Data Management Perspectives on Prescriptive Analytics

Alexandra Meliou
University of Massachusetts Amherst, MA, USA

Azza Abouzied
New York University, Abu Dhabi, UAE

Peter J. Haas
University of Massachusetts Amherst, MA, USA

Riddho R. Haque
University of Massachusetts Amherst, MA, USA

Anh Mai
New York University, Abu Dhabi, UAE

Vasileios Vittis
University of Massachusetts Amherst, MA, USA

Abstract

Decision makers in a broad range of domains, such as finance, transportation, manufacturing, and healthcare, often need to derive *optimal* decisions given a set of *constraints* and *objectives*. Traditional solutions to such constrained optimization problems are typically application-specific, complex, and do not generalize. Further, the usual workflow requires slow, cumbersome, and error-prone data movement between a database, and predictive-modeling and optimization packages. All of these problems are exacerbated by the unprecedented size of modern data-intensive optimization problems. The emerging research area of in-database prescriptive analytics aims to provide seamless domain-independent, declarative, and scalable approaches powered by the system where the data typically resides: the database. Integrating optimization with database technology opens up prescriptive analytics to a much broader community, amplifying its benefits. We discuss how deep integration between the DBMS, predictive models, and optimization software creates opportunities for rich prescriptive-query functionality with good scalability and performance. Summarizing some of our main results and ongoing work in this area, we highlight challenges related to usability, scalability, data uncertainty, and dynamic environments, and argue that perspectives from data management research can drive novel strategies and solutions.

2012 ACM Subject Classification Information systems → Decision support systems; Information systems → Database design and models; Information systems → Query languages; Information systems → Database query processing; Computing methodologies → Modeling and simulation; Applied computing → Decision analysis

Keywords and phrases Prescriptive analytics, decision making, scalable constrained optimization

Digital Object Identifier 10.4230/LIPIcs.ICDT.2025.2

Category Invited Talk

Funding *Alexandra Meliou*: NSF IIS-2211918.
Azza Abouzied: ASPIRE Award for Research Excellence (AARE-2020) grant AARE20-307; NYUAD CITIES, funded by Tamkeen under the Research Institute Award CG001.
Peter J. Haas: NSF IIS-2211918.

Acknowledgements This invited talk paper discusses prior and ongoing work, summarizing contributions from prior publications [1, 16, 6, 7, 12].

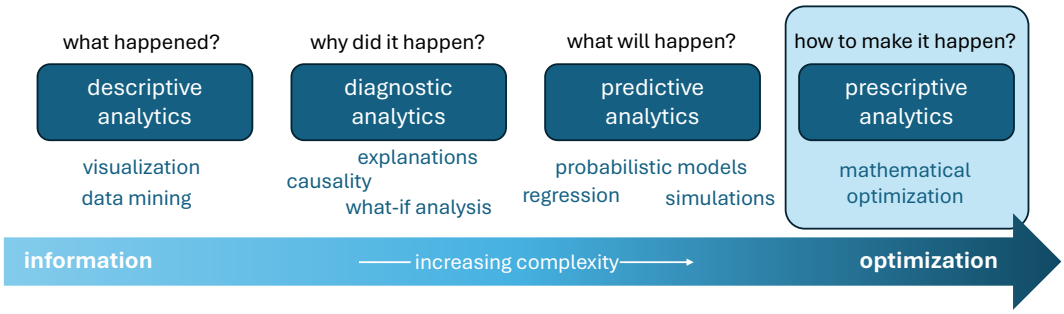

Figure 1 Different types of analytics rely on different tools that range in complexity. Data management research spans this spectrum, though prescriptive analytics has received relatively less attention.

1 Introduction

The unprecedented growth in the size and availability of data has contributed to fundamental shifts in systems, technology, and reasoning, revolutionizing a broad spectrum of applications through enhanced data-driven capabilities. Notably, this applies to decision-making: in a broad range of domains, including finance, transportation, manufacturing, and healthcare, decision makers need to derive *optimal* decisions given a complex set of interacting *constraints* and *objectives* over large datasets. Constraints arise from competition between activities for scarce resources such as time, budget, workers, trucks, tools, etc., as well as from users' tolerance for risk in uncertain environments; objective functions formalize organizational goals such as minimizing costs or delays, maximizing revenue, or minimizing disease mortality.

Constrained optimization problems fall under the umbrella of *prescriptive analytics* [11, 15] – analyzing data, using models, and solving optimization problems to identify the best action to achieve certain goals. Prescriptive analytics introduces significant specification and evaluation challenges compared to other types of analytics (Figure 1), because it requires more complex mathematical tools and representations. Nevertheless, other forms of analytics are also central to the development of optimization models. Optimization models rely on *predictive analytics* – using historical data to predict future trends as well as the future effects of current actions – in order to assess which actions will yield the best results. Predictive models can take the form of complex mechanistic simulation models that incorporate deep domain knowledge or data-driven models such as classical regression models or time series models or, more recently, predictive and generative machine learning models. Moreover, *descriptive analytics* – analyzing historical data to discover patterns and relationships – and *diagnostic analytics* – such as root cause analysis – also play a key role by informing the process of building optimization models so that they capture the most important relationships. Despite the fundamental interplay between descriptive, diagnostic, predictive, and prescriptive analytics, the latter has received much less attention from the database community.

DB research over the analytics spectrum

A common classification groups analytics tasks into *descriptive*, *diagnostic*, *predictive*, and *prescriptive*, based on the underlying questions they seek to address: "what happened", "why it happened", "what will happen", and "how to make it happen", respectively. In all cases, we seek to derive information from data, but the output and algorithmic tools employed differ. Crucially, there is *typically* an increase in computational complexity as we move along this spectrum (Figure 1).

Stock_Investments

ID	Stock	Price	Category	Gain
1	AAPL	195.71	tech	2.13
2	PFE	26.1	pharm	0.15
3	MSFT	373.04	tech	2.89
4	SHEL	65.95	energy	-3.2
...	...	...	...	...

Figure 2 In this simplified example of a list of stocks, we assume the gain attribute to be deterministic, representing the expected gain of buying a stock now and selling it after a specific time period.

Work in data visualization, data mining, and data summarization, among others, can be considered under the umbrella of descriptive analytics, where the goal is to identify patterns, relationships, and statistical trends in the data. Diagnostic analytics typically involves deeper reasoning about the *derivation* of the data, and includes work on provenance, explanations, causality, and what-if analysis. Predictive analytics typically seeks to derive predictive models from the data, which can then be used to produce new data. Related work often sits at the integration of databases with probabilistic models and machine learning tools, which can in turn apply to a variety of problems, such as supporting queries in spatio-temporal databases, value imputation, and so on. Finally, prescriptive analytics is concerned with determining the best way to achieve a desirable outcome, which is typically modeled through mathematical optimization. The search space for these optimization problems can have combinatorial growth, making the design of practical algorithms over large data particularly challenging. Machine learning has been helping to automate these different types of analytics, but still struggles with complex optimization problems.

In-database prescriptive analytics

Optimization problems arise in many application domains, including finance, transportation, and healthcare. Consider a simplified example (Figure 2) where we seek to select a portfolio of stocks, excluding the pharmaceutical sector, with total cost less than $1,000, that maximizes the total gain. A *feasible* solution to this optimization problem is a collection of stocks, with possible repetitions, that satisfy individual or *base* constraints (i.e., category $\neq$ pharm), as well as collective or *global* constraints (i.e., the sum of their costs is less than $1,000). Out of all feasible portfolios, we seek the one that optimizes the objective (i.e., the sum of gain).

Modeling and solving such problems has typically relied on application-specific solutions. Such solutions are often complex and do not generalize; a decision maker seeking to apply optimization techniques in a new application setting must either develop a new custom model from scratch, or must learn the intricacies of generic optimization software, which can be daunting for those with domain, but not optimization, expertise. Moreover, the usual workflow requires that data be extracted from a database and then reformatted and fed into a separate optimization package, after which the output must be reformatted and inserted back into the database; this process is slow, cumbersome, and error-prone. These challenges are further exacerbated by the unprecedented size of modern data-intensive optimization problems.

In-database prescriptive analytics is an emerging research area that aims to provide domain-independent, declarative, and scalable approaches, supported and powered by the system where the data relevant to these problems typically resides: the database. Within this context, we model an important subclass of constrained optimization problems as *package queries*. A package query returns a "package" of tuples that satisfy the query constraints and optimize its objective. In our simplified portfolio example, each stock i can appear x_i times in the package, where each *decision variable* x_i is a nonnegative integer. Both the package cost, which is constrained to lie below a threshold, and the gain, which is to be maximized, are linear functions of the x_i decision variables.

In-database support for such optimization problems makes modeling less ad-hoc, and the overall optimization process, from data preparation through solution and exploration of results, becomes much more efficient. Desirable data management functionality, such as efficient retrieval, consistency, persistence, fault tolerance, access control, and data-integration capability, become an integral part of the system "for free". Interest in native DB support for prescriptive analytics has therefore started to grow [10]. One line of research is exemplified by the SolveDB and SolveDB+ systems [19, 20]. These systems provide semi-declarative languages for specifying a broad range of optimization problems, allow easy sharing of optimization models across sub-problems of an overall predictive-analytics problem, and facilitate plugging in of various prediction models and optimization-problem solvers.

Opportunities for database technologies

The capabilities of mathematical optimization software have grown substantially, and modern solvers are able to handle complex problems. However, a critical challenge with this technology is that most solvers require the entire optimization problem to fit in memory or handle an expensive tradeoff between time and memory. Ultimately, optimization problems are NP hard, and solvers explore the space to find exact answers rather than produce approximations. As a result, existing solvers are often unable to deal gracefully, if at all, with very large amounts of data, which is typical for package queries. Thus, the naive use of black-box solvers is often not viable for modern large-scale optimization problems.

Data management research is well-positioned to tackle this scalability challenge. In principle, optimization problems typically seek to identify a small target set of tuples within a large dataset, and the data management community excels in developing such search technologies. Moreover, many optimization problems over large data exhibit special properties that can be leveraged in developing targeted solutions. Optimization software aims to model general-purpose optimization problems, frequently considering a large number of constraints (e.g., in knapsack problems, the number of constraints grows with the size of the data). In contrast, many in-database optimization problems exhibit a special structure: while the data grows large, the number of constraints is typically small. As an example, in our work [16], we have leveraged this structure to develop a dual simplex algorithm that outperforms state-of-the-art commercial solvers such as Gurobi and CPLEX.

2 The prescriptive analytics landscape

Package queries extend traditional database queries to express constraints and objectives over answer sets. Data management systems to not natively support package queries, and while there are ways to express some cases in SQL, these are cumbersome and inefficient. For example, in the earlier portfolio scenario, if the package is restricted to contain exactly three stocks, we can express it as a query with self-joins:

```
SELECT *
FROM  Stock_Investments S1, Stock_Investments S2, Stock_Investments S3
WHERE S1.category != 'pharm' AND S2.category != 'pharm'
      AND S3.category != 'pharm'
      AND S1.price + S2.price + S3.price <= 1000
ORDER BY S1.gain + S2.gain + S3.gain DESC
LIMIT 1
```

This query is efficient only for constructing packages with very small cardinality (i.e., number of tuples): larger cardinality requires a larger number of self-joins, quickly rendering evaluation time prohibitive. The benefit of this specification is that the optimizer can use the traditional relational algebra operators, and augment its decisions with package-specific strategies. However, this method does not apply for packages of unbounded cardinality.

An alternative is to use recursion in SQL. However, that requires generating a *powerset* table and evaluating each subset to confirm that it satisfies the query constraints. This approach is not declarative, the specification is tedious and complex, and it is not amenable to optimization in existing systems. Crucially, it is also impractical, due to the combinatorial growth of the powerset table.

2.1 Declarative specification of constrained optimization problems

In prior work, we proposed extensions to the SQL language to allow for the declarative specification of package queries. Using the Package Query Language (PaQL) [6], the portfolio example can be expressed as follows:

```
SELECT Package(*)
FROM  Stock_Investments
WHERE category != 'pharm'
SUCH THAT
      sum(price) <= 1000
MAXIMIZE sum(gain)
```

This query includes standard selection predicates on individual tuples (i.e., category != 'pharm'), as well as high-order *package-level* constraints (in the SUCH THAT clause) and objectives (to maximize the total gain). Additional constraints can restrict the package cardinality and repetitions of tuples in a package; see [5] for a complete language specification.

Uncertain data

In many practical settings, the data we want to analyze may be uncertain. In fact, in our earlier simplified portfolio example, it is unrealistic to model gain as a deterministic value. In practice, gain should be modeled as an uncertain attribute, simulated by a stochastic process; further, an optimization problem over such uncertain data may involve probabilistic constraints and objectives.

Figure 3 demonstrates a new version of the portfolio building scenario, in a more realistic setting. The example query in SPaQL (Stochastic Package Query Language) [7] requests a portfolio that costs less than $1000 and for which the probability of losing more than $10 is at most 5% – the latter constraint is called a *Value-at-Risk* (VaR) constraint. The package result is a bag of tuples, i.e., a portfolio of stocks and a selling schedule, that satisfies the constraints while maximizing the expected gain. In recent work [12], we extended the expressiveness of SPaQL to include *Conditional Value-at-Risk* (CVaR) constraints, also called "expected shortfall" constraints [2, 17, 18], expressed in the form:

```
EXPECTED SUM(Gain)>= -10 IN LOWER 0.05 TAIL
```

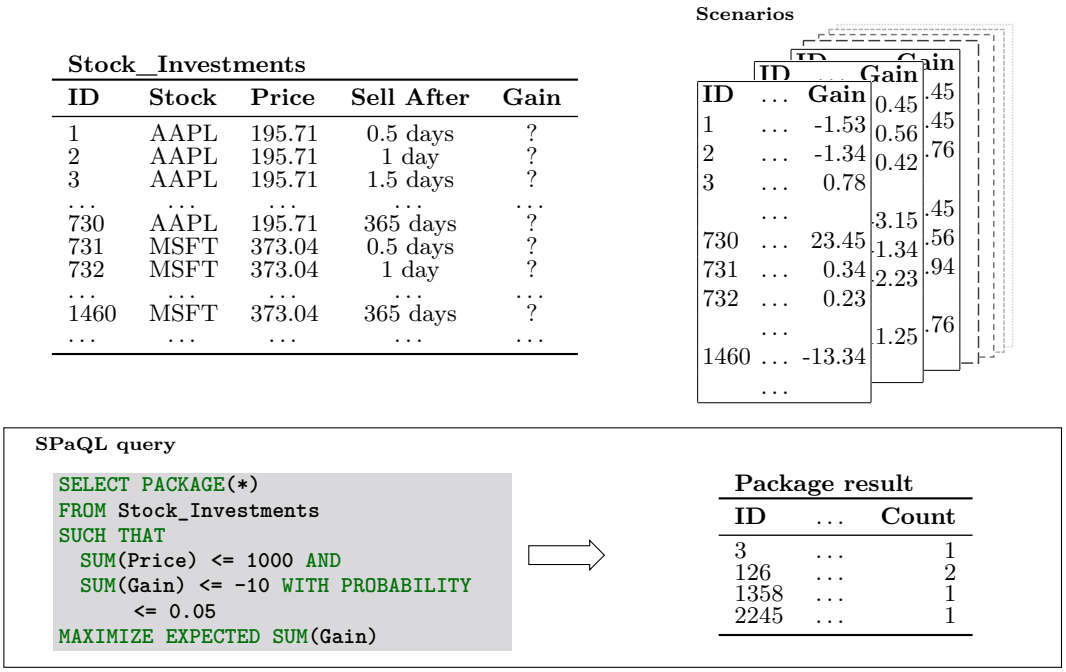

Figure 3 The gain in the Stock_Investments table is an uncertain attribute, simulated by stochastic processes. The scenarios represent different simulations (possible worlds). The example SPaQL query contains a value-at-risk (VaR) constraint, specifying that the probability of total loss (negative gain) exceeding \$10 is at most 5%.

Roughly speaking, the VaR constraint requires that the "bad event" where the loss exceeds \$10 occurs with probability of at most 0.05, whereas the CVaR constraint requires that, given the occurrence of a high-loss event of the form "loss exceeds \$$x$" having 0.05 probability, the expected value of the loss does not exceed \$10.

2.2 Evaluating deterministic problems

The PackageBuilder system [4, 6] transforms a PaQL specification into an Integer Linear Program (ILP) and uses an off-the-shelf ILP solver to compute the desired package. When, as is typical, the number of database rows is large, direct solution by the solver is infeasible because of the large size of the ILP. In prior work, we developed an iterative approximate solution algorithm called SketchRefine [6] to handle large numbers of rows while providing approximation guarantees. Briefly, the algorithm first partitions the rows into groups, where the rows in each group have similar attribute values, and then computes a representative for each group. A small ILP using only the representatives can be then easily solved – the "sketch". The sketch is then iteratively "refined" by carefully replacing each representative using the rows that it represents. The process maintains feasibility of the current solution until the final package is obtained. Limiting the size of each group to be a small number of τ tuples ensures that each refine phase can be executed efficiently and allows for approximation guarantees.

While SketchRefine is computationally efficient for datasets with up to tens of millions of tuples, it has three critical shortcomings:

- **False infeasibility:** Tight constraints can lead to sketches that are overly restrictive and exclude feasible solutions.
- **Suboptimal results:** The partitioning results in aggressive pruning that may discard valuable outlier tuples.
- **Scalability:** With datasets beyond a certain size (about 100 million tuples), SketchRefine can no longer break down the optimization problem effectively, as we will either have too many partitions or too large partitions.

Progressive Shading [16] addresses these issues by making refinement more gradual. It uses a partitioning *hierarchy*, so we can refine over increasingly finer partitions. During this process, it preserves additional neighboring tuples at every level, to ensure that potentially valuable tuples are not pruned aggressively. The novel Dynamic Low Variance (DLV) partitioning algorithm is highly adaptive, and minimizes attribute variance to improve homogeneity within each partition. Its dynamic nature allows it to efficiently partition datasets with widely varying distributions and sizes. It is not enough, however, to apply higher level database systems concepts such as divide-and-conquer over hierarchical partitions to achieve scalability over billions of tuples. We need to open up the solver black box and bring some novel ideas from optimizations research: with progressive shading, we mimicked LP rounding techniques well-studied in OR to solve simpler LP problems across layers, only applying the ILP at the lowest layer; we also implemented our own custom LP solver that exploits the unique structure of package queries that have billions of decision variables but only a handful of constraints.

2.3 Evaluating stochastic problems

Not all data is deterministic; uncertainty is inherent in many application settings, as data may be sampled from predictive models or simulations. Package queries over uncertain data, or stochastic package queries, are more computationally demanding than deterministic package queries, as problems grow not only along the tuple dimension, but also the *uncertainty dimension*. To solve the stochastic optimization problem specified by a SPaQL query, we can approximate it by a deterministic problem via a Monte Carlo technique called Sample Average Approximation (SAA) [14]. SAA creates numerous scenarios ("possible worlds"), each comprising a sample realization for every stochastic attribute of every tuple in the relation, e.g., a realized gain for every Stock–Sell_After pair as in Figure 3. Such scenarios can be created using the Variable Generation functions of Monte Carlo databases like MCDB [13]. In the SAA approximation, expectations are replaced by averages over scenarios and probabilities are replaced by empirical probabilities. E.g., the SAA approximation to the example query in Figure 3 maximizes the average profit over all the scenarios, with at most 5% of the scenarios having losses over $10.

A large number of scenarios is necessary for SAA to achieve good accuracy, especially if the uncertain attributes have high variance. A higher number of tuples further necessitates an increase in the number of scenarios to maintain accuracy, due to the increase in dimensionality [8]. However, using a large number of scenarios is impractical, as it is not only inefficient to generate them, but they also quickly cause the optimization problem to grow too large. Non-convex optimization problems are notorious for requiring large solver runtimes to optimize, and naive in-memory solvers crash when creating packages from millions of scenarios.

SummarySearch [7] is the first approach for in-database processing of stochastic package queries, and introduced two key ideas. First, it uses a small number (in the order of hundreds) of *optimization scenarios* to derive a package; then, it validates whether this package satisfies the constraints over a much larger set of *validation scenarios*. To prevent the optimizer from overfitting to the optimization scenarios and creating packages that will fail validation, SummarySearch combines multiple scenarios into *conservative summary* scenarios. These conservative summaries have a smaller feasibility region: a package that satisfies the constraints over the conservative summary scenarios is guaranteed to satisfy the constraints over the original scenarios. Thus, the resulting package is more likely to pass validation. The algorithm automatically tunes its level of conservativeness to avoid over-restricting the solution space. However, the optimization problems still remain non-convex, albeit with fewer indicator variables than the naive approach.

RCL-Solve [12] resolves this weakness of SummarySearch by replacing non-convex risk constraints with special "linearized" risk constraints. The resulting optimization problems are thus converted to Integer Linear Programs (ILPs). RCL-Solve automatically adjusts the parameters of the linearized risk constraints to find the closest linear approximation of the original non-convex optimization problem that results in a near-optimal and constraint-satisfying package. **Stochastic SketchRefine** scales RCL-Solve with respect to the number of tuples using a divide and conquer approach similar to SketchRefine. Like SketchRefine, it consists of the sketch and refine phases, but contains necessary algorithmic modifications that address challenges raised by the stochastic nature of the problem. In particular, partitioning creates groups of tuples with similar values *and* similar stochastic behavior, and representative tuples are represented by multiple stochastically identical duplicates. The latter allows more flexibility to the optimizer to reduce risk by diversifying packages.

2.4 Dynamic environments

Many datasets are dynamic: values get updated, tuples get added or removed, and the data distribution may shift over time. A key challenge for in-database constraint optimization is that even small changes in the data require computationally intensive package queries to be re-evaluated from scratch. In most practical settings, the overall structure of the constrained optimization problem is likely to remain the same, even if data and perhaps query parameters may change frequently. Re-executing such queries from scratch to maintain results up to date is tedious and computationally expensive, and can render exploratory analysis impractical. We are currently considering several strategies for dealing with evolving data.

- **Incremental package maintenance.** In this direction, the goal is to update packages without re-engaging the optimizer. For example, if a tuple gets deleted (e.g., if a stock is removed), a package containing that tuple could be heuristically updated by looking for a replacement for that tuple (rather than re-executing the optimization). Similarly, if a new tuple is introduced, we can check if its addition to the package, potentially replacing another tuple, improves the result. These approaches are heuristic, but they are likely to perform well in many cases of small changes.
- **Data reduction techniques.** In this direction, the goal is to re-execute the optimization, but over a drastically reduced problem dataset. For example, if several new tuples are added to the data, we could update a previous package result by executing the package query on the small dataset comprising of the tuples in the previous package result and the new tuples (rather than executing the query over the entire dataset). Other variants of this approach can be more sophisticated and use carefully crafted subsets of the data

seen so far (rather than simply the previous package) since previously non-optimal tuples might become optimal later on. This can be achieved by clever reasoning about the likelihood that a certain datapoint could be in a package result. For example, a stock with high price and poor gain would not be selected by the optimizer for the portfolio query, and thus it need not be included in the optimization. Generalizing this intuition has the potential to achieve significant reductions in the data used in the optimization, thus making re-evaluation of package queries more practical.

How to deal with changing query parameters remains a challenging problem. Optimization theory, specifically the theory of "sensitivity analysis" – see, e.g., [3, Chapter 5] – provides some insights on dealing with, for example, small perturbations in bounding values for constraints, but in general a more sophisticated strategy is needed for deciding which previous data to retain.

2.5 Challenges and vision

Prescriptive analytics plays a key role in a broad variety of domains, but has received relatively little attention from the database community. Package queries model a useful class of optimization problems, but many challenges remain in exploring broader and more general settings.

Broader classes of optimization problems. Deterministic package queries try to optimize linear objective functions subject to linear constraints. It would be desirable to extend this class of queries to encompass a broader range of optimization problems. These include non-linear objectives and constraints and multi-stage problems involving a sequence of optimizations in which new information becomes available at each stage. These challenges carry over to the setting of uncertain data, where even more complexities can arise when formulating a stochastic package query. For example, we might want to generalize VaR constraints to require that a *set* of constraints must simultaneously be satisfied with a specified probability.

Distributed and partitioned data. If the dataset involved in a package query is partitioned among heterogeneous servers, and if the query must be solved in a timely manner, then challenges arise as to which partitions should be fetched and the order in which to do the fetching. Even under a centralized processing model, there may be interesting opportunities for query optimization when the input table to the package query must be created via joins and other relational operators.

Expensive scenario generation. As mentioned previously, scenarios are generated in a Monte Carlo database system such as MCDB via calls to variable-generation (VG) functions. In the simplest case, a VG function might simply generate a sample from some standard probability distribution, which can be done very quickly. In complex settings, such as the stock market, a VG call may require running a stochastic simulation in order to generate a sample of, e.g., the future value of an exotic option. Such simulations can be expensive so that the stochastic package query will take too long to solve. One challenge is how to facilitate quick approximate generation of scenarios. The database community has experience in concise approximate representations of complex distributions via histograms and other techniques, and recent work on simulation metamodeling via generative neural networks [9] may be applicable.

Scheduling applications involve allocating tasks to different computing resources, such as processors. A scheduling optimization problem needs to derive an allocation that is consistent (e.g., a processor may only execute one task at a time, or one task may need to precede another), and that optimizes an objective like latency or throughput. This setting requires modeling and reasoning about temporal relationships, which is not straightforward with package queries in their current form. Scheduling problems can grow large in the context of computing providers, and are complicated by the dynamic nature of the setting (as new tasks arrive) and the uncertainty in task completion times, successful execution, and so on.

Partitioning maintenance. While we discussed the dynamic setting from the perspective of updating package results in a manner analogous to continuous queries, changes to the data pose further challenges to approximation algorithms for package evaluation such as SketchRefine and Progressive Shading. Since these methods use auxiliary partitioning structures, these structures need to be maintained dynamically.

High dimensionality. Our work on package queries explored scaling in the data and in the uncertainty dimensions, but we generally focused on data of low dimensionality, where most queries focus on a handful of attributes. The existing methods are unlikely to work well on high-dimensional data, such as vector data, where packages may represent recommendations with diversity-related constraints and objectives (e.g., retrieving news highlights, identifying risk-averse opportunities for financial investments, and forming diverse groups from a pool of applicants). The need for diversity brings forth the need to add constraints to these problem instances that prevent the solver from choosing tuples that are too similar. Identifying pairs of tuples that are "close together" in a scalable manner requires non-trivial customizations to vector database indexing techniques, and system-level considerations to enable parallelization.

Explainability and sensitivity analysis. The complexity of prescriptive analytics tasks implies the need for supporting explanations for package results. Helping users understand why certain tuples were selected in the package, why others were excluded, or how results would change with small perturbations of the query parameters would further enrich in-database prescriptive analytics support.

Autonomous decision-making agents tightly connect the frameworks of prescriptive and predictive analytics, leading to further research challenges. Such agents are often guided by a *policy*: a function that examines the current state of its environment to select an action that leads to the highest possible future rewards. Consider an auto-trader, which uses a stock time-series predictive model to simulate the effects (rewards) of different purchase decisions. This allows for a *planning* phase where the agent can experiment with different decisions to approximate the *value* of being in different possible states and to learn a policy that allows the agent to take an *action* that leads to high-valued states. In this setup, the predictive model is constantly updated from observations of the real world and the agent constantly updates its internal value and policy models to better determine which stocks to buy or sell on a daily basis.

As optimization problems become increasingly data-intensive, database research is poised to make key contributions to the creation of scalable, seamless, data-centric tools for supporting decision making. It is an area with a wealth of open challenges and high potential impact in building systems that enable improved, data-driven decision making.

References

1 Azza Abouzied, Peter J. Haas, and Alexandra Meliou. In-database decision support: Opportunities and challenges. *IEEE Data Engineering Bulletin*, 45(3):102–115, September 2022. URL: http://sites.computer.org/debull/A22sept/p102.pdf.

2 Gordon J Alexander and Alexandre M Baptista. A comparison of var and cvar constraints on portfolio selection with the mean-variance model. *Management science*, 50(9):1261–1273, 2004. doi:10.1287/MNSC.1040.0201.

3 Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

4 Matteo Brucato, Azza Abouzied, and Alexandra Meliou. A scalable execution engine for package queries. *SIGMOD Record*, 46(1):24–31, 2017. doi:10.1145/3093754.3093761.

5 Matteo Brucato, Azza Abouzied, and Alexandra Meliou. Package queries: efficient and scalable computation of high-order constraints. *The VLDB Journal*, 27(1), 2018. doi:10.1007/s00778-017-0483-4.

6 Matteo Brucato, Juan Felipe Beltran, Azza Abouzied, and Alexandra Meliou. Scalable package queries in relational database systems. *PVLDB*, 9(7):576–587, 2016. doi:10.14778/2904483.2904489.

7 Matteo Brucato, Nishant Yadav, Azza Abouzied, Peter J. Haas, and Alexandra Meliou. Stochastic package queries in probabilistic databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 269–283, 2020. doi:10.1145/3318464.3389765.

8 Marco C Campi, Simone Garatti, and Maria Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149–157, 2009. doi:10.1016/J.ARCONTROL.2009.07.001.

9 Wang Cen and Peter J. Haas. Enhanced simulation metamodeling via graph and generative neural networks. In *Winter Simulation Conference, WSC 2022, Singapore, December 11-14, 2022*, pages 2748–2759. IEEE, 2022. doi:10.1109/WSC57314.2022.10015361.

10 Davide Frazzetto, Thomas Dyhre Nielsen, Torben Pedersen, and Laurynas Siksnys. Prescriptive analytics: a survey of emerging trends and technologies. *The VLDB Journal*, May 2019. DOI: 10.1007/s00778-019-00539-y. doi:10.1007/s00778-019-00539-y.

11 Peter J. Haas, Paul P. Maglio, Patricia G. Selinger, and Wang Chiew Tan. Data is dead... without what-if models. *PVLDB*, 4(12):1486–1489, 2011. URL: http://www.vldb.org/pvldb/vol4/p1486-haas.pdf.

12 Riddho R. Haque, Anh L. Mai, Matteo Brucato, Azza Abouzied, Peter J. Haas, and Alexandra Meliou. Stochastic sketchrefine: Scaling in-database decision-making under uncertainty to millions of tuples. *CoRR*, abs/2411.17915, 2024. doi:10.48550/arXiv.2411.17915.

13 Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Chris Jermaine, and Peter J. Haas. The Monte Carlo Database System: Stochastic analysis close to the data. *ACM Trans. Database Syst.*, 36(3):18:1–18:41, 2011. doi:10.1145/2000824.2000828.

14 Sujin Kim, Raghu Pasupathy, and Shane G Henderson. A guide to sample average approximation. In *Handbook of Simulation Optimization*, pages 207–243. Springer, 2015.

15 Irv Lustig, Brenda Dietrich, Christer Johnson, and Christopher Dziekan. The analytics journey. *Analytics Magazine*, november/december 2010. URL: http://analytics-magazine.org/the-analytics-journey.

16 Anh L. Mai, Pengyu Wang, Azza Abouzied, Matteo Brucato, Peter J. Haas, and Alexandra Meliou. Scaling package queries to a billion tuples via hierarchical partitioning and customized optimization. *PVLDB*, 17, 2024. doi:10.14778/3641204.3641222.

17 Alexander J. McNeil, Rüdiger Frey, and Paul Embrechts. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press, second edition, 2015.

18 Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*, pages 270–294. INFORMS TutORials in Operations Research, September 2008. doi:10.1287/educ.1080.0052.

19 Laurynas Siksnys and Torben Bach Pedersen. SolveDB: Integrating optimization problem solvers into SQL databases. In *SSDBM*, pages 14:1–14:12, 2016. doi:10.1145/2949689.2949693.

20 Laurynas Siksnys, Torben Bach Pedersen, Thomas Dyhre Nielsen, and Davide Frazzetto. SolveDB+: Sql-based prescriptive analytics. In *Proc. EDBT*, pages 133–144, 2021. doi:10.5441/002/EDBT.2021.13.