Environmental Modelling & Software 74 (2015) 114–129

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

An open source framework for many-objective robust decision making

David Hadka [a, *], Jonathan Herman [f], Patrick Reed [b], Klaus Keller [c, d, e]

[a] Applied Research Laboratory, The Pennsylvania State University, University Park, PA, USA
[b] School of Civil and Environmental Engineering, Cornell University, Ithaca, NY, USA
[c] Department of Geosciences, The Pennsylvania State University, University Park, PA, USA
[d] Department of Engineering and Public Policy, Carnegie Mellon University, Pittsburgh, PA, USA
[e] Earth and Environmental Systems Institute, The Pennsylvania State University, University Park, PA, USA
[f] Department of Civil & Environmental Engineering, University of California, Davis, CA, USA

ARTICLE INFO

Article history:
Received 9 December 2014
Received in revised form
22 July 2015
Accepted 23 July 2015
Available online 1 October 2015

Keywords:
Robust decision making
Deep uncertainty
Threshold
Scenario discovery
Risk assessment
Robustness

ABSTRACT

This study introduces a new open source software framework to support bottom-up environmental systems planning under deep uncertainty with a focus on many-objective robust decision making (MORDM), called OpenMORDM. OpenMORDM contains two complementary components: (1) a software application programming interface (API) for connecting planning models to computational exploration tools for many-objective optimization and sensitivity-based discovery of critical deeply uncertain factors; and (2) a web-based visualization toolkit for exploring high-dimensional datasets to better understand system trade-offs, vulnerabilities, and dependencies. We demonstrate the OpenMORDM framework on a challenging environmental management test case termed the "lake problem". The lake problem has been used extensively in the prior environmental decision science literature and, in this study, captures the challenges posed by conflicting economic and environmental objectives, a water quality "tipping point" beyond which the lake may become irreversibly polluted, and multiple deeply uncertain factors that may undermine the robustness of pollution management policies. The OpenMORDM software framework enables decision makers to identify policy-relevant scenarios, quantify the trade-offs between alternative strategies in different scenarios, flexibly explore alternative definitions of robustness, and identify key system factors that should be monitored as triggers for future actions or additional planning. The web-based OpenMORDM visualization toolkit allows decision makers to easily share and visualize their datasets, with the option for analysts to extend the framework with customized scripts in the R programming language. OpenMORDM provides a platform for constructive decision support, allowing analysts and decision makers to interactively discover promising alternatives and potential vulnerabilities while balancing conflicting objectives.

Software availability

- **Name of Software**: OpenMORDM
- **Description**: OpenMORDM is an open-source R library for multiobjective robust decision making (MORDM). It includes support for loading datasets from a number of sources including CSV, XLS, XLSX, databases, and R matrices and data frames; visualizing the data sets using various 2D and 3D plots; performing scenario discovery and trade-off analysis; and computing uncertainty/robustness metrics. OpenMORDM also includes a web-based data exploration and visualization toolkit.
- **Developer: D**. Hadka (dmh309@psu.edu) with contributions by P. Reed and K. Keller.
- **Funding Source**: Development was partially supported by the National Science Foundation through the Network for Sustainable Climate Risk Management (SCRiM) under NSF cooperative agreement GEO-1240507 as well as the Penn State Center for Climate Risk Management.
- **Source Language**: R
- **Supported Systems**: Unix, Linux, Windows, Mac
- **License**: GNU General Public License, Version 3

* Corresponding author.
E-mail addresses: dmh309@psu.edu (D. Hadka), jdherman@ucdavis.edu (J. Herman), patrick.reed@cornell.edu (P. Reed), klaus@psu.edu (K. Keller).

http://dx.doi.org/10.1016/j.envsoft.2015.07.014

- **Availability**: http://github.com/dhadka/OpenMORDM

1. Introduction

A critical component of environmental planning and management is the search for robust solutions capable of withstanding deviations from our best projections of the future. This challenge is amplified by the presence of deep uncertainty, where the suite of all possible future events as well as their associated probability distributions are themselves uncertain (e.g., future climatological and hydroeconomic factors [Knight (1921), Lempert (2002), Olson et al. (2012)]). These challenges have led to several "bottom-up" decision support frameworks [Nazemi and Wheater (2014), Weaver et al. (2013)], which move beyond trying to predict the most probable future(s) to discover which states of the world (SOWs) may lead to high consequence system vulnerabilities. This step helps with the task of evaluating the likelihoods of the discovered system vulnerabilities as it can help to focus the analysis on a subset of plausible future scenarios (e.g., Lempert et al. (2012)). Bottom-up or robustness-based approaches include Decision Scaling [Brown (2010), Brown et al. (2012)], Information-Gap (Info-Gap) [Ben-Haim (2004)], Robust Decision Making (RDM) [Lempert (2002), Lempert et al. (2013, 2006), Groves and Lempert (2007), Lempert and Collins (2007)], and Many-Objective Robust Decision Making (MORDM) [Kasprzyk et al. (2013)]. As highlighted by Herman et al. (2015), these bottom-up frameworks can be generalized into four steps: identifying decision alternatives, sampling states of the world, specifying robustness measures, and performing scenario discovery to identify the most important uncertainties. The final step, scenario discovery, is commonly used to find policy-relevant controls by determining the ranges of each uncertainty leading to system failure [Lempert et al. (2006)]. Herman et al. (2015) note that while these methods are often defined at a conceptual level, specific implementations share a number of potentially interchangeable concepts which should be compared to understand consequences for decision support. This work addresses the need for software and visualization tools to flexibly support the quantitative components of these "bottom-up" environmental systems planning frameworks, which share the goal of identifying robust solutions. The following paragraphs introduce the conceptual frameworks for decision support under deep uncertainty, while the quantitative methods implemented in this work are described in Section II.

Robust Decision Making (RDM), like other "bottom-up" approaches, seeks to distinguish robust solutions which provide satisfactory performance across many plausible SOWs [Lempert (2002), Lempert et al. (2013), Groves and Lempert (2007), Lempert and Collins (2007)]. Given a pre-specified set of alternatives to analyze, RDM subjects each to an ensemble of SOWs that are treated as exploratory samples over plausible ranges of uncertain factors [Bryant and Lempert (2010), Groves and Lempert (2007), Lempert et al. (2006, 2012)]. The goal — as is generally the case in Decision Scaling and Info-Gap — is to identify future scenarios that may cause the system to fail. RDM studies often adopt a "satisficing" approach [Simon (1959)], in which solutions must satisfy performance requirements across many plausible futures rather than provide optimal performance in a single future. Using a satisficing approach, robustness can be quantified with the domain criterion [Schneller and Sphicas (1983), Starr (1962)] which aims to maximize the volume of the uncertain factor space in which performance requirements are satisfied [Lempert and Collins (2007)]. Additionally, RDM analyses typically employ the Patient Rule Induction Method (PRIM) [Friedman and Fisher (1999)] to perform a high-dimensional sensitivity analysis for scenario discovery in order to identify the ranges of uncertain factors most likely to cause system failure [Bryant and Lempert (2010), Groves and Lempert (2007), Lempert et al. (2006, 2008)]. RDM builds upon exploratory modeling [Bankes (1993), Kwakkel and Pruyt (2013)] by providing a systematic approach to identifying vulnerabilities.

Info-Gap analysis aims to quantify the maximum allowable deviation of deeply uncertain system factors that can be tolerated while still satisfying performance requirements [Ben-Haim (2004), Hipel and Ben-Haim (1999), Hall et al. (2012)]. Uncertain factors are sampled radially outward from a baseline (expected) future state of the world until a failure condition is reached; the distance from the baseline at which this occurs is termed α, or the "uncertainty horizon" [Hall et al. (2012), Korteling et al. (2013)]. Note that in this definition, there is no mention of probability distributions for the uncertainties. Rather, α defines the distance in the space of deeply uncertain factors between the baseline (expected) state of the world and the nearest state of the world in which the model predicts system failure. It assumes that a larger value of α implies the system is more resilient to perturbations in the deeply uncertain parameters. However, α fails to identify which specific uncertain factors, or combinations of factors, predict system failure. Recent examples of Info-Gap applications in water resources planning problems include Hine and Hall (2010) and Matrosov et al. (2013).

Decision Scaling, like other "bottom-up" approaches, inverts the decision making process. Rather than focusing on predictive distributions (derived, for example, by downscaling of Atmospheric-Ocean General Circulation Model (AOGCM) projections), Decision Scaling first aims to identify thresholds likely to trigger consequential system risks. The approach is a three-step process of (1) identifying key concerns and decision thresholds, (2) modeling the response to changing environmental conditions, and (3) estimating the relative probability of the critical environmental thresholds being crossed [Brown et al. (2012)]. Decision Scaling studies typically focus on uncertain climate factors, though recent work extends the approach to include hydroeconomic factors [Ghile et al. (2014), Lownsbery (2014)]. Decision Scaling's most significant difference from the other decision support frameworks is its assumption that the likelihoods associated with changes in temperature and precipitation can be inferred as subjective probabilities. The subjective probabilities are developed via expert evaluations of how SOWs attained from statistical weather generators relate to AOGCM projections [Brown et al. (2012)]. Decision Scaling has been most widely used as a discrete choice framework for choosing between pre-specified design alternatives [e.g., Moody and Brown (2013)] or as a vulnerability analysis to characterize the risks of existing systems [Ghile et al. (2014), Turner et al. (2014)].

In the Decision Scaling and Info-Gap frameworks, it is common to analyze a relatively small set of discrete decision alternatives that are pre-specified by stakeholders. This reflects a high degree of knowledge about system behavior under uncertainty, and may cause an analysis to be vulnerable to a significant status quo bias [Brill et al. (1990)]. Furthermore, pre-specified alternatives may overlook important trade-offs between conflicting objectives that reflect decision relevant performance requirements or tensions between stakeholders [Herman et al. (2014)]. RDM analyses can also suffer from these issues if the practitioner explores only a fixed set of alternatives. To overcome these challenges, Kasprzyk et al. (2013) propose Many-Objective Robust Decision Making (MORDM), in which alternatives are discovered via many-objective optimization in the projected future state of the world. MORDM supports constructive learning to improve decisions for complex, ill-defined environmental planning and management problems. This follows the framework of Many-Objective Visual Analytics (MOVA) [Woodruff et al. (2013)], a foundation for constructive decision aiding [Tsoukias (2008), Roy (1999)] in which problem framing is performed interactively with stakeholder feedback.

MORDM begins with problem formulation, where the computational model, performance objectives, and well-characterized uncertainties are elicited from stakeholders. Additionally, key decisions or alternative hypotheses of system performance are articulated. The problem formulation is then subjected to many-objective search to generate alternative designs (or solutions) to the problem. With many objectives, there is typically no single optimal design. Instead, there exists a collection of Pareto optimal designs that can be approximated by computational search, where improvements in one performance objective require a degradation in one or more other objectives [Pareto (1896)]. This "generate first, choose later" [Cohon and Marks (1975), Maass et al. (1962)] *a posteriori* elicitation of preference is well established in water resources planning. The process of understanding performance tradeoffs is typically supported by interactive visualization software [e.g., Kollat and Reed (2007)].

In the MORDM approach, robustness is assessed *a posteriori* for each Pareto-approximate solution (i.e., the solutions which are found to be non-dominated) using a four step process. (1) First, performance requirements and system constraints are elicited from stakeholders in the context of available performance trade-offs. (2) The deeply uncertain factors in the model must be identified and sampled based on an appropriate design of experiments developed in consultation with stakeholders. This step widens the envelope of deeply uncertain SOWs that will be used to evaluate the robustness of decision alternatives. (3) The vulnerabilities, trade-offs, and dependencies of the system are evaluated by subjecting Pareto-approximate alternatives (under well-characterized uncertainty) to the deeply uncertain SOWs sampled in the prior step. These SOWs represent hypothetical but plausible conditions that may be encountered in the uncertain future. (4) Finally, the solutions are annotated with one or more measures of robustness and explored using interactive visualization tools. Because the framework is inherently multi-objective, the robustness goals of multiple stakeholders can be considered, which may conflict [Herman et al. (2014)]. MORDM is able to quantify the trade-offs between robustness and the various performance objectives, explore the dependencies between uncertainties and system performance, and identify the vulnerable states of the system. MORDM provides an opportunity to flexibly bridge and advance bottom-up methodologies.

The choice of decision support framework under deep uncertainty (e.g., Decision Scaling, Info-Gap, RDM, MORDM) depends on the knowledge and preferences of the decision makers, the intrinsic predictability of the system under study, the quality of available computational models, and the types of answers sought by the decision makers (e.g., identifying key parameters impacting robustness versus quantifying resilience to small perturbations). Herman et al. (2015) highlight the importance of consistently choosing methodological options that support *a posteriori* decision support (e.g., that generate alternatives prior to expressing preferences [Cohon and Marks (1975)], and globally explore future SOWs to discover sensitive uncertainties rather than assuming them in advance of analysis [Bryant and Lempert (2010)]). Herman et al. (2015) also note that currently popular decision support frameworks focused on robustness to uncertainty share many interchangeable ideas, and need not be considered separate.

MORDM is a comprehensive conceptual framework that includes optimization, scenario discovery, and interactive visualization, and is a viable target for an open source software package to unify many of the ideas in bottom-up decision support frameworks. Moreover, given the growing interest among stakeholders in decision support frameworks to address deep uncertainty, the R basis of OpenMORDM is highly complementary to other open source tools such as the Python-based EMA Workbench [Kwakkel (2015)] for expanding access to large communities of practice. There is a strong opportunity for the R and Python open source programming communities to collaborate to expand the scope and quality of tools available for exploratory modeling, sensitivity analysis, tradeoff analysis, and assessments of robustness.

While developing a consistent application programming interface (API) for MORDM, we considered the availability of existing open source tools. For optimization, we considered metaheuristics designed for solving multi-objective, non-linear, mixed discrete and continuous, disjoint, multimodal problems commonly found in complex planning or design applications such as water resources management [Coello Coello et al. (2007), Nicklow et al. (2010)]. Metaheuristics are available as individual software libraries available from the original authors, such as AMALGAM [Vrugt and Robinson (2007)] or the Borg MOEA [Hadka and Reed (2013)], or contained within comprehensive software frameworks for multiobjective optimization, such as Paradiseo [Cahon et al. (2004)], JMetal [Durillo et al. (2010)], or the MOEA Framework [Hadka (2014)]. For scenario discovery, two commonly-used algorithms are the Patient Rule Induction Method (PRIM) and Classification and Regression Trees (CART). PRIM and CART are introduced in detail in section 2.1. There are two open-source implementations of PRIM written in the R programming language, including the original non-interactive `prim` package [Duong (2014)] and the interactive `sdtoolkit` package [Bryant (2014)]. `sdtoolkit` interactively allows the user to select the desired coverage and density of the generated PRIM hyperboxes. CART is available in the `rpart` package within R. Finally, there exist several toolkits to support interactive visualization. For example, Kollat and Reed (2007) developed the Visually Interactive Decision-making and Design using Evolutionary Multi-objective Optimization (VIDEO) software. VIDEO encompasses the optimization and interactive visualization aspect, but lacks scenario discovery and robustness quantification.

This study contributes OpenMORDM, an open-source implementation of MORDM in the R programming language that encompasses all three aspects: optimization, scenario discovery, and interactive visualization. R is selected as the host language due to its prevalent use in environmental modeling, the ability to leverage a powerful functional scripting language to support extensibility, and the availability of prebuilt analytical and statistical packages. We will demonstrate the decision support capabilities of OpenMORDM using a conceptually simple but sufficiently challenging hypothetical environmental management problem called the "lake problem" [Carpenter et al. (1999), Singh et al. (2015)]. Using OpenMORDM, we will demonstrate how to generate near-optimal alternative designs in the best available projection of the future SOW, sample deeply-uncertain SOWs, explore competing hypotheses for alternative definitions of robustness, and identify key deeply uncertain factors and scenarios that should inform ex-post monitoring of system performance. While OpenMORDM focuses on the application of MORDM, the software is not restricted in scope to MORDM. Rather, it is viewed as a decision support library empowered by a collection of complementary analysis tools. The ultimate goal is addressing the relevant questions and concerns of the decision maker, and allowing the decision maker to explore the impacts and significance of alternative management actions and conceptions of robustness.

The remainder of this paper is organized as follows. Section 2 discusses the software architecture of OpenMORDM. Section 3 introduces the lake problem. Section 4 demonstrates applying MORDM on the lake problem using OpenMORDM. Particular attention is given to the software API that facilitates MORDM and supports user-customizable visualizations and analyses. Section 5 concludes this paper by providing recommendations for further enhancements to OpenMORDM.

2. OpenMORDM

OpenMORDM is an open-source implementation of MORDM in the R programming language. Readers interested in in-depth conceptual details on MORDM should refer to Kasprzyk et al. (2013) and Herman et al. (2015). The remainder of this section is organized as follows. Section 2.1 discusses the overall MORDM workflow and the software architecture of OpenMORDM for supporting this workflow. Section 2.2 discusses the robustness measures built into OpenMORDM for quantifying the impacts of deep uncertainty on a model.

2.1. OpenMORDM software architecture

OpenMORDM is developed as an R package. As such, it provides a collection of data structures and functions (the API) within R for performing all of the functionality required by MORDM. Fig. 1 shows the overall software architecture of OpenMORDM. The API can be accessed through an interactive R session or through a web-based visualization toolkit accessible from any WebGL-enabled web browser. The web-based visualization toolkit, powered by Shiny [RStudio, Inc. (2014)], provides a standardized, interactive interface for visualizing, exploring and analyzing many-objective datasets. Furthermore, being a web-based technology, datasets can be publicly or privately shared with colleagues by instantiating a Shiny web server. See the documentation for the `explore` command for details on starting the web visualizations.

OpenMORDM supports a variety of data access mechanisms for generating or loading datasets. If the computational model is available as an R function or a standalone executable, OpenMORDM can evaluate designs directly against the model. Any R function of the form is

```
objective.fcn <- function(variables) {
        # Evaluate model using the provided input variables
        return(list(objectives, constraints))
}
```

supported. If the model is a standalone executable that resides on the host computer, then OpenMORDM can invoke the executable using the standardized external problem interface defined by the MOEA Framework [see Chapter 5.2 in Hadka (2014)]. This external problem interface uses the standard input/output channels to pass the input variables to the executable and read the resulting objectives and constraints. Alternatively, one could use the Model-R Coupler to interface with standalone executables [Wu et al. (2014)].

OpenMORDM can access pre-computed datasets stored in a database or structured input files. Any database with a corresponding R driver is supported (e.g., RPostgreSQL for accessing PostgreSQL databases [Conway et al. (2013)]). To remain agnostic of the database and table structure, OpenMORDM requires the decision maker to write the appropriate SQL queries (or other lookup mechanisms for non-SQL databases) for accessing the data. OpenMORDM also supports loading datasets stored in comma-separated value (CSV) files and some proprietary spreadsheet documents (XLS or XLSX).

OpenMORDM defines functions for each stage of the MORDM process. New problem formulations are defined using the `define.problem` command. This assumes a compatible R function or standalone executable exists as described above. In addition to providing the name of the R function or path to the executable, the user also specifies the lower and upper bounds for each model input and the preference direction for performance objectives (i.e., whether an objective is minimized or maximized).

Alternative designs are generated with the `borg.optimize` command. The `borg.optimize` command implements the Borg Multiobjective Evolutionary Algorithm (Borg MOEA) for many-objective search [Hadka and Reed (2013)]. The Borg MOEA has an established history of effectively solving many-objective problems from the water resources and broader engineering domains [Hadka et al. (2012), Hadka and Reed (2012), Woodruff et al. (2012), Reed et al. (2013), D'Ervau (2013), Giuliani et al. (2014), Hadka and Reed (2015), Singh et al. (2015)]. Note, however, that any many-objective search algorithm can be supported assuming it generates a dataset in the appropriate file format. OpenMORDM can load datasets from CSV files with `mordm.read.csv`, Excel files with `mordm.read.xls`, or load native R matrices and data frames with `mordm.read.matrix`.

After generating alternative designs, the next step in MORDM is to define and compute the robustness measures. OpenMORDM provides the `compute.robustness` function for this step. Typically, deep uncertainty requires defining multiple problem formulations for each deeply uncertain parameterization. One or more problem formulations created by the `define.problem` function can be passed as arguments. Several robustness calculations are provided by default, as described in Section 2.2. Otherwise, custom functions for computing robustness can be provided. This customization is necessary for satisficing-based robustness, where the decision maker defines required levels of performance and/or constraints specific to each application.

The result of the uncertainty function is a vector or matrix (in the case where multiple robustness measures are calculated). The dataset annotated with robustness values is visualized using `mordm.plot` to generate a 3D scatter plot of the alternative designs. As demonstrated in this study, points are color-coded by the robustness value. Nevertheless, robustness can be plotted on any axis if desired. Other 2D and 3D visualizations can be generated using functionality built into OpenMORDM, such as `mordm.plot.parallel` for generating a parallel coordinates plot synchronized with the 3D scatter plot, or plotting capabilities provided by other R packages.

For more control over the uncertainty calculations, it is often useful to split the deep uncertainty factor sampling from the robustness metric calculation. The two underlying R functions, `mordm.sample.uncertainties` and `mordm.evaluate.uncertainties` can be invoked separately. This is particularly useful for performing scenario discovery and vulnerability analysis, where the sampled uncertainty factors are required inputs.

MORDM also emphasizes the identification of key vulnerabilities and dependencies caused by deep uncertainties. For example, one could define vulnerable scenarios as those in which a solution fails to meet one or more of the stakeholders' performance requirements. OpenMORDM provides two “association rule learning” algorithms for identifying key uncertainties and their associated ranges that lead to vulnerabilities. The Patient Rule Induction Method (PRIM) attempts to find regions in uncertainty space with an above-average number of uncertainty parameterizations leading to vulnerabilities [Friedman and Fisher (1999)]. It achieves this goal by iteratively “peeling” away subregions to maximize the density of vulnerable solutions in the remaining volume, represented as a hyperbox. An alternative algorithm called Classification and Regression Trees (CART) recursively subdivides the uncertainty space into partitions exhibiting less and more frequent vulnerabilities [Breiman et al. (1984)]. The partitions are chosen to minimize Type 1 classification error (i.e., minimize the false-positive rate). Lempert et al. (2008) compared PRIM and CART when developing their RDM framework, concluding that neither approach was distinctly superior to the other. PRIM requires user

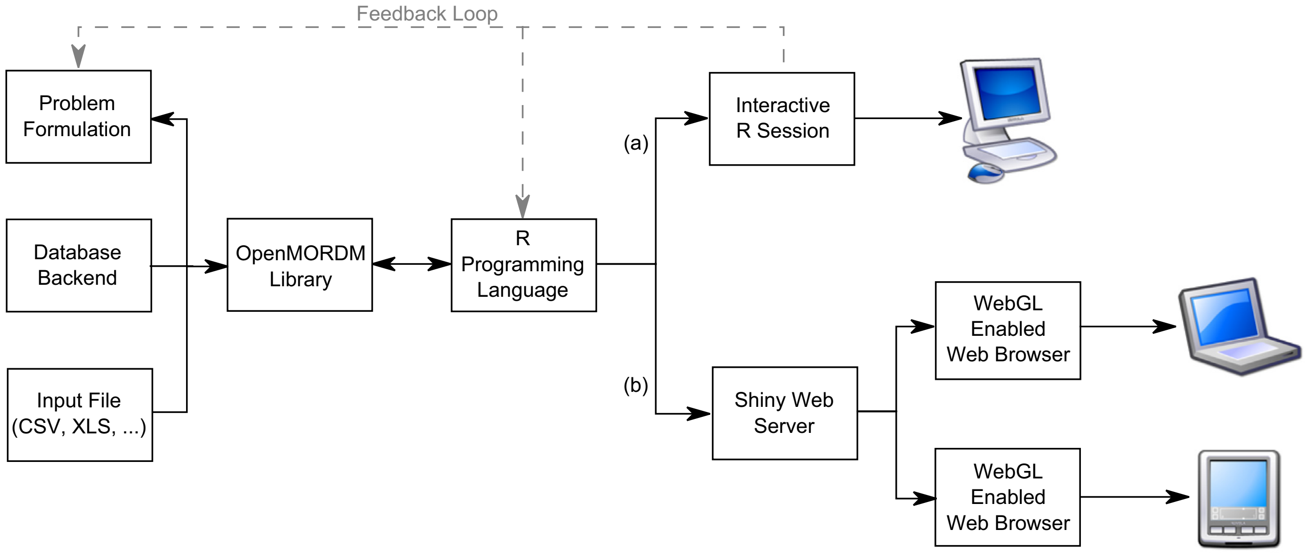

Fig. 1. Architecture of OpenMORDM. OpenMORDM is driven by data generated from candidate problem formulations that can be loaded from a database or formatted input file. The decision maker interacts with OpenMORDM in two ways: (a) an interactive R session allowing the decision maker to invoke OpenMORDM commands directly; or (b) a web-based visualization toolkit accessible via any WebGL-enabled web browser. Information learned during the analysis can feed back to modify the problem formulation, redefine robustness thresholds, or used to develop custom scripts for additional analysis.

interaction to select the appropriate peeling thresholds and stopping criteria, but produces easily-interpretable hyperboxes (i.e., factor mapping that identifies ranges of uncertainties leading to vulnerabilities). CART, on the other hand, requires no user interaction but may partition space into many disjoint sets. We do note that CART has the theoretical advantage of minimizing Type 1 classification errors [Breiman et al. (1984)]. Regardless, OpenMORDM supports both PRIM and CART via the `analyze.prim` and `analyze.cart` methods. For PRIM, OpenMORDM supports both versions from the `prim` [Duong (2014)] and `sdtoolkit` [Bryant (2014)] packages. Moreover, since the R community provides an extensive library of statistical methods, other association rule learning methods can be considered.

Sensitivity analysis is another technique to identify the vulnerabilities and dependencies within the model(s) used to support decisions. Unlike PRIM and CART, sensitivity analysis will not identify the specific ranges that lead to vulnerabilities, but it will rank order the uncertain factors by their influence on metrics of focus (i.e., factor prioritization [Saltelli et al. (2008)]). The R sensitivity package provides numerous sensitivity analysis routines, including ANOVA, Morris' elementary effects, and Sobol' sensitivity indices [Pujol et al. (2014)]. The `compute.sensitivity` function in OpenMORDM provides a standardized interface to these sensitivity routines. Additionally, we include an implementation of the delta-moment algorithm by Plischke et al. (2013) within OpenMORDM. The delta-moment method is notable due to its ability to compute sensitivities from “given data”. Whereas traditional sensitivity analysis routines (e.g., Sobol' sensitivity analysis) require specific sampling techniques, the delta-moment method computes sensitivities from arbitrary datasets.

OpenMORDM contains many additional functions not mentioned here for brevity. Documentation for these functions is provided in the OpenMORDM package. One of the advantages of using a functional language like R as the implementation language is the ability to embed hooks for extensibility. For example, users can define custom robustness functions as an argument to the `compute.robustness` method. In this manner, custom extensions can be developed to enhance OpenMORDM for any particular application.

2.2. Measuring robustness

The decision support frameworks described in Section I have proposed a number of measures to quantify robustness under deep uncertainty. OpenMORDM implements several of these to allow users to compare and select an appropriate measure for the system under consideration. The elicited definition of thresholds for robustness is a critical step in MORDM and drives the remainder of the analysis. The appropriate thresholds, based either on system constraints and/or required levels of performance, may differ across stakeholders and risk attitudes. Following Lempert and Collins (2007), Herman et al. (2015) distinguish two major classes of robustness measures in the literature: *regret* and *satisficing*. Generally speaking, regret-based robustness seeks to minimize deviations in system performance caused by deep uncertainties compared to a preferred or ideal design (i.e., minimizing expected loss), whereas satisficing-based robustness seeks to maximize the SOWs where system requirements as defined by stakeholders are met (see below for a more detailed and technical discussion). OpenMORDM implements four robustness measures. The decision support frameworks reviewed in Section 1 are defined in general terms to allow any of the following measures to be used. Herman et al. (2015) demonstrate the importance of selecting an appropriate metric regardless of the overarching framework under consideration.

The two regret-based robustness measures, Regret Type I and Regret Type II, measure the deviation in performance in deeply uncertain SOWs. Both metrics use the 90th percentile objective value in the uncertainty ensemble so capture the worst-case behavior of a solution while limiting the susceptibility to outliers. Regret Type I focuses on the impact of incorrect assumptions about the future SOW by computing the change in system performance across all sampled SOW [Kasprzyk et al. (2013)]. For a given design x,

$$\text{Regret Type I} = \max_i \left\{ \underset{s\in S}{\text{quantile}} \left(\left| \frac{f_i(x;s) - f_i(x;\bar{s})}{f_i(x;\bar{s})} \right|, 0.9 \right) \right\}, \quad (1)$$

where $f_i(x;s)$ is the value of the ith objective in SOW s, $\bar{s}$ is the baseline SOW under well-characterized uncertainties, and $\underset{s\in S}{\text{quantile}}(\cdot, 0.9)$ computes the 90th percentile value across all SOWs

$s \in S$. Minimizing Regret Type I differentiates designs with minimal fluctuation from the baseline performance across all sampled SOWs. By contrast, Regret Type II focuses on the impact caused by incorrectly selecting a design alternative for the future SOW [Savage (1951)]. We compute the degradation in system performance between the selected design and the "best" design within each sampled SOW:

$$\text{Regret Type II} = \max_i \left\{ \underset{s \in S}{\text{quantile}} \left(\left| \frac{f_i(x;s) - \sup_{y \in P} f_i(y;s)}{f_i(x;\ s)} \right|, 0.9 \right) \right\}, \tag{2}$$

where P is the set of alternative designs generated by the optimization step and $\sup_{y \in P} f(y;s)$ identifies the "best" design in P within SOW s. Here, the best design is computed as the ideal point. In multi-objective space, the ideal point marks the best value achieved in each objective. Note that the ideal point may not correspond to a valid design; instead, it serves as an upper bound on the best attainable performance in each objective. Note that both equations (1) and (2) assume all objective values are being maximized. If the objective is being minimized, then equations (1) and (2) should compute the minimum objective value.

The satisficing-based robustness measures, Satisficing Type I and Satisficing Type II, both determine the feasibility of a design with respect to some performance criteria defined by the stakeholder. The criteria are encoded with a satisficing indicator function, $I_S(\ldots)$, evaluating to 1 when all conditions are satisfied and 0 otherwise. For Satisficing Type I, feasibility is measured by the fraction of SOWs that satisfy the criteria:

$$\text{Satisficing Type I} = \frac{1}{|S|} \sum_{s \in S} I_S(f(x;s)). \tag{3}$$

Satisficing Type I is similar to the domain criterion introduced by Starr (1962). Rather than computing the volume of the uncertain factor space (i.e., the volume of sampled SOWs) satisfying the criteria, which cannot be computed precisely when using a discretized Latin hypercube sampling, Satisficing Type I instead computes the fraction of SOWs satisfying the criteria. Alternatively, Satisficing Type II measures feasibility as the maximum allowable uncertainty as a distance from the expected future SOW while still satisfying the criteria:

$$\begin{aligned}\text{Satisficing Type II} &= \text{Minimize} \|s - \bar{s}\| \text{ such that } I_S(f(x;s)) \\ &= 0 \text{ over all } s \in S.\end{aligned} \tag{4}$$

Note that Satisficing Type II is similar in nature to the approach used in Info-Gap (Ben-Haim, 2004). A larger value for Satisficing Type II indicates larger perturbations in the uncertainty factors (i.e., larger error between the expected and actual future SOW) is tolerable. To summarize, Regret Type I represents the deviation from baseline performance; Regret Type II represents the deviation from the "best" solution in each scenario; Satisficing Type I represents the volume or fraction of successful scenarios out of the total; and Satisficing Type II represents the minimum distance to a failure scenario.

3. The lake problem

In this paper, we demonstrate the OpenMORDM package on an environmental management test problem termed the "lake problem." The lake problem is a hypothetical and highly stylized decision problem where a town's inhabitants must determine the amount of allowable pollution that can be emitted into a nearby lake for a given planning horizon [Carpenter et al. (1999)]. As shown in Fig. 2, pollution enters the lake from two sources: anthropogenic pollution resulting from industrial and agricultural activities from the town, and uncontrolled natural inflows from the environment. In this exercise, we will represent the alternative values or objectives of the town's inhabitants while seeking an appropriate pollution control strategy to avoid irreversibly polluting the lake. The temporal dynamics of the pollution in the lake is approximated by

$$X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \varepsilon, \tag{5}$$

where X_t is the amount of phosphorus in the lake at time t, $a_t \in [0, 0.1]$ is the allowed anthropogenic pollution at time t (i.e., the pollution control strategy), b measures the lake's phosphorus removal rate

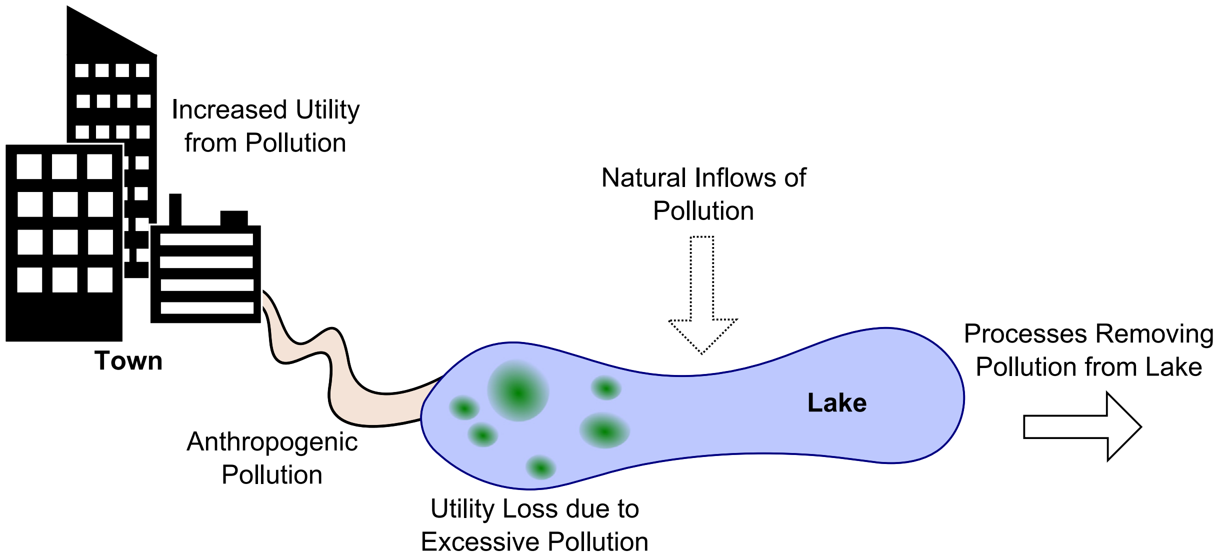
Increased Utility from Pollution
Town
Natural Inflows of Pollution
Lake
Processes Removing Pollution from Lake
Anthropogenic Pollution
Utility Loss due to Excessive Pollution

Fig. 2. Illustration of the lake problem. Pollution, in the form of phosphorus, enters the lake from runoff due to industrial and agricultural activities in the town, adding to natural inflows of pollution from the environment. The buildup of phosphorus leads to algal blooms that decrease the utility of the lake (reduced tourism, fishing, etc.). The lake is able to remove part of this pollution based on its properties.

(due to sedimentation, outflow, and biological sequestration), q defines the lake's recycling rate (primarily through sediments), and ε represents uncontrollable natural inflows of pollution modeled as a log-normal distribution with a given mean, μ, and standard deviation, σ [Singh et al. (2015)]. Here, the time t is expressed in years and we consider a planning horizon of 100 years. For certain combinations of b and q, there exists a critical phosphorus threshold, X_{crit}, above which the lake will experience irreversible eutrophication since the recycling rate exceeds the removal rate even in the absence of anthropogenic input. For more details on these parameters and their interpretation, see Carpenter et al. (1999) and Singh et al. (2015).

For the purposes of this study, we use the four-objective formulation of the lake problem developed by Ward et al. (2015). The four objectives are minimizing phosphorus in the lake, maximizing economic benefit, maximizing inertia, and maximizing reliability. The first objective, minimizing phosphorus in the lake, computes the average phosphorus level over all time periods and SOWs:

$$f_{phosphorus} = \frac{1}{|S||T|}\sum_{s\in S}\sum_{t\in T} X_{t,s}, \tag{6}$$

where S is the set of all sampled SOWs, T is the set of time steps, $X_{t,s}$ is the level of pollution in the lake at time t in SOW s. Note that $f_{phosphorus}$ is correlated with but different from the decision variables a_t in Equation (5). The decision variables control the allowed anthropogenic pollution released into the lake, such that if all other factors are kept equal, an increase in allowed anthropogenic pollution raises the average phosphorus level in the lake. However, $f_{phosphorus}$ accounts for phosphorus removal due to various natural mechanisms, recycling, and other natural sources of pollution.

Allowing increased anthropogenic pollution yields economic benefit for the town's inhabitants due to reduced control costs. The second objective approximates this economic benefit as:

$$f_{economic} = \frac{1}{|S|}\sum_{s\in S}\sum_{t\in T}\left(\alpha a_t - \beta X_{t,s}^2\right)\delta^t, \tag{7}$$

where α is the economic benefit realized per allowing a unit of anthropogenic pollution, β is a reduction in benefit resulting from polluting the lake (eutrophic cost), and δ is a fixed discount rate.

The third objective, inertia, measures the fraction of time steps where the year-to-year change in allowable pollution, $a_t - a_{t-1}$, remains below a defined threshold, τ:

$$f_{inertia} = \frac{1}{(|T|-1)}\sum_{t=1}^{|T|} I(a_t - a_{t-1} > \tau). \tag{8}$$

Since we are developing a pollution control strategy to place limits on anthropogenic pollution emitted by the town, ideally the year-to-year change in allowable pollution are below a certain threshold. Drastic changes year-to-year burdens policy makers as they must revise policies every year and as installed infrastructure may have to be changed before the end of the planned lifetime. An inertia value of 1.0 is ideal, indicating no major policy changes exceeding the threshold τ are required. In this study, τ is set for illustrative purpose to 20%.

Lastly, the fourth objective, reliability, measures the fraction of years in which the pollution level in the lake is below the critical phosphorus threshold of the lake, X_{crit}:

$$f_{reliability} = \frac{1}{|S||T|}\sum_{s\in S}\sum_{t\in T} I(X_{t,s} < X_{crit}), \tag{9}$$

where X_{crit} is the critical phosphorus threshold for the lake and $I(\ldots)$ is an indicator function returning 1 if the conditions are met and 0 otherwise. Exceeding X_{crit} results in irreversible eutrophication of the lake. Therefore, a value of $f_{reliability}$ near 1.0 is preferred, indicating the lake remains in an oligotrophic state during all time periods and across all SOWs. This formulation also specifies a hard constraint of $f_{reliability} > 0.85$.

Table 1
Baseline parameters for the lake problem with well-characterized uncertainty.

Name	Description	Baseline
b	Phosphorus removal rate	0.42
q	Phosphorus recycling rate	2
μ	Mean of natural inflows	0.02
σ	Standard deviation of natural inflows	0.0017
δ	Utility discount factors	0.98

A prior study of the lake problem (Singh et al., 2015) first considers the case where the parameters b, q, μ, σ and δ are assumed to be well-known. These well-characterized, baseline parameters are shown in Table 1. When a system is not well-characterized and model parameters not known to a degree of certainty, actionable decisions may be based on faulty assumptions. Under more realistic assumptions, the lake properties (b and q) and the natural pollution inflows (μ and σ) might be deeply uncertain. Additionally, the appropriate discount rate applied to the long-term future economic consumption is also an “unresolvable uncertainty” [Weitzman (1998), Newell and Pizer (2003)]. Singh et al. (2015) later considered deep uncertainty in their analysis, but limited the experiment to deeply uncertain natural pollution inflows (μ and σ). In this study, we extend this analysis and consider all five parameters under deep uncertainty (b, q, μ, σ and δ).

We stress that this problem is intentionally kept simple to both demonstrate OpenMORDM while also serving as a reproducible tutorial for interested readers. The intent is to demonstrate the analytic and visual tools provided by OpenMORDM on a conceptually simple problem, where the factors driving decision making are known *a priori* for verification and validation of these techniques and where users may have some initial intuition about the trade-offs. Even with its conceptually simple formulation, prior studies have demonstrated that the lake problem exhibits nonlinear dynamics illustrative of real-world applications, which pose significant challenges to modern decision support frameworks [Singh et al. (2015), Ward et al. (2015)].

4. Using OpenMORDM to analyze the lake problem

This section guides the reader through an example application of OpenMORDM to analyze the lake problem. This analysis is designed to demonstrate the use and capabilities of OpenMORDM. This section is styled as a tutorial, where the reader can follow these examples using the publicly-available OpenMORDM software. The supplemental material includes instructions for downloading and installing this software as well as example data files and scripts containing the results generated below. Readers replicating these results should note that these analyses are stochastic and observed results may differ slightly from the figures presented in this paper subject to their individual implementation choices.

4.1. Generating alternatives

We begin by loading the `OpenMORDM` and `lhs` (Latin hypercube sampling) packages:

```
library(OpenMORDM)
library(lhs)
```

Next, we construct the problem formulation for the lake problem under well-characterized uncertainty. Here, the lake problem is compiled into a standalone executable, `lake.exe`, configured with 100 decision variables, 4 objectives, and 1 constraint as detailed in Section 3:

```
nvars <- 100
nobjs <- 4
nconstrs <- 1

problem <- define.problem("./lake.exe", nvars, nobjs, nconstrs,
        bounds=matrix(rep(range(0, 0.1), nvars), nrow=2),
        names=c("Phosphorus in Lake", "Economic Benefit", "Inertia", "Reliability"),
        maximize=c("Economic Benefit", "Inertia", "Reliability"),
        epsilons=c(0.0001,0.0001,0.000001,0.000001))
```

This example adopts nomenclature for a Windows system, but Linux and Mac systems are also supported. Specifically, executable filenames on Mac and Linux typically do not include the ".exe" extension. Furthermore, Linux and Mac commands typically need "./" prepended to the command to run an executable. Additional information provided to OpenMORDM includes the decision variable bounds (each having a range of [0, 0.1]), the name of the four performance objectives, and whether an objective is maximized or minimized. By not specifying any arguments to the executable, we are using the default baseline parameters shown in Table 1. These default baseline parameters reflect the expected future SOW under well-characterized uncertainty.

Next, we generate the alternative designs for the lake problem under well-characterized uncertainty using the Borg MOEA [Hadka and Reed (2013)]. In the command below, we allow the optimization algorithm 100,000 function evaluations to find the Pareto approximate solutions:

```
data <- borg.optimize(problem, 100000)
```

Note that we are finding Pareto approximate solutions to discover the trade-offs that exist for a baseline of the best available estimate of the future SOW. The MORDM framework recommends establishing a baseline optimization to the projected future, because optimizing to a mixture of deeply uncertain future SOWs will implicitly sacrifice attainable system performance (i.e., Pareto efficiency) to maintain feasibility across the breadth of future possibilities in a manner that will be hidden from stakeholders. Establishing a baseline enables the effects of the deep uncertainties and alternative definitions of robustness to be explored explicitly to determine if the problem needs to be re-formulated and additional uncertainties included in the multi-objective optimization step. For the lake problem example, we illustrate this process by optimizing to a simulated best available estimate of the future SOW (i.e., assuming that natural phosphorus inputs are the only well-characterized uncertainty). Subsequently, alternative robustness measures are used to explore the potential impacts of a broader suite of deep uncertainties in the system.

The resulting baseline Pareto approximate set can be displayed in a three-dimensional (3D) scatter plot and a parallel coordinates plot to view the trade-offs between performance objectives. We also define a color palette for the graphical display of reliability that is arguably more intuitive than the default.

```
data <- borg.optimize(problem, 100000)
palette <- colorRampPalette(c("green", "yellow", "red"))(100)
mordm.plot(data, color="Reliability", palette=palette)
mordm.plot.parallel()
```

The result of these commands is shown in Fig. 3. In the 3D scatter plot displayed in Fig. 3(a), the three spatial axes display the phosphorus in the lake, economic benefit, and reliability and the glyphs (the spherical markers displayed in the scatter plot) are color-coded to represent the range of reliabilities, where green indicates 100% reliability and red indicates 86% reliability. Each point in the scatter plot represents a pollution control strategy (a set of decision variables a_t) that are approximately Pareto-optimal. Recall that since the problem formulation defines the constraint $f_{reliability} > 0.85$, no solutions with lower reliability are shown in the plot. This constraint threshold is built into the model (i.e., `lake.exe`). Each glyph in the plot represents the alternative lake pollution control strategies that compose the trade-offs facing decision makers. The parallel coordinates plot presented in Fig. 3(b) is an alternate visual representation of high-dimensional datasets, where each line represents an alternative lake pollution control strategy. The location where the line intersects each vertical axis designates the relative objective value. Note that the ideal point and ideal axis values are indicated by a star in both figures.

4.2. Negotiated selection of designs

The negotiated selection of solution(s) for further analysis or implementation requires diverse stakeholders to agree upon the levels of compromise between the strategies. The 3D scatter plot and parallel coordinates plot are useful for visualizing these trade-offs. Stakeholders can observe how performance objectives are affected, either positively or negatively, by their decisions. For instance, we observe in Fig. 3 that phosphorus in the lake is positively correlated with economic benefit. A stakeholder pursuing increased economic activity must accept that the pollution in the lake increases proportionally. The results show that phosphorus in the lake is inversely correlated with reliability. If a stakeholder is risk averse (i.e., they desire high reliability), then tighter pollution control strategies are required. Consequently, such restrictive pollution control strategies also dampen economic activity. This form of analysis has two advantages: (1) by exploring the effects of trade-offs, the stakeholders learn the dependencies between their decisions and system performance; and (2) by comparing and contrasting alternative designs, the stakeholders discover the cost-benefit compromise offered by various strategies and avoid myopic decisions.

There are several ways to explore these solutions in detail, ultimately identifying one or several designs of interest. If the stakeholder articulates preferences, one could use a weighting scheme to identify the most-preferred designs. If instead there are specific thresholds in performance required by the stakeholder, one can use "brushing" to eliminate all solutions that fail to meet those

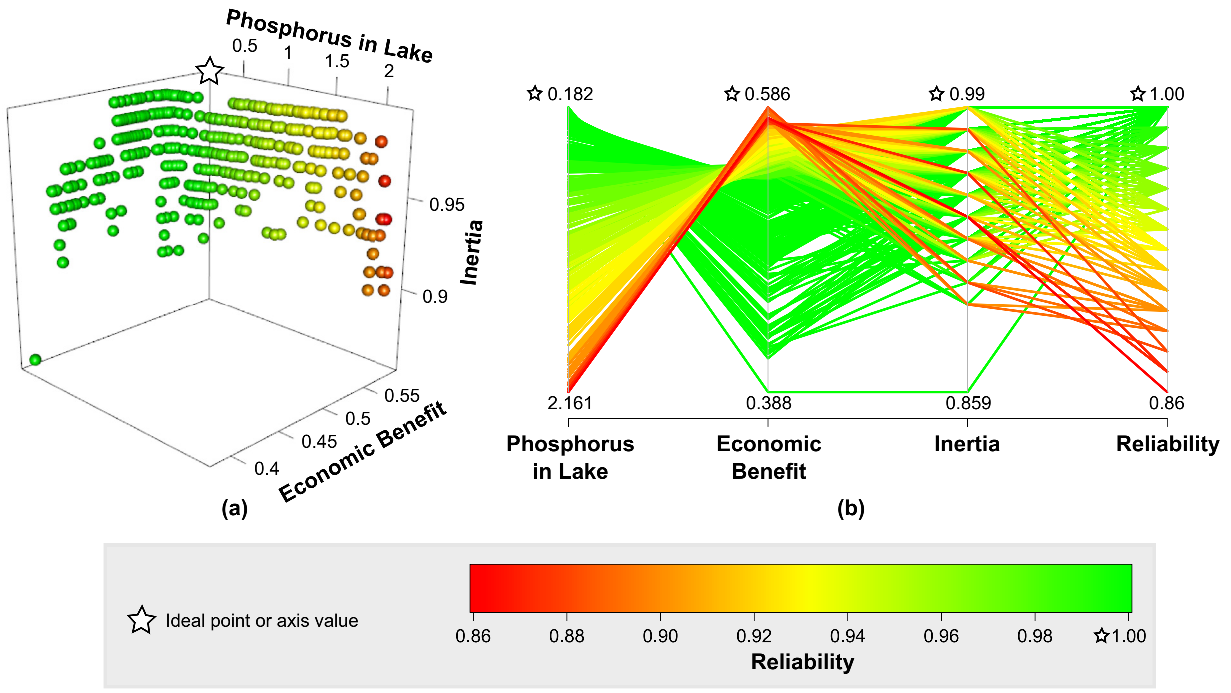

Fig. 3. Pareto approximate set attained while optimizing the four objective formulation assuming a single well-characterized distribution for the natural phosphorus inflows. Panel (a) shows a three dimensional scatter plot where the points represent the alternative lake pollution control strategies that compose the trade-offs facing decision makers. Panel (b) provides an alternative visualization of the lake pollution control trade-offs using a parallel coordinates plot. Each line represents an alternative lake pollution control strategy where its intersections on the vertical axes designate the relative objective values. The change in color from green to red depicts the trade-off in reliability against the three other objectives. The ideal point or axis value for each objective is indicated by a star. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

thresholds [Kollat and Reed (2007)]. OpenMORDM supports both methods. One easy way to access these controls is through the web-based visualization toolkit, which provides slider widgets for specifying the preferences or brushing limits (see the documentation for the `explore` command). Alternatively, preferences can be specified through R commands:

```
weights <- c(0.15, 0.325, 0.2, 0.325)
preference <- mordm.weight(data, weights)
mordm.plot(data, color=preference, selection=which.max(preference))
```

The result of these commands is shown in Fig. 4. Solutions are now color-coded by the preference weighting with green (in the web version) indicating the most preferred designs. In this example, the stakeholder prefers maximizing economic benefit and reliability (both with a weight of 0.325) with the other two performance objectives having less significance. Based on these weights, the preferred design can be identified and displayed in the plot as the enlarged glyph. It should be noted that this weighted preference is imposed *a posteriori* after visualizing the full set of Pareto-approximate alternatives; methods such as Multi-Criteria Decision Making (MCDM) [Triantaphyllou (2000)] requiring *a priori* preference weighting would produce only a single solution, not permitting an interactive decision process with evolving stakeholder preferences and increase the risk of myopic analysis.

In many cases, it is also desirable to see the decision variables (i.e., control policies) present in select designs. The underlying data for each design can be extracted with the following command:

```
mordm.get.set(data)
```

or specific decision variables can be selected with:

```
mordm.get.set(data)[,c("Var1", "Var2")]
```

In this example, OpenMORDM assigns default variable names of the form “Var1”, “Var2”, and so on, but user-defined names can also be assigned in the `define.problem` method. This data, which is stored as an R matrix, can subsequently be plotted by standard R functions.

4.3. *Defining uncertainties and evaluating robustness*

This simple first pass of the analysis assumes that the uncertainties are well characterized. In contrast, the uncertainties surrounding real-world environmental systems are often poorly characterized. The value of certain model inputs may be unknown or only known to lie within a range of values. Or the probability distributions representing the uncertainty of key parameters may be deeply uncertain. For example, testing the assumptions about the extreme values of the distribution of the natural inputs of phosphorus into the lake requires long observations that may not be available. As another example, consider the task of projecting the discount rate on a century time scale (e.g., Newell and Pizer (2003)). In both cases, adopting a probability density function is a deeply uncertain choice.

As discussed in Section 2, MORDM is a decision support framework for identifying the vulnerabilities, dependencies, and trade-offs in systems with deep uncertainties. Table 2 shows the

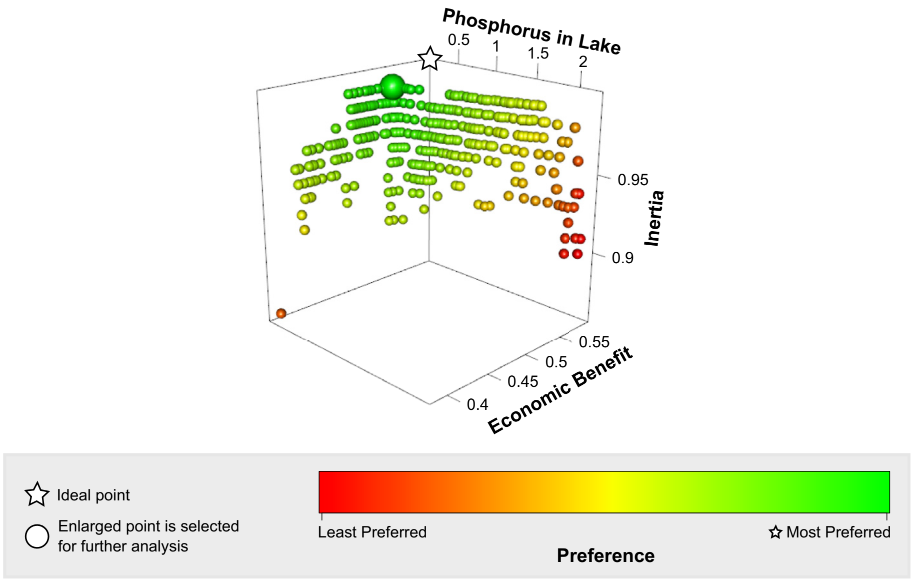

Fig. 4. The Pareto approximate set attained under well-characterized uncertainty color-coded by the stakeholder's preferences. In this hypothetical example, the stakeholder prefers maximizing economic benefit and reliability. The enlarged glyph indicates the most preferred design. The star indicates the ideal point, which is formed by taking the best value in each objective. Due to objective conflicts, the ideal point may not be attainable.

uncertain parameters now modeled under deep uncertainty. Since we are deeply uncertain of the values for these parameters, we select a range of plausible values for each uncertainty from which we sample using a Latin hypercube strategy. Below, we create 1000 randomly-generated Latin hypercube samples.

After these two commands are invoked, `models` stores a list of problem formulations, each of which models one uncertainty parameterization from `factors`.

Finally, we subject the designs discovered previously under well-characterized uncertainty to the deeply-uncertain parameterizations. The performance of designs across SOWs can be used to compute robustness measures defined in Section 2. These regret-based and satisficing-based measures emphasize different goals (e.g., to minimize deviation from performance in the expected future, or to maximize the number of plausible futures in which some criteria are met). The robustness measure may change the design recommendation, so this choice must align with stakeholder preferences [Herman et al. (2015)]. All four of these robustness measures are provided by OpenMORDM. For the satisficing measures, we define the satisficing function with two criteria: (1) satisfy the reliability constraint such that the sum of

```
nsamples <- 1000
param_names <- c("b", "q", "mean", "sigma", "delta")
param_bounds <- matrix(c(
        c(0.1, 0.45),
        c(2, 4.5),
        c(0.01, 0.05),
        c(0.001, 0.005),
        c(0.93, 0.99)), nrow=2)
baseline_factors <- c(0.42, 2, 0.02, 0.0017, 0.98)

factors <- randomLHS(nsamples, length(param_names))
factors <- t(apply(factors, 1, function(x) x*diff(param_bounds)+param_bounds[1,]))
colnames(factors) <- param_names
```

The output from `randomLHS` is a 1000×5 matrix (1000 rows and 5 columns) with values on the range $[0, 1]$. These values need to be scaled to the sample range for each uncertain parameter shown in Table 2. The next line, invoking `t(apply (...))`, applies the correct scaling to each row in the matrix. The result of `apply (...)` is transposed, so we use the transpose operator, `t`, to convert the matrix back into the appropriate form. For convenience, we also assign the column names to `factors`, which will appear as labels in any plots generated by OpenMORDM.

Next, for each uncertainty parameterization, we define a new problem formulation by passing the parameters to the executable as command-line arguments.

```
model_calls <- do.call(sprintf, c(list("./lake.exe -b %f -q %f -m %f -s %f -d %f"),
        lapply(1:ncol(factors), function(i) factors[,i])))

models <- lapply(model_calls, function(command) {
        define.problem(command, nvars, nobjs, nconstrs,
                bounds=matrix(rep(range(0, 0.1), nvars), nrow=2),
                names=c("Phosphorus in Lake", "Economic Benefit", "Inertia",
                        "Reliability"),
                maximize=c("Economic Benefit", "Inertia", "Reliability"))
})
```

constraint values is 0; and (2) preserve a minimum economic benefit of 0.1.

```
uncertainty.samples <- mordm.sample.uncertainties(data, nsamples, models)

robustness <- mordm.evaluate.uncertainties(uncertainty.samples,
                function(x) sum(abs(x$constrs)) == 0 && x$objs[2]>0.1,
                factors,
                baseline_factors)
```

Once the calculations to evaluate the selected robustness measures are complete, we can then update the 3D scatter plot to color-code each glyph with the robustness measure:

```
mordm.plot(data, color=robustness[,"Regret Type I"])
mordm.plot(data, color=robustness[,"Regret Type II"])
mordm.plot(data, color=robustness[,"Satisficing Type I"])
mordm.plot(data, color=robustness[,"Satisficing Type II"], clim=(-1, 0))
```

Fig. 5 shows the result of these commands. Each panel in the figure, labeled (a)-(d), depicts the glyphs colored by the respective robustness measure. Glyphs colored green indicate relatively strong robustness while red glyphs indicate poor robustness. Fig. 5 illustrates several key arguments, as discussed below.

First, the two regret-based measures in panels (a) and (b) identify opposite regions with strong and poor robustness. Solutions with lower phosphorus in the lake have poor robustness under Regret Type I but have superior robustness under Regret Type II. Recall that Regret Type I measures the performance deviation from the baseline SOW whereas Regret Type II is the deviation from the best performance within the current SOW. Regret Type I is a weaker indicator of robustness because we expect solutions with more restrictive pollution control strategies (i.e., less phosphorus in the lake) to be more robust. It is instead identifying the solutions with more “room to fail”. Solutions with high levels of phosphorus in the lake in the baseline SOW are already struggling, resulting in less impact from the addition of deep uncertainties. Regret Type II, which measures the performance deviation from the best performance within the current SOW, identifies that solutions with more restrictive pollution control strategies perform better across all SOWs.

Second, Regret Type II in panel (b) and Satisficing Type I in panel (c) show similar regions with strong robustness. The left-most region corresponding to more restrictive pollution control strategies offers superior robustness. This result is interesting since Regret Type II and Satisficing Type I measure different criteria. Regret Type II is based on the deviation in performance, whereas Satisficing Type I measures the fraction of SOWs that satisfy the user's criteria. Observing similarities between these two robustness measures is consistent with the hypothesis that restrictive pollution control strategies are more robust to deep uncertainties.

Third, we observe that the results in panels (a)-(c) all show that the ideal design identified earlier in Fig. 4 exhibits inferior robustness. Had we assumed well-characterized uncertainties, then the town would potentially implement a risky pollution control strategy that has a higher chance of polluting the lake. By conducting a broader experiment and exploring the deeply uncertain states, we can suggest solutions that are resilient to deeply uncertain futures. Beyond this simple illustrative example, MORDM is a constructive learning process where it would be expected that concerns raised in the robustness analysis may require the generation and exploration of new multi-objective problem formulations with new decisions, changes in the modeled uncertainties, and potentially modified objectives or constraints [Kasprzyk et al. (2012), Kasprzyk et al. (2013)].

One additional advantage of this analysis is the ability to quantify changes in robustness with performance trade-offs. For example, there is a trade-off between phosphorus in the lake and economic benefit. Figures (a)-(c) allows the stakeholders to assess degradation to any objectives sacrificed to achieve stronger robustness. Robustness Type I in panel (a) suggests severe restrictions on economic activity are required. Robustness Type II and Satisficing Type II in panels (b) and (c), respectively, are more relaxed, allowing higher phosphorus levels. As demonstrated above, the choice of robustness measure can impact the decision making process. OpenMORDM provides tools to experiment with different robustness criteria. Understanding how different robustness criteria interact can help to gain a deeper understanding of the ramifications of a decision analysis.

In Fig. 5, the Satisficing Type II result in panel (d) shows all designs exhibit poor robustness (i.e., all solutions colored red). Recall that Satisficing Type II measures the maximum allowable uncertainty as a distance from the expected future SOW while still satisfying the criteria (see equation (4) in section 2.2). In this case, the SOW causing failure of all solutions is $b = 0.32$, $q = 2.04$, $\mu = 0.017$, $\sigma = 0.0016$, and $\delta = 0.98$. This SOW unveils a severe vulnerability causing all solutions to pollute the lake. This severe vulnerability was not detected by the other robustness measures considered in this study since their averaging tends to hide outliers. Again, this demonstrates the need to consider multiple robustness measures in an analysis. Such a result is an indication that further investigation of the system vulnerabilities is required.

Table 2
Lower and upper bounds of deeply-uncertain parameters.

Name	Description	Low	High
b	Phosphorus removal rate	0.1	0.45
q	Phosphorus recycling rate	2	4.5
μ	Mean of natural inflows	0.01	0.05
σ	Standard deviation of natural inflows	0.001	0.005
δ	Utility discount factors	0.93	0.99

4.4. Vulnerability analysis

To narrow the scope of the vulnerability analysis, we select a single solution for further analysis. Nevertheless, it is common in practice to explore several alternative designs at this stage. Clicking on a point using the middle mouse button within the 3D scatter plot of OpenMORDM prints the id (row number) of the selected point. On computers lacking a middle mouse button, run the command `mordm.identify (button = ?)` replacing `?` with `1` to enable clicking with the left button or `2` for the right mouse button.

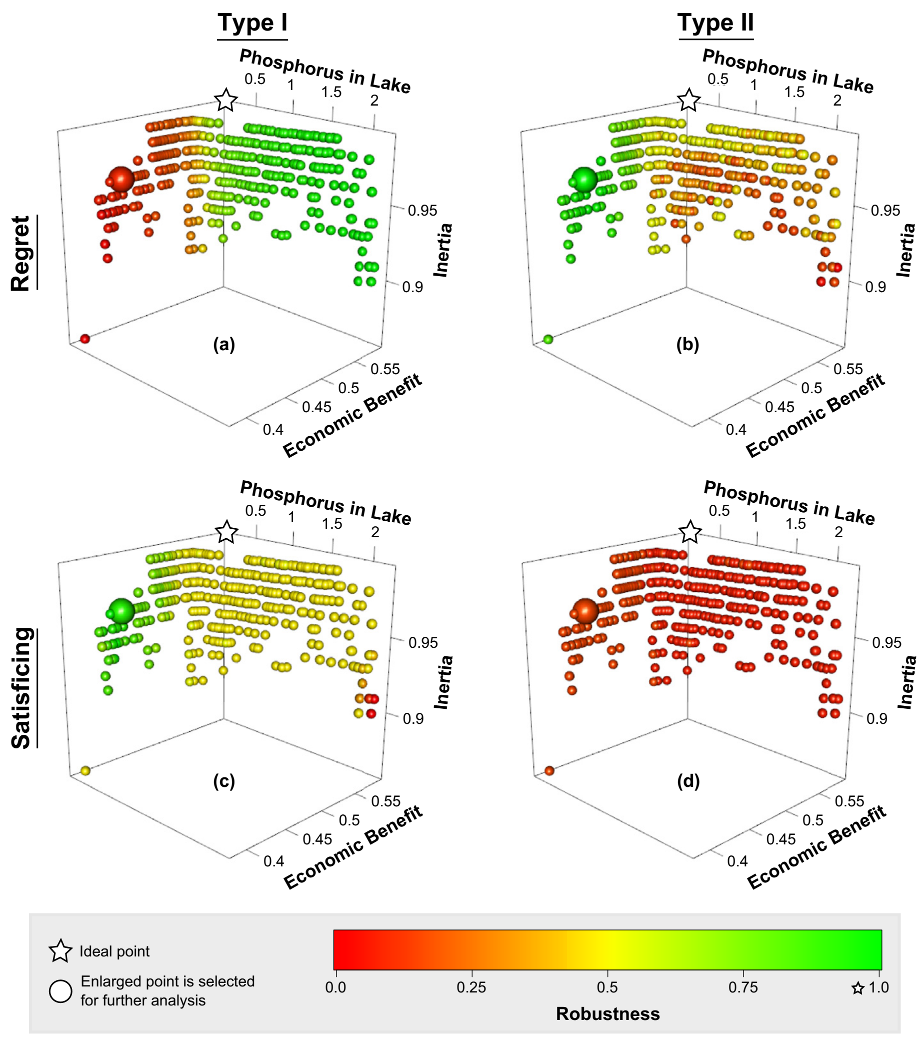

Fig. 5. The original Pareto approximate designs re-evaluated under deep uncertainty using the four regret and satisficing robustness metrics. Panels (a)–(c) highlight different regions with poor robustness, but consistently identify the left-most points, with stricter pollution control strategies, as exhibiting superior robustness (colored green). Panel (d) indicates all designs fail the Info-Gap criterion, suggesting the existence of at least one SOW triggering failure in all designs. Based on its consistent demonstration of robustness in panels (a)–(c), the enlarged point is selected for further analysis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In this example, we have selected the point in Fig. 5 as indicated by the enlarged glyph. This point was selected based on the strong robustness performance shown in panels (a)-(c) of Fig. 5. For our particular dataset, the selected point is #274:

```
selected.point <- uncertainty.samples[[274]]
```

The poor robustness indicated by Satisficing Type II suggest there exists key vulnerabilities leading to failures — the user-specified criteria is not satisfied under all future SOWs — but detailed analysis using MORDM can reveal the factors causing these vulnerabilities. Two methods for exploring these vulnerabilities are PRIM and CART, as introduced in Section 2. PRIM is the method often used in RDM and MORDM (see, for example, Lempert et al. (2008), Kasprzyk et al. (2013), and Herman et al. (2014)). Both PRIM and CART support *a posteriori* scenario discovery [Lempert et al. (2008)] to identify the combinations of deeply uncertain factors to which a system is vulnerable. Here we use the ease of switching operators within the OpenMORDM toolbox to compare and contrast the efficacy of PRIM and CART in extracting salient vulnerabilities for the lake problem.

Both PRIM and CART succeed in identifying the key drivers of vulnerabilities. Fig. 6 shows the vulnerability results using PRIM resulting from the following command:

```
mordm.prim(factors, selected.point$objs[,"Reliability"],
    threshold.type=-1)
```

In this example, we are defining vulnerable states as those with low reliability. Recall that reliability measures the fraction of years that the phosphorus level in the lake remains below the critical threshold, X_{crit}. The argument `threshold.type = -1` specifies that the PRIM algorithm identifies ranges of the deeply uncertain parameters leading to below average reliability (by default the threshold is the average response value, a specific threshold can be defined using the `threshold = t` argument). Each of the vertical bars in Fig. 6 corresponds to one of the uncertainty parameterizations in Table 2. The red overlaid boxes represent the ranges of each parameter that lead to vulnerabilities. If all uncertainty parameters fall within the red boxes, then there is a greater chance that the lake will fail to satisfy the stakeholders' criteria. Fig. 6 illustrates the bounds for each deeply uncertain factor that are of concern. For this

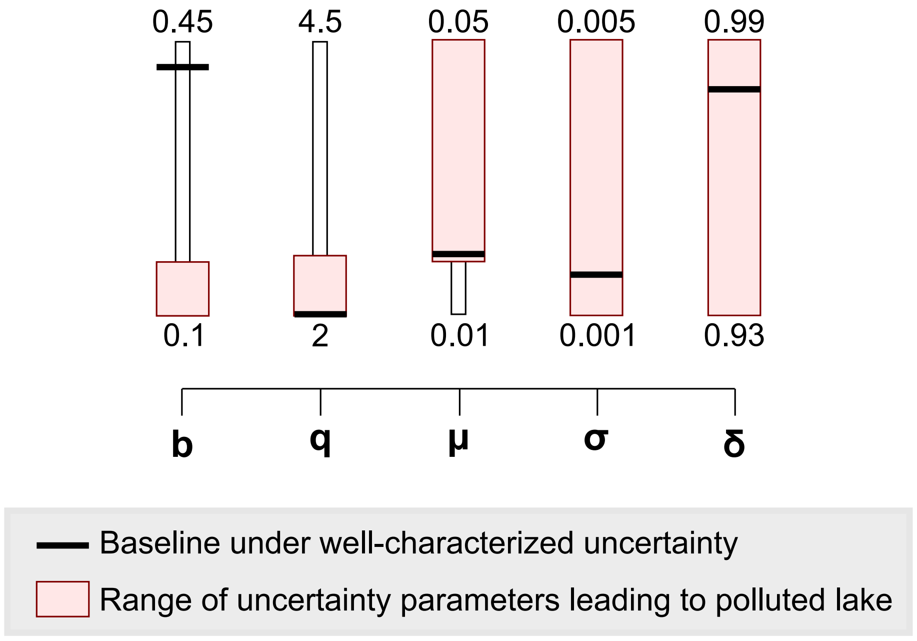

Fig. 6. The Patient Rule Induction Method (PRIM) is an association rule learning algorithm that identifies ranges of the uncertainties which lead to a polluted lake with higher probability. If all uncertainties fall within the red boxes, then the lake is at higher risk of becoming polluted. The black horizontal bars indicate the baseline parameters under well-characterized uncertainty. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

example, low values of b and q and moderate to large values of μ are identified is critical factors. Note that the red boxes covering the entire range of σ and δ in Fig. 6 indicates that these two factors are not significant with respect to vulnerabilities — any value of σ or δ will lead to vulnerabilities as long as the uncertainties fall within the critical factor's bounds (b, q, and μ).

PRIM enables the user to influence the generated boxes by specifying the desired coverage, the percentage of all vulnerable states contained within the box, and density, the fraction of states within the box that are vulnerable. In the example shown in Fig. 6, the coverage is 76% and density is 33%. The `prim` package, used in this study, is not interactive but can be controlled using optional

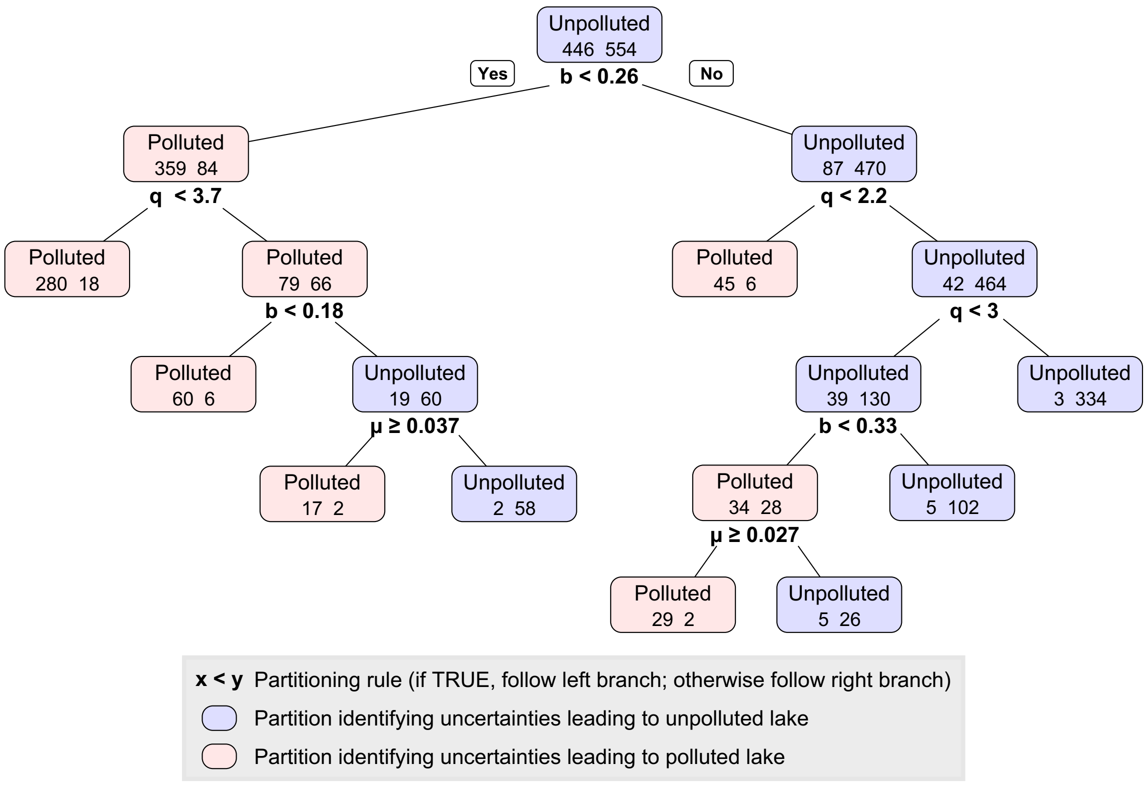

Fig. 7. The Classification and Regression Trees (CART) method is an association rule learning algorithm that recursively partitions the uncertainty space to minimize type 1 error (false positives). The two numbers shown in each partition indicate the number of polluted SOWs (left) and unpolluted SOWs (right) contained within the partition.

arguments. In this study, we kept the default arguments. Alternatively, the `sdtoolkit` PRIM implementation allows the user to interactively see the tradeoff between coverage and density and select a PRIM box with the desired attributes. Due to this tension between coverage and density, the user must often sacrifice density to gain better coverage, or reduce coverage to improve density. This increases the burden of using PRIM since the user must choose the appropriate tradeoff between coverage and density.

Similarly, Fig. 7 quantifies how CART identifies key drivers of vulnerabilities. Fig. 7 is generated by the following command.

```
analyze.cart(factors,
    ifelse(selected.point$objs[,"Reliability"]<1, "Polluted", "Unpolluted"))
```

In this example, we partition solutions into two classes: `Polluted` and `Unpolluted`. A solution is `Unpolluted` if the lake has 100% reliability, meaning the phosphorus level remains below the critical threshold, X_{crit}, in all SOWs. Otherwise, the lake is classified as `Polluted`, meaning the lake becomes eutrophic in at least one SOW. As shown in Fig. 7, CART produces a decision tree where every node is a recursive partition of the uncertainty space. Below each node is a rule, such as $b < 0.26$, that further divides the partition. Each partition is color-coded either red or blue (in the web version) if the partition identifies polluted (vulnerable) or unpolluted states, respectively. CART recursively partitions the space to minimize Type 1 errors. The depth of each uncertainty parameter in the tree (and note that an uncertainty can appear multiple times) is informative of the significance of each uncertainty in classifying vulnerabilities. Uncertainties nearer to the root offer more predictive power in identifying vulnerabilities.

Both PRIM and CART are consistent in the key vulnerabilities identified. From Figs. 6 and 7, we learn that b and q are the most critical uncertainties. In order to avoid vulnerable states, small values of b and q should be avoided. Since b and q are uncontrollable properties of the lake in our problem formulation, these observations suggest *ex post* monitoring recommendations for the town's inhabitants. The optimal pollution control strategies devised in this exercise are satisfactory for most conditions, except for when b and q are small. μ, the mean of natural pollution inflows, has a less significant effect. σ and δ have no significant effect in this case, indicated in PRIM by a hyperbox spanning the entire range of uncertainties and in CART by the parameter not appearing in any partitioning rule.

Since CART generates a decision tree, we can identify partitions that provide a desired misclassification rate and derive specific *ex post* monitoring recommendations. For example, following the entire right-branch of the tree, if the town's inhabitants can monitor the system to ensure that $b \geq 0.26$ *and* $q \geq 2.2$, then only 8% of the remaining SOWs would be vulnerable to failure. Further restrictions can reduce the failure rate. Ensuring $b \geq 0.33$ *and* $q \geq 3.0$ results in a 2% failure rate. If the system properties violate these monitoring recommendations, then further MORDM analysis and problem reformulations may be necessary to discover satisfactory designs. Furthermore, identifying the key drivers of vulnerabilities advises stakeholders where further analysis may be beneficial (i.e., conducting experiments to provide better estimates of b and q). Such monitoring recommendations are similar to the "adaptation tipping points" of dynamic adaptive policy pathways [Haasnoot et al. (2013), Kwadijk et al. (2010)], triggering a shift to alternative policies better suited for the realized SOW, a concept also developed by the Adaptive Robust Design approach [Hamarat et al. (2013)].

Our analysis illustrates one limitation of PRIM. Recall from Fig. 5 that the Satisficing Type II measure indicated the existence of at least one uncertainty parameterization near the expected future state that caused all designs to fail. On further inspection of the data, there was a single parameterization causing all designs to fail with $b = 0.32$ *and* $q = 2.04$. This parameterization is correctly classified as 'Polluted' by CART, as shown in Fig. 7. However, PRIM fails to identify this uncertainty parameterization as vulnerable — it falls outside the hyperbox indicated in Fig. 6. For this particular application, CART has demonstrated stronger predictive power than PRIM, possibly due to CART's ability to minimize Type 1 classification error (false positives) by creating disjoint partitions of the data. On applications with more uncertain factors, this could lead to complex partitions that are difficult to interpret. PRIM aids interpretability by identifying a single contiguous region based on the desired coverage and density. Given these observations, it seems advisable to consider both PRIM and CART for the scenario discovery to assess whether their use is appropriate. These results expand on the analysis of Lempert et al. (2008) to highlight why the methodological flexibility facilitated by OpenMORDM improves the power, transparency, and value of bottom-up decision support frameworks under deep uncertainty.

5. Conclusion

We present the design of OpenMORDM, an open-source implementation of MORDM in the R programming language. In addition, we demonstrate the application of OpenMORDM on a hypothetical environmental management problem, called the "lake problem". Using OpenMORDM, the decision maker can rapidly interrogate a model to identify optimal designs under well-characterized uncertainty, explore the trade-offs, dependencies, and vulnerabilities revealed under deep uncertainty, and develop design or *ex post* monitoring recommendations based on the identified vulnerabilities. Moreover, since OpenMORDM is open-source, users can build upon the "tower of R", which has been built and refined over the past two decades to become one of the most prominent statistical programming languages for scientific computing.

While OpenMORDM is designed to support MORDM analyses, we observe many similarities between the so-called "bottom-up" decision support frameworks of Decision Scaling, Info-Gap, RDM, and MORDM. Future work entails developing a comprehensive toolbox for robust decision support that enables access to all decision support frameworks in a unified software API. Furthermore, one challenging aspect is defining the actual computer model that describes the problem and coupling it to the analytical tools. We have addressed this by leveraging an existing API from the MOEA Framework [Hadka (2014)] for connecting analytical tools to computer models, but this requires some level of programming to implement. Wu et al. (2014) have also developed a coupling tool using file input/output. Finally, it is important to note that current and future parallel computing capabilities have transformative value for expanding the size and scope of OpenMORDM applications, especially for the multi-objective search and exploratory sampling of deeply uncertain SOWs. The ultimate aim is to provide decision makers with unrestricted access to a comprehensive suite of extensible tools that facilitate learning and discovery under deep uncertainty.

Acknowledgments

This work was partially supported by the National Science Foundation (GEO-1240507) through the Network for Sustainable Climate Risk Management (SCRiM) under NSF cooperative agreement GEO-1240507 and the Penn State Center for Climate Risk Management. Icons in Fig. 1 provided by the Nuvola Icon Theme by David Vignoni and are available under the GNU Lesser General Public License, version 2.1. The authors gratefully acknowledge Riddhi Singh and Tori Ward for providing source code for the lake problem and Elmar Plischke for giving permission to translate his Matlab sensitivity analysis codes into R. We also thank Gregory Garner, Xuan Yi, Yu Li, Tori Ward, Xihao Deng, Perry Oddo, Bernardo Tindade, Julie Quinn, and the three anonymous reviewers for testing and providing feedback on OpenMORDM.

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.envsoft.2015.07.014.

References

Bankes, S., 1993. Exploratory modeling for policy analysis. Oper. Res. 41, 435–449.

Ben-Haim, Y., 2004. Uncertainty, probability and information-gaps. Reliab. Eng. Syst. Saf. 85 (1–3), 249–266.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth.

Brill, E.D., Flach, J., Hopkins, L., Ranjithan, S., 1990. MGA: a decision support system for complex, incompletely defined problems. IEEE Trans. Syst. Man, Cybern. 20 (4), 745–757.

Brown, C., 2010. Decision-scaling for Robust Planning and Policy Under Climate Uncertainty. World Resources Report 2010-2011 Uncertainty Series.

Brown, C., Ghile, Y., Laverty, M., Li, K., 2012. Decision scaling: linking bottom-up vulnerability analysis with climate projections in the water sector. Water Resour. Res. 48, W09537.

Bryant, B.P., 2014. Sdtoolkit: Scenario Discovery Tools to Support Robust Decision Making. R package version 2.33-1. http://CRAN.R-project.org/package=sdtoolkit.

Bryant, B.P., Lempert, R.J., 2010. Thinking inside the box: a participatory, computer-assisted approach to scenario discovery. Technol. Forecast. Soc. Change 77 (1), 34–49.

Cahon, S., Melab, N., Talbi, E.-G., 2004. ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics. J. Heuristics 10 (3), 357–380.

Carpenter, S.R., Ludwig, D., Brock, W.A., 1999. Management of eutrophication of lakes subject to potentially irreversible change. Ecol. Appl. 9 (751), 771.

Coello Coello, C., Lamont, G.B., van Veldhuizen, D.A., 2007. Evolutionary Algorithms for Solving Multi-objective Problems, second ed. Springer-Verlag New York, Inc.

Cohon, J., Marks, D., 1975. A review and evaluation of multiobjective programming techniques. Water Resour. Res. 11 (2), 208–220.

Conway, J., Eddelbuettel, D., Nishiyama, T., Prayaga, S.K., Tiffin, N., 2013. RPostgreSQL: R Interface to the PostgreSQL Database System. R package version 0.4. http://CRAN.R-project.org/package=RPostgreSQL.

D'Ervau, E.L., 2013. Optimizing Early-warning Monitoring Systems for Improved Drinking Water Resource Protection (Master's thesis). Universität Stuttgart, 8 October.

Duong, T., 2014. Prim: Patient Rule Induction Method (PRIM). R package version 1.0.14. http://CRAN.R-project.org/package=prim.

Durillo, J.J., Nebro, A.J., Alba, E., 2010. The jMetal framework for multi-objective optimization: design and architecture. In: 2010 IEEE Congress on Evolutionary Computation, 4138-4325, 18–23 July.

Friedman, J.H., Fisher, N.I., 1999. Bump-hunting for high dimensional data. Stat. Comput. 9, 123–143.

Ghile, Y., Taner, M., Brown, C., Grijsen, J., Talbi, A., 2014. Bottom-up climate risk assessment of infrastructure investment in the Niger river Basin. Clim. Change 122, 97–110.

Giuliani, M., Herman, J., Castelletti, A., Reed, P., 2014. Many-objective reservoir policy identification and refinement to reduce policy inertia and myopia in water management. Water Resour. Res. 50 (4), 3355–3377.

Groves, D.G., Lempert, R.J., 2007. A new analytic method for finding policy-relevant scenarios. Glob. Environ. Change 17, 73–85.

Haasnoot, M., Kwakkel, J.H., Walker, W.E., ter Matt, J., 2013. Dynamic adaptive policy pathways: a method for crafting robust decisions for a deeply uncertain world. Glob. Environ. Change 23, 485–498.

Hadka, D., Reed, P.M., Simpson, T.W., 2012. Diagnostic assessment of the borg MOEA on many-objective product family design problems. WCCI 2012 world congress on computational intelligence. Congr. Evol. Comput. 21 (2), 231–259.

Hadka, D., Reed, P.M., 2012. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. Evol. Comput. 20 (3), 423–452.

Hadka, D., Reed, P.M., 2013. Borg: an auto-adaptive many-objective evolutionary computing framework. Evol. Comput. 21 (2), 231–259.

Hadka, D., 2014. MOEA Framework: a Free and Open Source Java Framework for Multiobjective Optimization. User Manual. Version 2.4. http://www.moeaframework.org/.

Hadka, D., Reed, P.M., 2015. Large-scale parallelization of the Borg MOEA for many-objective optimization of complex environmental systems. Environ. Model. Softw. v69, 353–369.

Hall, J.W., Lempert, R.J., Keller, K., Hackbarth, A., Mijere, C., McInerney, D.J., 2012. Robust climate policies under uncertainty: a comparison of robust decision making and info-gap methods. Risk Anal. 32 (10), 1657–1672.

Hamarat, C., Kwakkel, J.H., Pruyt, E., 2013. Adaptive robust design under deep uncertainty. Technol. Forecast. Soc. Change 80 (3), 408–418.

Herman, J.D., Reed, P.M., Zeff, H.B., Characklis, G.W., 2015. How should robustness be defined for water systems planning under change? J. Water Resour. Manag. 04015012. http://dx.doi.org/10.1061/(ASCE)WR.1943-5452.0000509.

Herman, J.D., Zeff, H.B., Reed, P.M., Characklis, G.W., 2014. Beyond optimality: multistakeholder robustness tradeoffs for regional water portfolio planning under deep uncertainty. Water Resour. Res. 50 http://dx.doi.org/10.1002/2014WR015338.

Hine, D., Hall, J.W., 2010. Information gap analysis of flood model uncertainties and regional frequency analysis. Water Resour. Res. 46, W01514.

Hipel, K.W., Ben-Haim, Y., 1999. Decision making in an uncertain world: information-gap modeling in water resources management. IEEE Trans. Syst. Man, Cybern. – C Appl. Rev. 29 (4), 506–517.

Kasprzyk, J.R., Reed, P.M., Kirsch, B.R., Characklis, G.W., 2012. Many-Objective de novo water Supply portfolio planning under deep uncertainty. Environ. Model. Softw. 34, 87–104.

Kasprzyk, J.R., Nataraj, S., Reed, P.M., Lempert, R.J., 2013. Many objective robust decision making for complex environmental systems undergoing change. Environ. Model. Softw. 42, 55–71.

Knight, F.H., 1921. Risk, Uncertainty and Profit. Hart, Schaffner and Marx, N.Y.

Kollat, J.B., Reed, P.M., 2007. A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (VIDEO). Environ. Model. Softw. 22, 1691–1704.

Korteling, B., Dessai, S., Kapelan, Z., 2013. Using information-gap decision theory for water resources planning under severe uncertainty. Water Resour. Manag. 27, 1149–1172.

Kwadijk, J.C.J., Haasnoot, M., Mulder, J.P.M., Hoogvliet, M.M.C., Jeuken, A.B.M., van der Krogt, R.A.A., van Oostrom, N.G.C., Schelfhout, H.A., van Velzen, E.H., van Waveren, H., de Wit, M.J.M., 2010. Using adaptation tipping points to prepare for climate change and sea level rise: a case study in the Netherlands. WIREs Clim. Change 1, 729–740.

Kwakkel, J.H., 2015. Exploratory modelling and analysis (EMA) workbench. http://simulation.tbm.tudelft.nl/ema-workbench/contents.html (accessed 27.04.15.).

Kwakkel, J.H., Pruyt, E., 2013. Exploratory modeling and Analysis, an approach for model-based foresight under deep uncertainty. Technol. Forecast. Soc. Change 80, 419–431.

Lempert, R.J., 2002. A new decision Sciences for complex systems. Proc. Natl. Acad. Sci. 99, 7309–7313.

Lempert, R.J., Collins, M., 2007. Managing the risk of an uncertain threshold response: comparison of robust, optimum, and precautionary approaches. Risk Anal. 27 (4), 1009–1026.

Lempert, R.J., Sriver, R., Keller, K., 2012. Characterizing Uncertain Sea Level Rise Projections to Support Infrastructure Investment Decisions. California Energy Commission. Publication Number CEC-500-2012-056.

Lempert, R.J., Popper, S.W., Bankes, S.C., 2013. Shaping the Next One Hundred Years: New Methods for Quantitative. Long-Term Policy Analyses, RAND, Santa Monica, CA.

Lempert, R.J., Groves, D.G., Popper, S.W., Bankes, S.C., 2006. A general, analytic method for generating robust strategies and narrative scenarios. Manag. Sci. 52 (4), 514–528.

Lempert, R.J., Bryant, B.P., Bankes, S.C., 2008. Comparing Algorithms for Scenario Discovery. RAND Corporation. Working Paper WR-557-NSF.

Lownsbery, K.E., 2014. Quantifying the Impacts of Future Uncertainties on the Apalachicola-Chattahoochee-Flint Basin (Masters Project). Department of Civil and Environmental Engineering, University of Massachusetts - Amherst.

Maass, A., Hufschmidt, M., Dorfman, R., Thomas, H., Marglin, S., Fair, G., 1962. Design of Water-resource Systems: New Techniques for Relating Economic Objectives, Engineering Analysis, and Governmental Planning. Harvard University Press, Cambridge, MA.

Matrosov, E.S., Woods, A.M., Harou, J.J., 2013. Robust decision making and info-gap decision theory for water resource system planning. J. Hydrol. 494, 43–58.

Moody, P., Brown, C., 2013. Robustness indicators for evaluation under climate change: application to the upper Great Lakes. Water Resour. Res. 49, W20228.

Nazemi, A., Wheater, H.S., 2014. Assessing the vulnerability of water supply to changing streamflow conditions. EOS Trans. Am. Geophys. Union 95 (32), 288.

Newell, R.G., Pizer, W.A., 2003. Discounting the distant future: how much do uncertain rates increase valuations? J. Environ. Econ. Manag. 46, 52–71.

Nicklow, J., Reed, P.M., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., Karamouz, M., Minsker, B., Ostfeld, A., Singh, A., Zechman, E., 2010. State of the art for genetic algorithms and beyond in water resources planning and management. J. Water Resour. Plan. Manag. 136 (4), 412–432.

Olson, R., Sriver, R., Goes, M., Urban, N.M., Matthews, H.D., Haran, M., Keller, K., 2012. A climate sensitivity estimate using global average observations and an earth system model with a dynamic 3D ocean. J. Geophys. Res. Atmos. 117, D04103. http://dx.doi.org/10.1029/2011JD016620.

Pareto, V., 1896. Cours d'Économie Politique. Droz, Geneva.

Plischke, E., Borgonovo, E., Smith, C.L., 2013. Global sensitivity measures from given data. Eur. J. Oper. Res. 226 (3), 536–550.

Pujol, G., Iooss, B., Janon, A., Lemaitre, P., Gilguin, L., Gratiet, L.L., touati, T., Ramos, B., Fruth, J., Veiga, S.D., 2014. Package 'sensitivity': Sensitivity Analysis. R package version 1.9. http://CRAN.R-project.org/package=sensitivity.

R Development Core Team, 2008. R: a Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN: 3-900051-08-0. http://www.R-project.org/.

Reed, P.M., Hadka, D., Herman, J., Kasprzyk, J., Kollat, J., 2013. Evolutionary multiobjective optimization in water resources: the past, present, and future. Adv. Water Resour. 51, 438–456.

Roy, B., 1999. Decision-aiding Today: What Should We Expect? Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications. Springer, pp. 1–35.

RStudio, Inc, 2014. Shiny: Web Application Framework for R. R Package Version 0.10.1. http://CRAN.R-project.org/package=shiny.

Saltelli, A., Chan, K., Scott, E., 2008. Sensitivity Analysis. Wiley, New York, NY.

Savage, L.J., 1951. The theory of statistical decision. J. Am. Stat. Assoc. 46 (253), 55–67.

Schneller, G., Sphicas, G., 1983. Decision making under uncertainty: Starr's domain criterion. Theory Decis. 15 (4), 321–336.

Simon, H.A., 1959. Theories of decision-making in economics and behavioral science. Am. Econ. Rev. 49 (3), 253–283.

Starr, M.K., 1962. Product Design and Decision Theory. Prentice-Hall, Englewood Cliffs, N. J.

Singh, R., Reed, P.M., Keller, K., 2015. Many-objective robust decision making for managing an ecosystem with a deeply uncertain threshold response. Ecol. Soc. 20 (3), 12. http://dx.doi.org/10.5751/ES-07687-200312.

Triantaphyllou, E., 2000. Multi-criteria Decision Making: a Comparative Study. Dordrecht. Kluwer Academic Publishers, ISBN 0-7923-6607-7, The Netherlands.

Turner, S.W., Marlow, D., Ekström, M., Rhodes, B.G., Kularathna, U., Jeffrey, P.J., 2014. Linking climate projections to performance: a yield-based decision scaling assessment of a large urban water resources system. Water Resour. Res. 50, 3553–3567.

Tsoukias, A., 2008. From decision theory to decision aiding methodology. Eur. J. Oper. Res. 187 (1), 138–161.

Vrugt, J.A., Robinson, B.A., 2007. Improved evolutionary optimization from genetically adaptive multimethod search. Proc. Natl. Acad. Sci. 104 (3), 709–711.

Ward, V., Hadka, D.M., Reed, P.M., 2015. Confronting Decision Cliffs: Diagnostic Assessment of Multi-objective Evolutionary Algorithm Performance for Addressing Uncertain Environmental Thresholds. Accepted to Environmental Modelling & Software. 10.1016/j.envsoft.2015.07.020.

Weaver, C.P., Lempert, R.J., Borwn, C., Hall, J.A., Revell, D., Sarewitz, D., 2013. Improving the contribution of climate model information to decision making: the value and demands of robust decision frameworks. Wiley Interdiscip. Rev. Clim. Change 4, 39–60.

Weitzman, M.L., 1998. Why the far-distant future should be discounted at its lowest possible rate. J. Environ. Econ. Manag. 36, 201–208.

Woodruff, M.J., Hadka, D.M., Reed, P.M., Simpson, T.W., 2012. Auto-adaptive search capabilities of the new borg MOEA: a detailed comparison on product family design problems. In: 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, Indiana.

Woodruff, M., Reed, P.M., Simpson, T., 2013. Many-objective visual analytics: rethinking the design of complex engineered systems. Struct. Multidiscip. Optim. 48, 201–219.

Wu, Y., Shuguang, L., Wende, Y., 2014. A universal model-R coupler to facilitate the use of R functions for model calibration and analysis. Environ. Model. Softw. 62, 65–69.