferences, or contextual insights may be omitted, creating a "reality gap" between the model and real-world decision needs. Traditional IO methods often rely on repeatedly asking the same type of preference question or require users to directly modify the mathematical model. The proposed framework addresses this by: (1) Asking targeted, maths-informed questions to extract the most valuable user knowledge. (2) Translating human responses into model inputs using a Monte Carlo scenario-based approach. (3) Incorporating Conditional Value at Risk (CVaR) to balance expected performance with risk tolerance. (4) Iteratively refining the model until the solution meets the user's confidence and performance expectations. Computational experiments on a supplier selection problem show that the framework narrows the reality gap, converges towards ground-truth solutions, and can recommend solutions with tighter performance guarantees.

ARTICLE HISTORY
Received 14 March 2025
Accepted 16 May 2026

KEYWORDS
Stochastic mixed integer linear programs; multi-objective; interactive optimisation; human-in-the-loop; CVaR

1. Introduction

Stochastic mixed integer linear programming (MILP) models are optimisation models that recommend the best systematic decisions when there are combinatorially many options, parameter uncertainty, and decision trade-offs due to systemic interactions, and constraints on finite resources. Multi-objective stochastic MILP models consider multiple decision criteria simultaneously, and are commonly used to guide a wide range of design problems. For example, in supply chain design, determining what network of suppliers, as well as the associated supply contract amounts and shipment values, is often aided by multi-objective stochastic MILP models (Ntube & Li, 2023; Jamali et al., 2022; Yamchi et al., 2021). Not only are there combinatorially many different ways to design such a network, but supplier selection decisions are strategic in nature, which means they must

CONTACT Jennifer Pazour pazouj@rpi.edu Industrial and Systems Engineering Department, Rensselaer Polytechnic Institute, 110 8th street, Troy, 12180, NY, USA

be made and locked in with only an uncertain forecast of future supplier performance and costs, as well as future demand. Such decisions also tend to be based on multiple criteria (e.g., minimising costs, maximising resilience, maximising on-time performance).

The construction of Stochastic-MILP models are subjective; they only capture the features and information that they were designed to represent. This means the optimisation 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 integrates a human user (HU) into the optimisation-modelling process (Meignan et al., 2015).

Interactive optimisation (IO) incorporates advanced optimisation 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 optimisation approaches (Meignan et al., 2015). 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.

In this paper, we propose a new IO framework to elicit and incorporate a broader range of information from the HU to arrive at a higher fidelity Stochastic MILP. The IO framework is applicable to optimisation problems that can be formulated as a two-stage stochastic program, and where multiple decision criteria can be captured through a weighted sum approach. The framework provides the optimal solution to the HU, given the currently modelled mathematical structure and input data. The refinement layer takes in the current model-based solution and presents possible questions to the HU. Then, based on the HU's feedback, the model fidelity is improved through incorporating more accurate uncertainty quantification, activating new decision criteria or constraints, or updating input data. This iterative process repeats until a terminating criterion is met.

This IO 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. IO has a solid and long-standing background, where most IO work has focused on multi-objective problems, where methods interactively elicit a HU's preferences to iteratively update the preference weights among multiple decision criteria (Afsar et al., 2023; Rivera et al., 2023; Trachanatzi et al., 2020; Ruiz et al., 2019; Zhang et al., 2023). While we are also interested in interacting with a HU in an iterative way, the goal of this work varies. Instead of updating preference weights, we focus on increasing the model fidelity of a Stochastic-MILP formulation when much more varied information is queried from the HU. To do so in an efficient way, the framework uses information from the mathematical formulation, specifically dual variable values obtained from the LP-relaxation of the Stochastic-MILP, to determine the most valuable information to be extracted from the user (which we term as maths-informed targeted questions). A key challenge with creating a method that elicits different and varied types of information from the HU is how to incorporate the response from an HU and use it to update the maths 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 Stochastic MILP *via* a scenario-based optimisation model formulation. Another challenge is to arrive at a solution the HU finds satisfactory, particularly one that the HU is sufficiently confident would perform well. To address this, we use a Conditional Value at Risk (CVaR) formulation to minimise the risk associated with the solution and the framework can iteratively elicit the HU's preferred trade-off between expected performance and expected risk. We demonstrate the key features of our framework on a supplier selection problem.

2. Related work

IO refers to a specialised area of study that empowers a HU to actively engage in the optimisation process by providing input and guidance in an iterative manner (Meignan et al., 2015). IO has significantly advanced over the past twenty years, with a plethora of research surveyed in the following review articles (Meignan et al., 2015; Zhang et al., 2023). While the interaction between the HU and the optimisation 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 optimisation model is solved, and asks the HU to provide feedback that can improve the accuracy of

the input data, see for example (Deb & Sundar, 2006; Jaszkiewicz & Słowiński, 1999; Miettinen & Mäkelä, 2000; Hu et al., 2021; Jatschka, Raidl, & Rodemann, 2021; Hanine et al., 2021; Jatschka, Rodemann, & Raidl, 2021; van Vliet et al., 1992; Ahani et al., 2021).

Many IO frameworks employ a user interface that allows the HU to inspect the solution manually (van Vliet et al., 1992; Singh et al., 2008). Furthermore, these interfaces, such as NIMBUS and NAUTILUS (Ruiz et al., 2019; Miettinen & Mäkelä, 2000; Cai et al., 2009) allow the user to update input data, which enables efficient transaction of information. In addition, the HU can interactively track the evolution of the solutions and associated variables.

IO frameworks have most commonly been applied to multi-objective optimisation problems. Unlike single-objective optimisation, which focuses on improving a single criterion, multi-objective optimisation seeks to find a solution for problems with multiple, often conflicting objectives simultaneously. Approaches commonly identify the Pareto front, which is a set of solutions where no solution can be improved in one objective without sacrificing performance in another. If the preference weights among criteria can be estimated and if the multiple criteria can either be normalised or are of the same units, then a weighted sum approach can be used, which is guaranteed to find a point on the Pareto front. While this allows for the optimisation of a single function instead of multiple conflicting objectives, the choice of weights can significantly impact the resulting solution. Thus, a well-researched area of IO is to ask HU to provide more information to update the preference weights across multiple criteria or goals (Afsar et al., 2023; Rivera et al., 2023; Trachanatzi et al., 2020; Ruiz et al., 2019; Jatschka et al., 2019a, 2019b; Greco et al., 2008; Miettinen et al., 2010; Laukkanen et al., 2012; Miettinen, 2007; Piemonti et al., 2017; Sindhya et al., 2014). The literature on interactive multi-objective optimisation has developed different strategies to efficiently and accurately elicit from HU such preferences. For example, in Miettinen and Mäkelä (1999), an IO strategy using reference points is presented. In such a strategy, a multi-objective optimisation model is solved, starting with an approximated Pareto optimal set. Then, the HU is asked to classify objective functions, and specify upper and lower bounds, and weighting coefficients. A subproblem is formulated by 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 is employed in (Tabatabaei et al., 2019). In this work, the HU provides different reference points so that they can explore the Pareto frontier through several interactions. Meta-objective optimisation, which optimises a single function that evaluates how well a candidate solution satisfies the original multiple objectives, instead of optimising several objectives separately, has also been explored in IO frameworks (Kato et al., 2010; Deb et al., 2010). A common key element in the existing meta-objective and multi-objective IO approaches is that they employ a static query, where the query presented to the HU does not change from one iteration to the next. Hence, there is a need for IO methods that can change what type of information they elicit and incorporate this broader range of human knowledge into optimisation models.

Stochastic optimisation captures uncertainty in optimisation models, and IO frameworks for stochastic optimisation techniques exist, e.g. Vallerio et al. (2015). One stochastic optimisation approach is robust optimisation, which makes decisions that ensure that solutions remain feasible under identified uncertainty set conditions. Interactive multi-objective robust optimisation frameworks have been proposed to actively involve decision-makers in directing the search for robust solutions under deep uncertainty (Shavazipour et al., 2025). Furthermore, there have been efforts in portfolio optimisation (Hassanzadeh et al., 2014), and reliability improvement (He et al., 2021) that utilise robust IO frameworks. In this work, we employ a Conditional Value at Risk (CVaR) approach (Rockafellar & Uryasev, 2000), which is a stochastic optimisation approach that makes decisions by minimising the extreme loss in the tail of the distribution, thereby making decisions that are more risk-averse than a traditional expected value approach.

2.1. Knowledge gap and contribution

While the IO domain provides useful frameworks for HUs to interactively engage with outputs of optimisation models and provide feedback to improve specific data input estimates, 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 updated preference information for multi-objective optimisation. Generally, HUs have additional context that could be used to increase model fidelity. In many cases, IO frameworks are assisted by an analyst (Pajasmaa et al., 2025) who has optimisation expertise and provides further information to the HU while guiding them through the iterative process. Existing IO frameworks do not support posing multiple types of questions to a HU and interpreting their responses

for integration into an optimisation framework without specialised optimisation knowledge.

Thus, our approach contributes to the IO literature by expanding the information elicited and incorporated from the HU into a Stochastic MILP by 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. Our methodology also allows the HU to be a domain expert, but does not require them nor an additional analyst to be an optimisation expert. Instead of having the HU directly make adjustments to the optimisation formulation, the methodology queries domain information from the HU, and the methodology transforms this information into inputs for the optimisation model.

3. Problem statement and scope

In this work, we focus on complex design problems that can be posed as a stochastic multi-objective mixed integer linear programming optimisation problem. Our framework is applicable when multiple criteria can either be expressed in similar units of measure or a normalisation technique can be applied. We assume the preference weights among criteria can be estimated and used to formulate a Stochastic MILP that combines the multiple criteria into a single objective by using a weighted sum method that assigns weights to each criteria that reflect their relative importance (Ehrgott, 2005). We also focus on problems in which, for all uncertain parameters, their distribution is available to us and these distributions are used to capture uncertainty *via* a scenario-based approach. Hence, our framework is not applicable for situations where uncertainties cannot be modelled using a probability distribution.

As model-based design relies on an abstraction, the Stochastic-MILP problem formulation and the associated data will be different from the characteristics pertaining to the real-world design. Therefore, 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 optimisation 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 confidence in the solution's performance across all evaluation criteria should be high enough for the HU to pursue the recommended design.

Our framework aims 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 a domain expert, but not an expert in optimisation modelling. 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 interact directly with the Stochastic-MILP formulation. This requires a way to first interpret broad feedback and translate it into a format that can be fed into the optimisation 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. Given the well-researched area to elicit preference information from a HU, our framework does not exclusively focus on this, but instead is designed to ask about a broader range of information from a HU to improve model fidelity.

4. Proposed methodology

The proposed IO framework for refining Stochastic-MILP models by posing targeted and different queries to the HU at each iteration is given in Figure 1. We adopt a two-stage stochastic modelling approach (Li & Grossmann, 2021), where stage one decisions are made *now* and stage two decisions are made after uncertainty in the system has been realised. We solve this two-stage stochastic MILP using a scenario-based approach (Shavazipour et al., 2021; Hsieh, 2024; Niknam et al., 2012). Specifically, the uncertain parameter values are represented with a set of scenarios, denoted as $\omega \in \Omega$. In the scenario-based Stochastic-MILP model, decisions are made by optimising a weighted sum of expected performance across the set of design criteria $m \in M$. Our framework solves such a model at each iteration k, and we present a general scenario-based Stochastic MILP in iteration k in (1).

$$
\begin{aligned}
\hat{Z}^k = \min_{y^k,\, x^k_\omega} \; & (c^k)^\top y^k + \sum_{\omega\in\Omega} \mathbb{P}_\omega \sum_{m\in M} \gamma^k_m (F^k_{\omega m})^\top x^k_\omega \\
\text{s.t.} \quad & \mathbf{A}^k y^k \le b^k \\
& \mathbf{T}^k y^k + \mathbf{H}^k_\omega x^k_\omega \le g^k_\omega \qquad \forall \omega \in \Omega \\
& x^k_\omega \in X^k_\omega, \quad X^k_\omega = \{x^k_\omega | x^k_\omega \ge 0, x^k_{i\omega} \in \{0,1\}, \forall i \in I^k_1\} \\
& y^k \in Y^k, \quad Y^k = \{y^k | y^k \ge 0, y^k_j \in \{0,1\}, \forall j \in J^k_1\}
\end{aligned}
\tag{1}
$$

Sets

I set of first-stage variables	$I_1 \subset I$ set of first-stage binary variables
J set of second-stage variables	$J_1 \subset J$ set of second-stage binary variables
M set of objectives	Ω set of scenarios

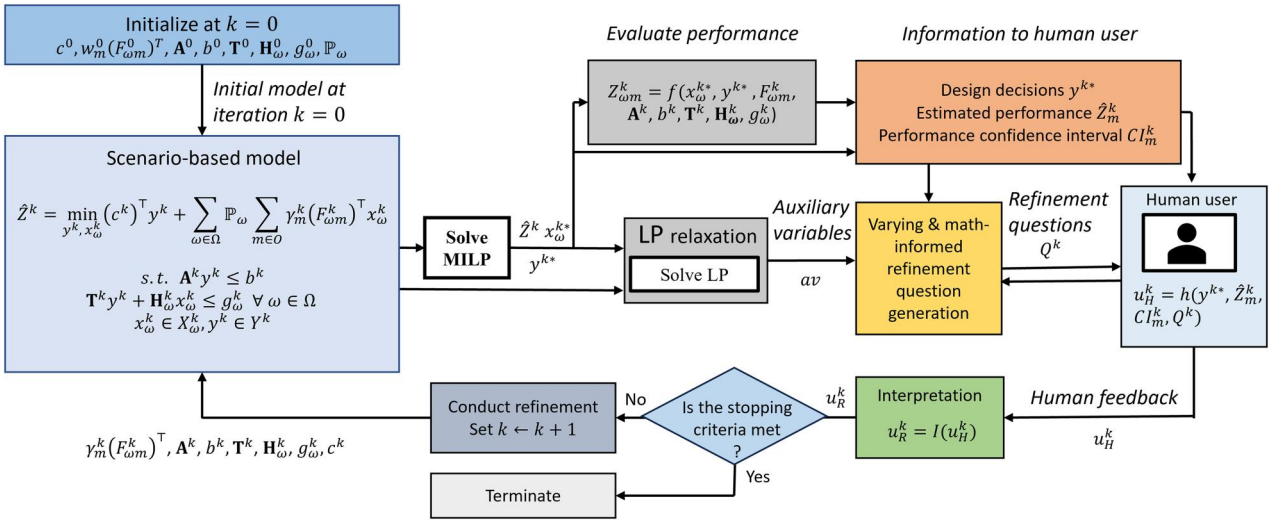

Figure 1. Proposed interactive optimisation framework.

Decision variables

x_ω^k	Vector of second stage decisions for $\omega \in \Omega$ in iteration k
y^k	Vector of first stage decisions in iteration k
$\hat{Z}^k$	Objective function value in iteration k

Input data

$\mathbf{A}^k$, $\mathbf{T}^k$	Coefficient matrices for $y^k \in I$ in constraints not influenced by uncertainty (i.e., not varying by scenarios $\omega \in \Omega$) in iteration k
b^k	Vector containing the right-hand side constants for the first-stage variables in iteration k
c^k	Coefficient vector of $y^k \in I$ in the objective function in iteration k
$F_{\omega m}^k$	Coefficient vector of the second-stage decision variables in scenario $\omega \in \Omega$ for objective $m \in M$ in iteration k
g_ω^k	Vector containing the right-hand side constants for the second-stage variables in scenario $\omega \in \Omega$ in iteration k
$\mathbf{H}_\omega^k$	Coefficient matrix for $x_\omega^k \in J$ in constraints influenced by uncertainty (i.e., varying by scenarios $\omega \in \Omega$) in iteration k
$\mathbb{P}_\omega$	Probability of scenario $\omega \in \Omega$ occurring
γ_m^k	Preference weight assigned for each objective $m \in M$ in iteration k

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 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 $\omega \in \Omega$, x_ω^{k*}.

After solving the scenario-based Stochastic-MILP model, the framework asks the HU to evaluate the recommended design solution, its performance, and its performance confidence intervals across multiple criteria. These values are presented to the HU, who is asked whether or not the solution and its performance are satisfactory. If not, the HU is then presented with a set of targeted refinement questions. The HU selects a question and responds to it. These responses are then interpreted so that they can be meaningfully and tractably incorporated back into the optimisation model. 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. LP relaxation

The 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 $x_{i\omega}^{k*} \in \{0,1\}, \forall i \in I_1^k,\ \forall \omega \in \Omega$ and $y_j^{k*} \in \{0,1\}, \forall j \in J_1^k$ from solving (1) at iteration k are fixed at their optimal values, and then we solve the resulting linear programming formulation, which provides a dual variable value for each constraint in the LP. This is executed in the LP relaxation block.

4.2. Evaluating performance

The scenario-based model solved at iteration k determines a single set of design solutions (i.e., y^{k*} values) that optimises 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 criteria to another. Therefore, at iteration k we evaluate the performance of each scenario $\omega \in \Omega$ for each objective $m \in M$ as $Z_{\omega m}^k$ as

$$Z_{\omega m}^k = (c^k)^\top y^k + \gamma_m^k (F_{m\omega}^k)^\top x_\omega^k. \tag{2}$$

Then, using these $Z_{\omega m}^k$ values, we calculate performance confidence intervals, CI_m^k for each objective $m \in M$ at iteration k.

4.3. 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 terms of each objective criteria Z_m^k, and the performance confidence interval in each objective CI_m^k. This information is displayed to the HU, along with a set of refinement questions.

4.4. Refinement questions

In our framework, the HU is queried using different refinement questions that elicit a wide range of knowledge. At iteration k, the set of refinement questions Q^k is presented to the HU alongside with the current solution and its performance, i.e., y^{k*}, $\hat{Z}_m^k$, and CI_m^k. These refinement questions ask the HU different inquiries adapted to the current state of the model and the outputs at each iteration k. The refinement questions are personalised to the problem at hand and are designed to ask for 1) better-quality information when previously provided inputs were incomplete, or 2) more detailed information or granular data because the previously provided inputs did not possess the appropriate specificity.

Maths-informed refinement questions utilise the information from the mathematical formulation, specifically dual variables (Section 4.1), to ask targeted questions that focus on extracting the most valuable information from the user, which we determine using the dual variable values. This allows us to personalise the questions displayed at each iteration. We provide example questions in Section 5.2 for our demonstration problem.

4.5. Human feedback

The HU responds by selecting a question to answer and answering it. The human feedback denoted as u_H^k, corresponds to the feedback provided by the HU in response to the selected refinement question 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 $\hat{Z}_m^k$, the performance confidence intervals CI_m^k, and the refinement questions Q^k. Thus, we denote it as a function of these inputs by $u_H^k = h(y^{k*}, \hat{Z}_m^k, CI_m^k, Q^k)$. 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 input into the optimisation model, which we explain in the next section.

4.6. 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 modelling fidelity of our scenario-based model. As shown in Figure 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. The refinement actions u_R^k can update input data, so that the model is of higher fidelity. If further improvement is not required, the process is terminated. For all of these refinement actions, we provide illustrative examples in Section 6.2.

4.6.1. 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, γ_m^k, $A^k, b^k, \mathbf{T}^k$, $\mathbf{H}_\omega^k$, or g_ω^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* (Metropolis & Ulam, 1949) 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 $\mathbf{H}_\omega^k$, or g_ω^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.

4.6.2. Designing a solution for risk aversion

We design a refinement mechanism able to produce solutions that have lower risk (i.e., have a smaller estimated confidence interval of a given performance criterion). This allows our methodology to generate a design solution of which the confidence in the solution's performance is high enough that the HU can pursue the recommended design. To achieve this, we employ a CVaR formulation (Rockafellar & Uryasev, 2000), which is a risk

aversion approach that minimises the extreme losses in the tail of a distribution of interest. It does this by quantifying the average loss over worst-case scenarios (whose loss is excessive) beyond a given threshold value.

Interpretation: In response to a too risky solution, the interpretation is to increase the threshold value that defines the worst outcome.

5. Demonstration problem: A supplier selection problem

As supply chain design problems are commonly guided through the use of two-stage Stochastic MILP (Chai & Ngai, 2020; Aouadni et al., 2019), in this section, we introduce a specific multi-echelon capacitated supplier selection problem used to demonstrate our proposed methodology. Our demonstration 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. The HU for this demonstration problem is assumed to be a supply chain domain expert, who is responsible for supplier selection in their organisation, and we use a simulated HU in the demonstration of our framework to this specific problem.

5.1. Scenario-based two-stage stochastic MILP 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 Stochastic 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 the shipment flows, unmet demand, and on-time performance for a given scenario (which occur after the stochastic values in demand and other supply chain resources have materialized).

Sets

C	set of customer nodes	R	set of resources
M	set of objectives	S	set of all suppliers
Ω	set of scenarios	S_1	set of echelon 1 suppliers
P	set of commodities	S_2	set of echelon 2 suppliers
P_e	set of end commodities	V	set of suppliers and customers
P_r	set of raw commodities		

We refer the reader to C for a visual of the structure of the network. Note that echelon 2 suppliers supply raw commodities processed at echelon 1 suppliers to produce end commodities; thus, these sets have the following relationships with each other:

$$S_1 \subset S,\ S_2 \subset S,\ V = S \cup C,\ P = P_r \cup P_e.$$

Input Parameters

α_{spr}	Estimated rate of resource $r \in R$ consumed by supplier $s \in S$ for commodity $p \in P$
β	Threshold defining the worst outcome ($\beta \in [0,1)$)
$d_{spv\omega}$	Cost to make one unit of commodity $p \in P$ and ship from supplier $s \in S$ to node $v \in V$ in scenario $\omega \in \Omega$
Δ_m	Normalizing coefficient of the m^{th} objective criteria, where $m \in M$
$E_{cp\omega}$	Customer demand for end product $p \in P_e$ at customer node $c \in C$ for scenario $\omega \in \Omega$
f_s	Fixed cost of establishing a contract with supplier $s \in S$
γ_m	Weight of m^{th} objective criteria, where $m \in M$
$l_{spv} =$	$\begin{cases} 1, & \text{if commodity } p \in P \text{ is sent from supplier s} \in \text{S to v} \in \text{V} \\ 0, & \text{otherwise} \end{cases}$
$\mathbb{P}_\omega$	Probability of scenario $\omega \in \Omega$ occurring
M^{λ}	Maximum time to process at any supplier or customer
M^{lead}	Maximum lead time at any customer
$\bar{M}$	Maximum allowable amount of any product to ship from any supplier to any node in any scenario
q_{sp}	Expected number of units requiring rework of commodity $p \in P$ at supplier $s \in S$
θ_{pl}	Number of units required of commodity $p \in P$ to make one unit of $l \in P$
τ_{spv}	Time to ship commodity $p \in P$ from supplier $s \in S$ to node $v \in V$
T_c^{Lead}	Required lead time at customer node $c \in C$
u_{sr}	Capacity of supplier $s \in S$ to handle resources $r \in R$

Decision Variables

$h_{spv} =$	$\begin{cases} 1, \text{ if sent commodity } p \in P \text{ from suppliers} \in \text{S to v} \in \text{V in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$
λ_v	Time of the last delivery to node $v \in V$ in some scenario $\omega \in \Omega$ (starting from echelon 1)
μ	Value at risk of unmet demand
$o_{sv} =$	$\begin{cases} 1, \text{if any commodity is sent from supplier } s \in S \text{ to v} \in \text{V in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$
$\phi_{sv} =$	$\begin{cases} T_s, \text{if } o_{sv} > 0 \text{ for s} \in \text{S to v} \in \text{V in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$
$\psi_c =$	$\begin{cases} (\lambda_c - T_c^{Lead}), \text{if } v_c > 0 \text{ for customersc} \in \text{C in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$
$r^{IN}_{psl\omega}$	Total inflow of commodity $p \in P$ at supplier $s \in S_1$ to be converted to $l \in P$ for scenario $\omega \in \Omega$
$r^{OUT}_{sp\omega}$	Total outflow of commodity $p \in P$ at supplier $s \in S_1$ for scenario $\omega \in \Omega$
R_ω	Excess unmet demand for scenario $\omega \in \Omega$
T_v	Maximum time to receive commodities at node $v \in V$ from the previous echelon in some scenario $\omega \in \Omega$
$v_c =$	$\begin{cases} 1, \text{if } \lambda_c - T_c^{Lead} > 0 \text{ for customers c} \in \text{C in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$
$w_{spv\omega}$	Number of units of commodity $p \in P$ shipped from $s \in S$ to $v \in V$ for scenario $\omega \in \Omega$
$x_{cp\omega}$	Percentage of unmet demand at customer $c \in C$ of end product $p \in P_e$ for scenario $\omega \in \Omega$
$y_s =$	$\begin{cases} 1, \text{if contract is established with supplier } s \in S \\ 0, \text{otherwise} \end{cases}$
$z_{spv} =$	$\begin{cases} 1, \text{if commodity } p \in P \text{ is sent from supplier s} \in \text{S to v} \in \text{V in some scenario } \omega \in \Omega \\ 0, \text{otherwise} \end{cases}$

Objective

$$\begin{aligned}\min \quad & \gamma_1\Delta_1\left(\sum_{s\in S} f_s y_s + \sum_{\omega\in\Omega}\mathbb{P}_\omega \sum_{s\in S}\sum_{p\in P}\sum_{v\in V} d_{ipc\omega} w_{spv\omega}\right) \\ & +\gamma_2\Delta_2 \sum_{\omega\in\Omega}\mathbb{P}_\omega \sum_{c\in C}\sum_{p\in P_e} x_{cp\omega}E_{cp\omega} + \gamma_3\Delta_3\sum_{c\in C}\psi_c \\ & +\gamma_4\Delta_4\left(\mu + \frac{1}{|\Omega|(1-\beta)}\sum_{\omega\in\Omega} R_\omega\right)\end{aligned} \tag{3}$$

Constraints

$$\sum_{s\in S} w_{spc\omega} \geq E_{cp\omega}(1 - x_{cp\omega}) \forall c \in C,\ \forall p \in P_e,\ \forall \omega \in \Omega \tag{4}$$

$$\sum_{u\in S_2} w_{ups\omega} = \sum_{l\in P} r^{IN}_{psl\omega} \forall s \in S_1,\ \forall p \in P_r,\ \forall \omega \in \Omega \tag{5}$$

$$\sum_{l\in P} r^{OUT}_{sp\omega} = \sum_{g\in V} w_{spg\omega} \forall s \in S_1,\ \forall p \in P_e,\ \forall \omega \in \Omega \tag{6}$$

$$r^{IN}_{psl\omega} = \theta_{lp} r^{OUT}_{sp\omega} \forall s \in S_1,\ \forall p \in P(i),\ \forall l \in P,\ \forall \omega \in \Omega \tag{7}$$

$$\sum_{v\in V}\sum_{p\in P} \alpha_{spr}(w_{spv\omega} + q_{sp}y_s) \leq u_{sr}y_s \quad \forall s \in S,\ \forall r \in R,\ \forall \omega \in \Omega \tag{8}$$

$$w_{spv\omega} \geq 0 \quad \forall s \in S,\ \forall p \in P,\ \forall v \in V,\ \forall \omega \in \Omega \tag{9}$$

$$r^{IN}_{psl\omega}, r^{OUT}_{sp\omega} \geq 0 \quad \forall s \in S_1,\ \forall p \in P(i),\ \forall l \in P,\ \forall \omega \in \Omega \tag{10}$$

$$0 \leq x_{cp\omega} \leq 1 \forall c \in C,\ \forall \omega \in \Omega,\ \forall p \in P_e \tag{11}$$

$$w_{spv\omega} \leq \overline{M} z_{spv} \quad \forall s \in S,\ \forall p \in P,\ \forall v \in V,\ \forall \omega \in \Omega \tag{12}$$

$$T_v \geq \tau_{spv} z_{spv} \geq 0 \quad \forall s \in S,\ \forall p \in P,\ \forall v \in V \tag{13}$$

$$T_v \leq \tau_{spv} z_{spv} + \overline{M}(1 - h_{spv}) \quad \forall s \in S,\ \forall p \in P,\ \forall v \in V \tag{14}$$

$$\sum_{s\in S}\sum_{p\in P} h_{spv} = 1 \quad \forall v \in V \tag{15}$$

$$\sum_{p\in P} z_{spv} \leq \overline{M} o_{sv} \quad \forall s \in S,\ \forall v \in V \tag{16}$$

$$\lambda_s = 0 \forall s \in S_2 \ \text{(initialization)} \tag{17}$$

$$0 \leq \phi_{sv} \leq M^{\lambda} o_{sv} \quad \forall s \in S,\ \forall v \in V \tag{18}$$

$$\phi_{sv} \leq T_v \quad \forall s \in S,\ \forall v \in V \tag{19}$$

$$\phi_{sv} \geq \lambda_s - M^{\lambda}(1 - o_{sv}) \quad \forall s \in S,\ \forall v \in V \tag{20}$$

$$\lambda_v \geq \lambda_s + \phi_{sv} \geq 0 \quad \forall s \in S,\ \forall v \in V \tag{21}$$

$$\lambda_c - T^{lead}_c \leq M^{lead} v_c \quad \forall c \in C \tag{22}$$

$$0 \leq \psi_c \leq \overline{M} v_c \quad \forall c \in C \tag{23}$$

$$\psi_c \leq \lambda_c - T^{Lead}_c \quad \forall c \in C \tag{24}$$

$$\psi_c \geq (\lambda_c - T^{Lead}_c) - \overline{M}(1 - v_c) \quad \forall c \in C \tag{25}$$

$$R_\omega \geq \sum_{c\in C}\sum_{p\in P_e} x_{cp\omega}E_{cp\omega} - \mu \quad \forall \omega \in \Omega \tag{26}$$

$$R_\omega \geq 0 \quad \forall \omega \in \Omega \tag{27}$$

$$\mu \geq 0 \tag{28}$$

The objective function in (3) is to minimise the total weighted expected supply chain costs across four objectives, which are the expected costs, expected unmet demand, expected lateness, and CVaR of the unmet demand. 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 the s^{th} node to the v^{th} node. The expected unmet demand cost is a penalty for not being able to fulfil the demand requirements in a given scenario, which is captured *via* constraint (4). Constraints (5), (6), and (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) and (10) define the lower bound on the number of commodities shipped, and the total inflow and outflow commodities, respectively. Constraint (11) defines the lower and upper bounds on the percentage of unmet demand.

In order to minimise the *makespan*, we employ a series of constraints from (12) to (25). We construct these constraints to approximate makespan to reduce computational time. The binary variable z_{spv} in constraint (12) captures the paths along which the commodities are supplied, aggregated across scenarios. Constraints (13), (14), and (15) compute the maximum time to receive commodities at node $v \in V$. Constraint (16) ensures that, for any scenario, the model retains information regarding which supplier provides each product and to which downstream supplier. Constraint (17) initialises the time of last delivery at S_2 suppliers. Constraints (18)–(21) compute the time of last delivery to node $v \in V$. If the lead time prescribed at a particular customer node (T^{lead}_c for $c \in C$) is less than the computed time λ_v $v \in V$, then ψ_c $c \in C$ captures this phenomenon, and it is penalised in the objective. This logic is captured by constraints (22)–(25). Finally, a CVaR formulation for unmet demand is constructed through constraints (26)–(28) and the last term in the objective function in (3).

5.2. 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_s), and such a design's expected performance estimate to the HU. This includes the optimal objective function value calculated using (3). In addition, we also present to the HU the expected unmet demand (*UD*), calculated using (29), the 95% confidence interval (CI) width of unmet demand *UD*, calculated using (30), and the expected cost of production, calculated using (31).

$$UD = \sum_{\omega\in\Omega} \mathbb{P}_\omega \sum_{c\in C} \sum_{p\in P_e} x_{cp\omega} E_{cp\omega} \tag{29}$$

$$CI\ \ width = 2\frac{1.96}{\sqrt{|\Omega|}}\sqrt{var(x_{cp\omega}E_{cp\omega})} \tag{30}$$

where $var(x_{cp\omega}E_{cp\omega})$ is the variance of the random variable $x_{cp\omega}\cdot E_{cp\omega}$. Assuming that these scenario-dependent unmet demand variables are normally distributed, the *CI width* is computed using the standard Z-score for the 95^{th} confidence interval (Casella & Berger, 2024).

$$\sum_{s\in S} f_s y_s + \sum_{\omega\in\Omega} \mathbb{P}_\omega \sum_{s\in S}\sum_{p\in P}\sum_{v\in V} d_{ipc\omega} w_{spv\omega}. \tag{31}$$

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's performance is deemed unrealistic (i.e., the performance is based on a single point estimate, hence a confidence interval cannot be computed), 2) the percentage of unmet demand is too high, 3) the *CI width* of unmet demand is too wide, 4) there is an unrealistic production cost, and 5) satisfied with the solution and its performance or a maximum number of iterations has been reached.

Based on which option is selected, a follow-up refinement question is asked of the HU. Specifically, if they select option 1, then "What is the percentage of anticipated variation in demand?" is asked. The proposed model in (3)–(28) is driven by the demand. Therefore, the HU's response to this question is incorporated by representing customer demand as a distribution, rather than a single point value, and this distribution is characterised using the HU's *anticipated demand variation* response. We then sample values from this distribution to generate our set of demand scenarios. If option 2 is selected, the framework asks "Do you have additional suppliers that can provide a targeted product?;" if option 3 is selected, the framework asks "What is the required level of confidence in the solution's unmet demand that is deemed necessary?;" and if option 4 is selected, the framework asks "Do you have additional data on cost?" is asked. If option 5 is selected, the framework stops asking questions, and a final solution is presented to the HU. After selecting one of the five options, the HU responds to the specific question by providing information that the framework must then interpret and incorporate into the Stochastic MILP (this process is described in more detail in Section 6).

We assume we have the required information from the HU to accurately capture the inputs needed to model lateness; therefore, for this demonstration, the refinement questions do not inquire about the lateness-related input data. Further, another query could be to ask the HU about preference weights, and our framework could adopt existing approaches to incorporate an HU's updated preference weight information into the Stochastic MILP (e.g. Afsar et al., 2023; Rivera et al., 2023; Trachanatzi et al., 2020; Ruiz et al., 2019; Jatschka et al., 2019a; Jatschka et al., 2019b). As it has been studied extensively, in this study, we focus on querying the HU for other information and for this demonstration, do not inquire about preference weight information.

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 adjust the optimisation model in response to human feedback about a solution's risk profile.

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. We incorporate a *simulated HU* that selects options (described in Section 5.2) based on the information provided. The stopping criteria is either: the *number of maximum iterations* the HU sets ahead of time and is used as an input in Method 2 in Appendix A, or if the HU is satisfied. The iterative framework is implemented using Python, and the optimisation models are solved using the GUROBI solver. All computations are conducted on a single computer with an Intel Core i7 3.8 GHz with 16GB RAM.

6.1. Performance evaluation measure

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 the model defined in Section 5.1. The ground truth represents the situation when this model and all its input parameters are known. The known input data are denoted as *ground truth data*, and the details of this data set are described in Appendix B.1, B.2, B.3, B.4, and 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 apply our framework that updates the modelling fidelity based on HU feedback, which results in solving different optimisation 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 term the solution we obtain in iteration k as the *local* solution, as this solution is only based on *partial* information. The objective function value corresponding to this local solution is termed a *local objective value*. We are interested in understanding how close such local 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 optimisation 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 (32)

$$\text{Percentage distance to ground truth at iteration } k = \frac{|GTO(\mathbf{y}^*) - GTO(\mathbf{y}^k)|}{GTO(\mathbf{y}^*)} \cdot 100 \tag{32}$$

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

A key feature of our methodology is that it asks the HU to respond to different types of questions so that the method can extract a wider range of information iteratively. Thus, in this section, we design a computational experiment with the goal of demonstrating that by starting with an initial model, the framework has the capability to reach the ground truth by interacting broadly with the HU.

Table 1 provides detailed information for 18 iterations of our methodology, which includes (i) the iteration, (ii) the percent distance to the ground truth, and (iii) the local objective value (iv) the local expected cost of production, calculated in (31), (v) the 95% confidence interval width of unmet demand *UD*, calculated in (30), (vi) the expected unmet demand *UD*, calculated in (29), and (vii) the refinement to be executed, see Section 5.2 for a description. Each of these are obtained from solving the local model in each iteration k. Further, the last column (refinement option type) 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 5.2) they found unsatisfactory. Then, the framework presents a follow-up question and the

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 5.2)
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

HU responds to it. This feedback from the HU is interpreted and acted on. In what follows, we describe in detail the process step by step, but we note that the refinement executed in each refinement is not individually discussed to avoid repetition.

To guide this discussion, Figure 2 illustrates the evolution of the solution, i.e., the selected suppliers, with each iteration k. Selected suppliers are marked in blue with a check mark, the unselected are marked in gray with an x, unavailable suppliers in the current iteration are marked in white, and suppliers that are available in the next iteration are marked in gray (without a check mark).

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 where only 9 suppliers are selected. Information from the model's output is provided to the HU (as described in Section 5.2), and the HU responds that the solution's performance is deemed unrealistic because demand is only represented as a deterministic value. This results in an undefined confidence interval for unmet demand UD. 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% of the current demand estimate E_{cp}. The framework interprets this additional feedback and generates scenarios assuming demand $E_{cp\omega}$ at customer node $c \in C$ for product $p \in P_e$ varies for each scenario $\omega \in \Omega$, by employing the Monte Carlo method and sampling from the uniform distribution $U[E_{cp}(1-\Delta E), E_{cp}(1+\Delta E)]$ where E_{cp} is the initial demand at customer node $c \in C$ for product $p \in P_e$. After incorporating the new input data, the resulting Stochastic-MILP is solved. The information shown at iteration $k = 1$ indicates that the HU can now be presented a 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 utilised to generate targeted questions to extract information from the HU beneficial to the optimisation model. After being presented with the current design solution, its estimated performance, and a set of refinement questions, the HU identifies that the current design creates excess average unmet demand. The expected 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

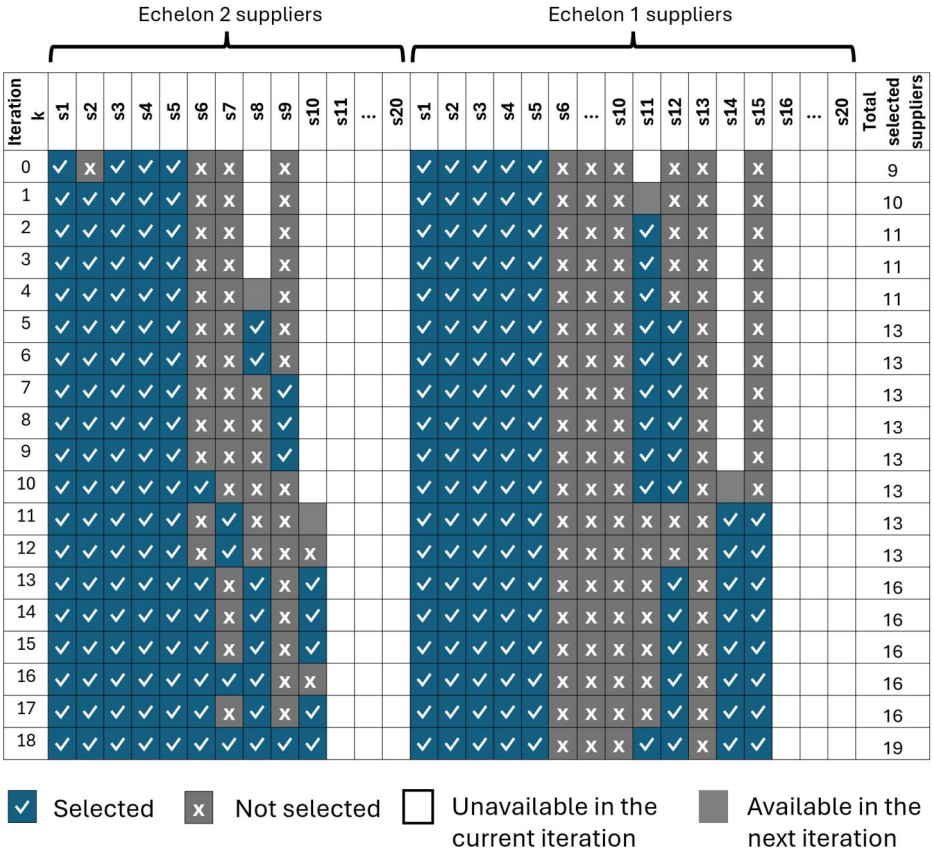

Figure 2. Evolution of the design solution, i.e., selected suppliers, across iterations.

calculated. As the goal is to minimise 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 *S2* has the highest negative dual variable value. This can be interpreted as supplier *S2 in Echelon 1* that makes end product 1, creating a bottleneck. Therefore, a targeted question is generated, "Do you know of an additional Echelon 1 supplier that could make end product 1? ." This methodology is explained in Method 1 in Section 6.3. In response, the HU provides information about a new supplier who is S11 in Echelon 1, which expands the set of suppliers to the set where $|S|^{k+1} = |S|^k + 1$ (refer to Section 6.3). This new supplier is illustrated in a gray box (available in the next iteration) in Figure 2. Then, the optimisation model is solved at iteration $k = 2$, and we observe that the newly added supplier S11 in Echelon 1 is selected as illustrated in Figure 2. In iteration $k = 2$, the objective value decreased from 1.62 to 1.42, and the expected *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. The framework asks for the value of the minimum required level of confidence (β) that should be satisfied in calculating the unmet demand. In the interpretation step, the framework sets the value $\beta = 0.2$ in (3) and sets $\omega_4 = 0.25$ (as opposed to being set to 0 in $k = 0$) so that CVaR is accounted for in the objective function. After conducting the refinement, we observed that, in iteration $k = 3$, this results in the confidence interval width of unmet demand remaining unchanged. This is because another solution with reduced risk cannot be identified with the current available suppliers. Given that the solution did not change, the distance to ground truth is unchanged; however, the local objective increased from 1.46 to 1.97 because the CVaR is now included in the local objective function.

In iteration $k = 5$, the HU thinks that the representation of the expected cost of production is inaccurate. Therefore, the framework asks "Do you have additional information on the cost of product 1?" which is denoted as $d_{s,p=1,v,\omega}$. In response, the HU provides additional information in the form of $D_p \; p \in P$, which is then converted to $D_p^d \; p \in P \; \omega \in \Omega$ using the Monte Carlo method to represent the scenario data inputs (refer to line 14 in Method 2). Here, we assume that the HU has access to the ground truth information of $D_p \; p \in P$. After solving the refinement problem formulation, in iteration $k = 6$, we observe that the local objective value has increased. Note that from iteration 5 to 10, the distance to the ground truth, illustrated in Figure 3, does not change. This occurs because, across these iterations, the selected echelon 1 suppliers remain unchanged. Although the echelon 2 supplier selections vary, multiple optimal solutions exist, specifically different combinations of selecting suppliers S6, S8, and S9, yield the same objective function value.

After presenting the solution at iteration $k = 12$ and the 95% confidence interval of the unmet demand, the HU responds that the confidence interval width of unmet demand, which is 61.8, is too wide, which implies that the solution is too risky. Then, the framework asks for another value of the minimum required level of confidence (β) that should be satisfied for the unmet demand. In iteration $k = 13$, we observed that the confidence interval width of unmet demand is reduced to 48. Notably, the objective function value increased from 2.28 to 2.75. Figure 2 illustrates that selecting more suppliers (16 suppliers as opposed to 13 suppliers in the 12^{th} iteration) increases the capacity of the network and reduces the risk of facing unmet demand. Unlike in iteration $k = 2$ to $k = 3$, the model has an increased number of potential suppliers, and by selecting more suppliers, the model can reduce the CI width of unmet demand. In iteration $k = 18$, the HU is satisfied with the solution and its performance. Therefore, the iterative refinement is terminated. We observe that a total of 19 suppliers are selected at the end of the process.

Figure 3 displays the evolution of the local objective value and the percentage distance to ground truth as a function of the human iterations.

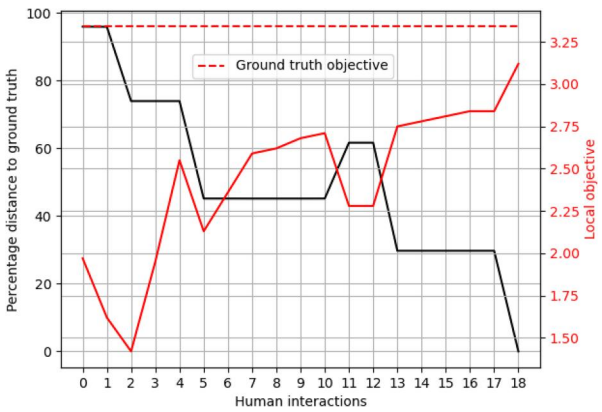

Figure 3. Evolution of local objective function value and percentage distance to ground truth with human interactions.

As refinement is carried out, the local objective value increases, and the solution generated after 18 refinement actions approaches the ground truth solution, illustrating how the framework, which is able to ask the HU for a wide range of information, can increase modeling fidelity to represent the HU's ground truth model. The average computational time to solve the Stochastic-MILP model for a given iteration is 278 s (4.6 min).

6.3. Refinement by maths-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 optimisation model. The overview of the computational experiment is described in Appendix A.

We design this experiment to consider a refinement, where the HU is queried iteratively for new potential suppliers. We consider two settings. One approach is to ask the HU for information about a random supplier and the other approach is to ask the HU targeted questions that request not just any supplier, but a supplier possessing a clearly defined capability. For the targeted approach, as described in Method 1, first we convert the Stochastic-MILP to a Stochastic-LP by substituting the integer solution $\mathbf{y}^k$ (set of selected suppliers) and obtaining the LP's dual variable values. 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 Stochastic-MILP, $\mathbf{y}^k$

1: Substitute $\mathbf{y}^k$ to Stochastic-MILP and convert it to SM-LP as in Section 4.1
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?"

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 the model defined in Section 5.1 in conducting this experiment. The experiment starts with an initial set of 20 suppliers as defined in Appendix B.5 and with 50 scenarios ($|\Omega| = 50$). Then, depending on the expected unmet demand in each iteration, the HU is asked to add additional potential suppliers whose capabilities are specified in the targeted question presented to the HU. As the refinement actions occur, the set and size of potential suppliers grow, i.e., $|S|^k = |S|^k + 1$. 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) the percentage distance to the ground truth, and 2) the computational time.

When additional potential suppliers are identified by the HU (either through a targeted question or through the alternative approach), Figure 4 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 distance to the ground truth decreases with both approaches. Yet, the method of asking targeted questions results in a sharper decrease.

Furthermore, the average computational time to execute one iteration employing targeted questions is 350s, whereas the average computational time to execute one iteration of the alternative method is 245s. We observe a 42% increase in computational

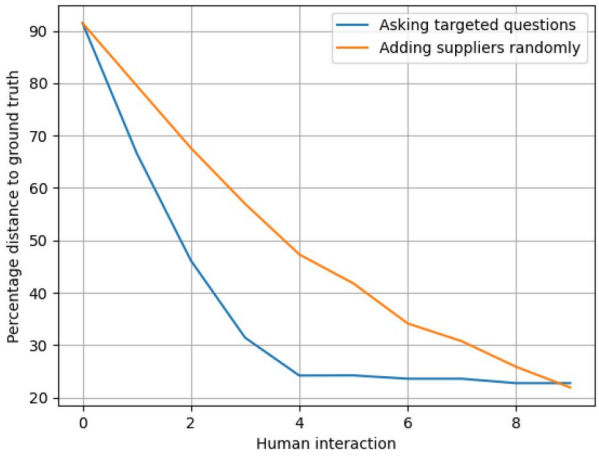

Figure 4. The percentage distance to ground truth with new potential suppliers being added to the network using the maths-informed targeted question approach versus an alternative approach that asks for a generic supplier as a function of human iterations.

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

How the solutions generated by the Stochastic-MILP will perform is uncertain and is reflected through the confidence interval of a particular measure of interest. As described in Section 4.6.2, in response to the HU's feedback that the confidence interval width of a particular objective is too wide, our framework implements a risk aversion method in the form of CVaR. This requires adjusting the worst-outcome threshold, β, according to the HU's feedback.

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 optimise 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 framework asks the HU to specify the required level of confidence β in response. In each iteration, the HU increases the worst-outcome threshold (level of required confidence) iteratively, which forces the optimisation 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 at iteration $k = 0$ we have perfect knowledge of all the suppliers in the network, which defines the magnitudes of the sets $|S_1|^{k=0}$, $|S_2|^{k=0}$. At iteration $k = 0$, the optimisation model is initialised with $\beta = 0$ and weight $\gamma_4 = 0$, then at iteration $k = 1$, they are set to $\beta = 0.2$ and $\gamma_4 = 0.25$. The key performance indicators for this experiment are three-fold: 1) confidence interval width of unmet demand (*UD*), 2) the local objective function value, and 3) the percentage distance to the ground truth.

Figure 5 depicts the change in the confidence interval width of *UD* as the required level of confidence (β given in Equation (3)) varies from 0% confidence to 99%. When the CVaR is introduced, there is a decrease in the confidence interval width. As the level of confidence increases, the confidence width continues to decrease. Figure 6 depicts the evolution of the local objective function value as the worst threshold of the outcome β is varied from 0% to 99%. When CVaR is introduced, there is an increase in the objective function value (which is interested in minimising 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. The average computational time per iteration for this experiment was 241 s.

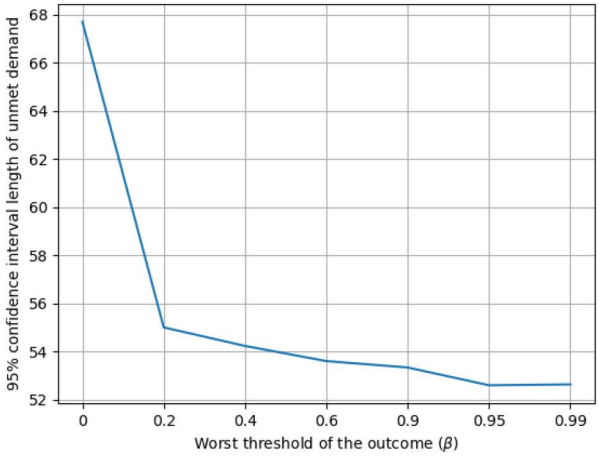

Figure 5. Evolution of confidence interval width of the unmet demand with different levels of confidence β.

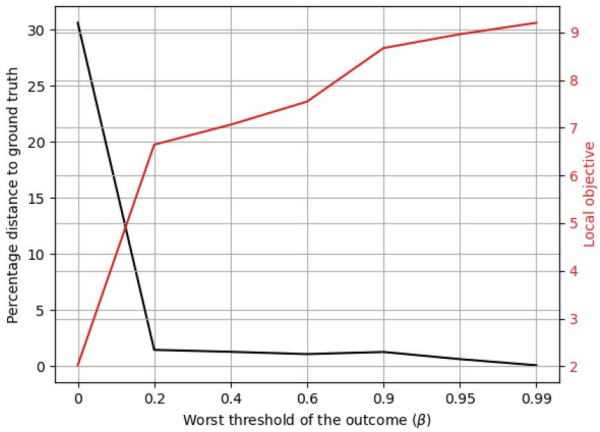

Figure 6. Evolution of confidence interval width of the unmet demand with different levels of confidence β.

7. Conclusion and future works

In this article, we propose an interactive optimisation framework to elicit a broader range of information from an HU to increase the modelling fidelity of a stochastic multi-objective mixed integer linear programming (Stochastic-MILP) model. The IO framework is designed with a goal for the Stochastic-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 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's performance too uncertain.

While this work focused on providing an IO framework to elicit a broad range of knowledge from an HU, more research is needed to provide guidance on how to generate appropriate questions. The framework presented requires the set of refinement questions to be generated for the specific optimisation model of interest, and future research could also explore whether more general questions could be created to further generalise 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. The design of user interfaces that are both intuitive and easy to use is critical to the deployment of IO frameworks, and future work on the design of a suitable interface to assist the HU in the process is needed. Moreover, different approaches to refinement, such as those that consider multiple pieces of HU feedback in parallel or creating a refinement that allows a HU to fundamentally change model structure or to fix parts of the solution, are interesting areas for further exploration.

Acknowledgment

We would like to thank the reviewers for their insightful feedback to improve the quality of the manuscript.

Author contributions

Credit: **Damsara Jayarathne**: Conceptualization, Formal analysis, Methodology, Software, Writing – original draft; **Sandipan Mishra**: Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing; **John E. Mitchell**: Conceptualization, Funding acquisition, Methodology, Supervision, Writing – review & editing; **Jennifer Pazour**: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

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

References

Afsar, B., Silvennoinen, J., Misitano, G., Ruiz, F., Ruiz, A. B., & Miettinen, K. (2023). Designing empirical experiments to compare interactive multiobjective optimization methods. *Journal of the Operational Research Society*, *74*(11), 2327–2338. https://doi.org/10.1080/01605682.2022.2141145

Ahani, N., Andersson, T., Martinello, A., Teytelboym, A., & Trapp, A. C. (2021). Placement optimization in refugee resettlement. *Operations Research*, *69*(5), 1468–1486. https://doi.org/10.1287/opre.2020.2093

Aouadni, S., Aouadni, I., & Rebaï, A. (2019). A systematic review on supplier selection and order allocation problems. *Journal of Industrial Engineering International*, *15*(S1), 267–289. https://doi.org/10.1007/s40092-019-00334-y

Cai, Y., Huang, G. H., Lin, Q., Nie, X., & Tan, Q. (2009). An optimization-model-based interactive decision support system for regional energy management systems planning under uncertainty. *Expert Systems with Applications*, *36*(2), 3470–3482. https://doi.org/10.1016/j.eswa.2008.02.036

Casella, G., & Berger, R. (2024). *Statistical inference*. Chapman and Hall/CRC.

Chai, J., & Ngai, E. W. (2020). Decision-making techniques in supplier selection: Recent accomplishments and what lies ahead. *Expert Systems with Applications*, *140*, 112903. https://doi.org/10.1016/j.eswa.2019.112903

Deb, K., Sinha, A., Korhonen, P. J., & Wallenius, J. (2010). An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, *14*(5), 723–739. https://doi.org/10.1109/TEVC.2010.2064323

Deb, K., Sundar, J. (2006). Reference point based multiobjective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (pp. 635–642).

Ehrgott, M. (2005). *Multicriteria optimization*. Springer.

Greco, S., Matarazzo, B., & Słowiński, R. (2008). Dominance-based rough set approach to interactive multiobjective optimization. In *Multiobjective optimization: Interactive and evolutionary approaches* (pp. 121–155) Springer. https://doi.org/10.1007/978-3-540-88908-3_5

Hanine, Y., Lamrani Alaoui, Y., Tkiouat, M., & Lahrichi, Y. (2021). Socially responsible portfolio selection: An interactive intuitionistic fuzzy approach. *Mathematics*, *9*(23), 3023. https://doi.org/10.3390/math9233023

Hassanzadeh, F., Nemati, H., & Sun, M. (2014). Robust optimization for interactive multiobjective programming with imprecise information applied to R&D project portfolio selection. *European Journal of Operational Research*, *238*(1), 41–53. https://doi.org/10.1016/j.ejor.2014.03.023

He, Y., He, Z., Kim, K.-J., Jeong, I.-J., & Lee, D.-H. (2021). A robust interactive desirability function approach for multiple response optimization considering model uncertainty. *IEEE Transactions on Reliability*, *70*(1), 175–187. https://doi.org/10.1109/TR.2020.2995752

Hsieh, T.-J. (2024). Scenario-based multi-objective optimization for manufacturing reliability with production routes and available machine types. *Computers & Industrial Engineering*, *198*, 110731. https://doi.org/10.1016/j.cie.2024.110731

Hu, S., Li, D., Jia, J., & Liu, Y. (2021). A self-learning based preference model for portfolio optimization. *Mathematics*, *9*(20), 2621. https://doi.org/10.3390/math9202621

Jamali, A., Ranjbar, A., Heydari, J., & Nayeri, S. (2022). A multi-objective stochastic programming model to configure a sustainable humanitarian logistics considering deprivation cost and patient severity. *Annals of Operations Research*, *319*(1), 1265–1300. https://doi.org/10.1007/s10479-021-04014-2

Jaszkiewicz, A., & Słowiński, R. (1999). The 'light beam search' approach – An overview of methodology applications. *European Journal of Operational Research*, *113*(2), 300–314. https://doi.org/10.1016/S0377-2217(98)00218-5

Jatschka, T., Raidl, G. R., & Rodemann, T. (2021). A general cooperative optimization approach for distributing service points in mobility applications. *Algorithms*, *14*(8), 232. https://doi.org/10.3390/a14080232

Jatschka, T., Rodemann, T., & Raidl, G. R. (2019, September). *Exploiting similar behavior of users in a cooperative optimization approach for distributing service points in mobility applications* [Paper presentation]. Machine Learning, Optimization, and Data Science: 5th International Conference, LOD 2019 (Vol. 5, pp. 738–750). Siena, Italy: Springer. https://doi.org/10.1007/978-3-030-37599-7_61

Jatschka, T., Rodemann, T., & Raidl, G. R. (2019, April). Evolutionary computation in combinatorial optimization: 19th European Conference, EvoCOP 2019, Held as Part of EvoStar 2019. A cooperative optimization approach for distributing service points in mobility applications (Vol. 19, pp. 1–16). Leipzig, Germany: Springer.

Jatschka, T., Rodemann, T., & Raidl, G. R. (2021). *Large neighborhood search for a cooperative optimization approach to distribute service points* [Paper presentation]. International Conference on Metaheuristics and Nature Inspired Computing, A in mobility applications (pp. 3–17). Springer. https://doi.org/10.1007/978-3-030-94216-8_1

Kato, K., Sakawa, M., Katagiri, H., & Perkgoz, C. (2010). An interactive fuzzy satisficing method based on fractile criterion optimization for multiobjective stochastic integer programming problems. *Expert Systems with Applications*, *37*(8), 6012–6017. https://doi.org/10.1016/j.eswa.2010.02.002

Laukkanen, T., Tveit, T.-M., Ojalehto, V., Miettinen, K., & Fogelholm, C.-J. (2012). Bilevel heat exchanger network synthesis with an interactive multi-objective optimization method. *Applied Thermal Engineering*, *48*, 301–316. https://doi.org/10.1016/j.applthermaleng.2012.04.058

Li, C., & Grossmann, I. E. (2021). A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering*, *2*, 34. https://doi.org/10.3389/fceng.2020.622241

Meignan, D., Knust, S., Frayret, J.-M., Pesant, G., & Gaud, N. (2015). A review and taxonomy of interactive optimization methods in operations research. *ACM Transactions on Interactive Intelligent Systems*, *5*(3), 1–43. https://doi.org/10.1145/2808234

Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, *44*(247), 335–341. https://doi.org/10.1080/01621459.1949.10483310

Miettinen, K. (2007). Using interactive multiobjective optimization in continuous casting of steel. *Materials and Manufacturing Processes*, *22*(5), 585–593. https://doi.org/10.1080/10426910701322468

Miettinen, K., & Mäkelä, M. M. (1999). Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation. *Journal of the Operational Research Society*, *50*(9), 949–959. https://doi.org/10.1057/palgrave.jors.2600786

Miettinen, K., & Mäkelä, M. M. (2000). Interactive multiobjective optimization system www-nimbus on the internet. *Computers & Operations Research*, *27*(7–8), 709–723. https://doi.org/10.1016/S0305-0548(99)00115-X

Miettinen, K., Eskelinen, P., Ruiz, F., & Luque, M. (2010). Nautilus method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research*, *206*(2), 426–434. https://doi.org/10.1016/j.ejor.2010.02.041

Niknam, T., Azizipanah-Abarghooee, R., & Narimani, M. R. (2012). An efficient scenario-based stochastic programming framework for multi-objective optimal micro-grid operation. *Applied Energy*, *99*, 455–470. https://doi.org/10.1016/j.apenergy.2012.04.017

Ntube, N., & Li, H. (2023). Stochastic multi-objective optimal sizing of battery energy storage system for a residential home. *Journal of Energy Storage*, *59*, 106403. https://doi.org/10.1016/j.est.2022.106403

Pajasmaa, J., Miettinen, K., & Silvennoinen, J. (2025). Group decision making in multiobjective optimization: A systematic literature review. *Group Decision and Negotiation*, *34*(2), 329–371. https://doi.org/10.1007/s10726-024-09915-8

Piemonti, A. D., Babbar-Sebens, M., Mukhopadhyay, S., & Kleinberg, A. (2017). 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), 4303–4326. https://doi.org/10.1002/2016WR019987

Rivera, G., Cruz-Reyes, L., Fernandez, E., Gomez-Santillan, C., & Rangel-Valdez, N. (2023). An interactive a co-enriched with an eclectic multi-criteria ordinal classifier to address many-objective optimisation problems. *Expert Systems with Applications*, *232*, 120813. https://doi.org/10.1016/j.eswa.2023.120813

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *The Journal of Risk*, *2*(3), 21–41. https://doi.org/10.21314/JOR.2000.038

Ruiz, A. B., Ruiz, F., Miettinen, K., Delgado-Antequera, L., & Ojalehto, V. (2019). Nautilus navigator: Free search interactive multiobjective optimization without trading-off. *Journal of Global Optimization*, *74*(2), 213–231. https://doi.org/10.1007/s10898-019-00765-2

Shavazipour, B., Kwakkel, J. H., & Miettinen, K. (2025). Let decision-makers direct the search for robust solutions: An interactive framework for multiobjective robust optimization under deep uncertainty. *Environmental Modelling & Software*, *183*, 106233. https://doi.org/10.1016/j.envsoft.2024.106233

Shavazipour, B., López-Ibáñez, M., & Miettinen, K. (2021). Visualizations for decision support in scenario-based multiobjective optimization. *Information Sciences*, *578*, 1–21. https://doi.org/10.1016/j.ins.2021.07.025

Sindhya, K., Ojalehto, V., Savolainen, J., Niemistö, H., Hakanen, J., & Miettinen, K. (2014). Coupling dynamic simulation and interactive multiobjective optimization for complex problems: An apros-nimbus case study. *Expert Systems with Applications*, *41*(5), 2546–2558. https://doi.org/10.1016/j.eswa.2013.10.002

Singh, A., Minsker, B. S., & Valocchi, A. J. (2008). An interactive multi-objective optimization framework for groundwater inverse modeling. *Advances in Water Resources*, *31*(10), 1269–1283. https://doi.org/10.1016/j.advwatres.2008.05.005

Tabatabaei, M., Hartikainen, M., Sindhya, K., Hakanen, J., & Miettinen, K. (2019). An interactive surrogate-based method for computationally expensive multiobjective optimisation. *Journal of the Operational Research Society*, *70*(6), 898–914. https://doi.org/10.1080/01605682.2018.1468860

Trachanatzi, D., Rigakis, M., Marinaki, M., & Marinakis, Y. (2020). An interactive preference-guided firefly algorithm for personalized tourist itineraries. *Expert Systems with Applications*, *159*, 113563. https://doi.org/10.1016/j.eswa.2020.113563

Vallerio, M., Hufkens, J., Van Impe, J., & Logist, F. (2015). An interactive decision-support system for multi-objective optimization of nonlinear dynamic processes with uncertainty. *Expert Systems with Applications*, *42*(21), 7710–7731. https://doi.org/10.1016/j.eswa.2015.05.038

van Vliet, A., Boender, C. G. E., & Rinnooy Kan, A. H. (1992). Interactive optimization of bulk sugar deliveries. *Interfaces*, *22*(3), 4–14. https://doi.org/10.1287/inte.22.3.4

Willems, S. P. (2008). Data set—Real-world multiechelon supply chains used for inventory optimization. *Manufacturing & Service Operations Management*, *10*(1), 19–23. https://doi.org/10.1287/msom.1070.0176

Yamchi, H. B., Safari, A., & Guerrero, J. M. (2021). A multi-objective mixed integer linear programming model for integrated electricity-gas network expansion planning considering the impact of photovoltaic generation. *Energy*, *222*, 119933. https://doi.org/10.1016/j.energy.2021.119933

Zhang, M., Pazour, J., Mishra, S., Mitchell, J. E., Jayarathne, D. (2023). Classification of human enrichment and refinement in interactive optimization. In *Proceedings of the IISE Annual Conference & Expo 2023, IISE.*

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.

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 B.3, the arcs of the network (replication-dependent) as in C.1 and data specific to ground truth as in Appendix B.4
6: Solve Stochastic-MILP (Section 5.1) using *GTD*
7: Generate *local data* (denoted as $LD(s,\omega)$ where $s \in S^k \ \omega \in \Omega$) by setting the initialisation 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 Stochastic-MILP with *local data* $LD(s,\omega)s \in S^k \ \omega \in \Omega^k$ to get $\mathbf{y}^k$
10: **while** $k <$ max *refinements* **do**
11: Record (local) information as in Section 5.2 to present to HU
12: Present the selectable options to the HU as in Section 5.2 and record HU's response
13: Present the relevant question to the HU as in Section 5.2 and record HU's feedback
14: Interpret the HU feedback
15: Do refinement: refinement for demand estimate, new suppliers, CVaR, cost. update model using local data $LD(s,\omega)s \in S^k \ \omega \in \Omega^k \leftarrow GroundTruthData\ (GTD)$
16: Solve Stochastic-MILP with *local data* $LD(s,\omega)s \in S^k \ \omega \in \Omega^k$ to get $\mathbf{y}^k$
17: $k \leftarrow k+1$
18: **end while**
19: *replication* $\leftarrow$ *replication* $+ 1$
20: **end while**

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 initialisation. In each refinement step k, we solve the relevant Stochastic-MILP model, which provides a local solution. The obtained information is recorded and presented to the HU along with the selectable options. Then, the HU chooses a refinement to carry out, which is interpreted to be included in the optimisation model. Note that in line 15, if CVaR is executed in k^{th} iteration, $\gamma_{1,2,3} = 0.25$ (as opposed to $\gamma_{1,2,3} = 0.33$), $\gamma_4 = 0.25$ (as opposed to $\gamma_4 = 0$) and remains the same throughout that particular replication. The data that is utilised 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

B.1. Set generation

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

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

B.2. Generation of replication-independent

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

We generate θ_{lp} which is the ratio between the commodities assembled as,

$\theta_{lp} = 4, 2, 1, 3, 1, 1, 1, 1, 1, 1, 1$ for $p = 1.....10, p \in P_r$ and $l = 11, l \in P_e$.

$\theta_{lp} = 4, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1$ for $p = 1.....10, p \in P$ and $l = 12, l \in P_e$, else, $\theta_{lp} = 0$.

Resource consumption: $\alpha_{spr} = 0.3 \forall s \in S, p \in P, r \in R$

Capacity of suppliers: $u_{sr} = 620 \forall s \in S_1, \forall r \in R$ and $u_{sr} = 700 \forall s \in S_2, \forall r \in R$

The remaining parameters are generated as follows: $\overline{M} = 1000$, $M^{\lambda} = 100$, $M^{lead} = 100$, $\gamma_m = 1/|M| \forall m \in \{1, 2, 3, 4\}$ and $\Delta_m = [2.5 \times 10^{-6}, 0.1, 0.02, 0.1]$

B.3. Generation of replication-dependent data

We generate replication-dependent input data for demand based on the following distribution as: $E_{cp\omega} \sim U[5, 120] \forall c \in C \forall p \in P_e \forall \omega \in \Omega$.

The time to ship commodities between suppliers and/or customers are given by: $\tau_{spc} \sim U[2, 4] \forall s \in S, v \in V, p \in P$, $T_c^{lead} \sim U[4, 6] \forall c \in C$.

Returned commodities are defined by: $q_{cp} \sim U[q_{LB}, 2q_{LB}] \forall c \in C$ and $p \in P$ where $q_{LB} = \sum_{\omega \in \Omega} \frac{\mathbb{P}_\omega E_{cp\omega}}{|\Omega|}$.

B.4. Ground truth specific data

Transportation cost: $d_{spc\omega} = L_{s,c} D_{p\omega}$ where $L_{s,c} \sim U[0.5, 1] \forall s \in S, c \in C$ is the distance matrix and $D_{p\omega} \sim U[50, 200] \forall p \in P, \forall \omega \in \Omega$ is the transportation cost per unit per unit distance. The threshold of the worst outcome (β) is set to 0.99.

B.5. Local model initialisation at iteration $k = 0$

Number of potential suppliers: $|S_1|^{k=0} = 10$, $|S_2|^{k=0} = 10$ as $S_1 = 1, ..., 10$ and $S_2 = 1, ..., 10$.

Number of scenarios: $|\Omega|^{k=0} = 1$, $\beta = 0$, $\gamma_{1,2,3} = 0.33$, $\gamma_4 = 0$ (ignoring CVaR at the start and $\gamma_{1,2,3,4} = 0.25$ whenever CVaR is executed and remains the same throughout the process.)

Transportation cost: $d_{ipc\omega} = L_{s,c} D_{p\omega}$ where $D_{s,c} \sim U[0.5, 1] \forall s \in S, c \in C$ is the distance matrix and $D_{p\omega} \sim U[50, 55] \forall p \in P, \forall \omega \in \Omega$ 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 C1, which is a modified version of the 2-tier networks based on empirical data from (Willems, 2008). There are 20 Echelon-2 potential suppliers (n_1) that can provide the commodities $1, ..., 10$. 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 that establish the connections between each supplier are defined as I_{spv} for $s \in S$, $p \in P$, $v \in V$, which is given below.

C.1. Set of arcs in the network

We generate the arcs, I_{spv}, 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_{spv}, between echelon 1 and customers such that all arcs are equally likely and all the customers receive commodities from echelon 1 suppliers.

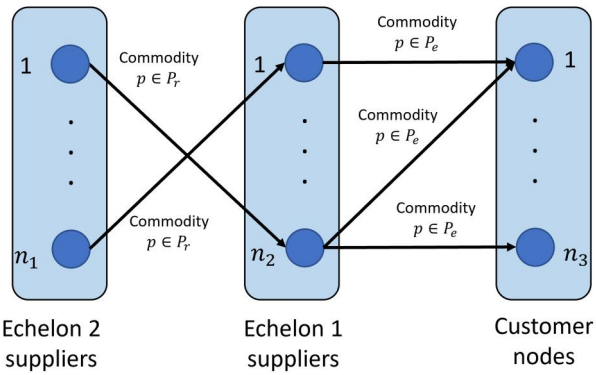

Figure C1. Two-echelon supply chain network.