Trusted and Auditable Decision Aids over Data Streams

Dominic J. Duxbury
University of Manchester
M13 9PL, Manchester, UK,
dominic.duxbury@manchester.ac.uk

Norman W. Paton
University of Manchester
M13 9PL, Manchester, UK
norman.paton@manchester.ac.uk

John A. Keane
University of Manchester
M13 9PL, Manchester, UK
john.keane@manchester.ac.uk

ABSTRACT

Data stream management systems exist to support dynamic analysis of streaming data, often to inform decision-making. Decision support systems exist to enable decisions to be made that take into account user priorities. However, although these categories of system are now quite mature, there has been little work investigating their use together. In this paper we bring together a well established streaming platform (Storm) and a widely used decision-support methodology (Analytic Hierarchy Process) to provide dynamic decision support over data streams. In so doing, we also investigate approaches making recommendations auditable (using provenance) and trustable (using explanations). The resulting stream decision support system is illustrated using an application that supports train journey planning.

1 INTRODUCTION

Data streams exist as an abstraction to support analysis of dynamic data as it is produced [11]. Decision Support systems exist to support users in navigating a space of options [3]. These seem to be complimentary paradigms, which can be brought together to support decision making with dynamic data. Current practice in stream data processing makes extensive use of Stream Processing Engines (SPEs) which provide a framework for acting upon elements in a stream. For decision support, an interesting problem is how to build on these capabilities to support real-time decision support over streams.

For real-time decision support systems, the choices made by decision makers often affect the state of the system. It is therefore useful to model decision makers as not just users, but as components of a cyber-physical-social system (CPSS). CPSS span the physical, information, cognitive and social domains. In the CPSS field, human users are considered a component of the system; falling within the cognitive domain [7]. Human components can be a necessary part of a system, such as when making life or death decisions. Decision support systems are therefore often vital, as they bridge the information and cognitive domains by distilling data to assist decision makers.

Decision support systems are enabled by decision analysis. Decision analysis is the field concerned with the study of complex decisions. Multi-criteria decision analysis is a sub-discipline of decision analysis comprising techniques for evaluating solutions with multiple conflicting criteria [3]. A common example of this is purchasing a car; the safest car is not often the cheapest and so these criteria are conflicting. These criteria can have different importance to different decision makers so we require a method for users to specify their preferences. If the values of these criteria are also changing then we call the problem dynamic. In this paper we outline our approach to building a decision support platform for these dynamic multi-criteria optimisation problems.

First International Workshop on Data Science for Industry 4.0.

Decision support systems are only useful if they are trusted by a decision maker. Trust is especially challenging when working with dynamic data; a decision maker does not have time to ascertain if a black box system has made a mistake, and therefore it is beneficial to provide provenance data to the decision maker, ensuring that the information motivating a recommendation is readily available. Data provenance provides a historical record of data and its origins, which allows the user to assess data quality and suitability. In addition to the underlying evidence, it is also important that the user has some understanding of the space of possible solutions; as a result, some form of explanation mechanism is required that explains how a recommendation has been arrived at, and/or describes the relationship between alternative options.

All this is required in a context where there may be genuine uncertainty relating to criteria that inform a recommendation. As such, it is important for maintaining trust to ensure that the uncertainty intrinsic in a recommendation is either presented to a user or able to be reflected within the decision-making process.

Drawing this together, we have the following 5 desiderata for dynamic multi-criteria decision support systems:

(1) declarative specification of preferences,
(2) dynamic revision of recommendations,
(3) provenance capturing the data underpinning decisions,
(4) explanation of how a proposal was made, and
(5) explicit support for uncertain data.

To investigate how these desiderata can be supported in stream decision support, a running example based on train journey planning is introduced in Section 2. An architecture for dynamic decision support is described in Section 3. The application of the architecture to support the above desiderata is discussed in Section 4. Section 5 describes some related work, and conclusions are presented in Section 6.

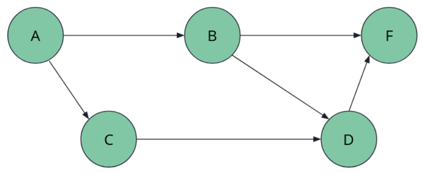
2 MOTIVATING EXAMPLE

To illustrate multi-criteria decision support over streams, we consider an application relating to train journey planning. We assume that a user can state where they need to go *from* and *to*, along with the proposed start time. We also assume that the most suitable journey time for a user may depend on different criteria, specifically the *arrival time* of the journey, the *price* of the journey, and the *number of changes*.

For example, in Figure 1, a decision maker must choose a route from A to F in a way that takes into account price, arrival time and number of changes.

Table 1 shows the solutions to this example. We note that the solution *ABF* dominates *ABDF* as it is equal or better for all criteria values. This leaves us with two potential solutions; *ABF* and *ACDF*. A business person may prefer *ABF* because it is quicker, whereas a student may prefer to save money and take *ACDF*. There is no optimal solution for everyone and so we require user specification of criteria preferences (Desiderata 1).

Figure 1: Example Train Routing Scenario



Circles and arrows depict stations and trains respectively.

Solution	Price (£)	Changes	Arrival Time
ABF	15	1	14:00
ABDF	16	2	14:00
ACDF	9	2	14:40

Table 1: The solutions to figure 1.

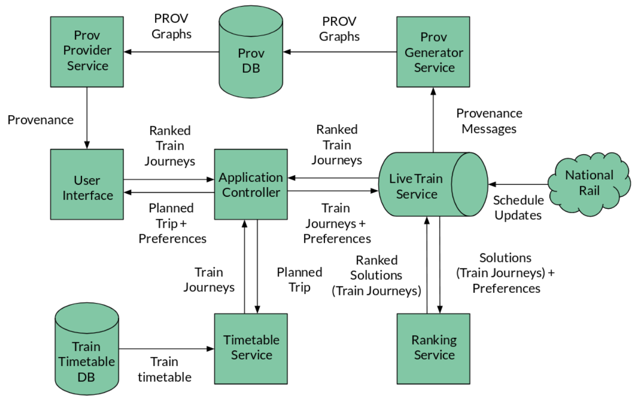


Figure 2: Prototype Architecture

One such criterion, arrival time, indicates the expected arrival time of a journey. This is subject to change, as trains may be delayed or lines closed. Ticket prices are also subject to change up until the time of purchase. If a train is delayed or the price increases, the resulting solution may no longer be optimal, therefore dynamically revising recommendations (Desiderata 2) to reflect the most recent information is clearly beneficial. The user may also move between stations as a part of their interaction with the system; hence requiring an entirely new set of solutions.

A decision maker may see these solutions and choose option *ACDF* because they believe it will only take 10 minutes. However, this route could unreliable due to engineering works, so it may be important for the user to understand the source and derivation of criteria values (Desiderata 3) to improve trustability, or to understand the uncertainty that is characteristic of this particular train service (Desiderata 5).

Finally, after expressing their preferences, accepting criteria values and understanding uncertain aspects, a user is left with a recommended journey. It may be difficult to trust this recommendation without understanding why it was selected. Therefore we should provide the user with an explanation of where the recommendation falls in the solution space, so that they can understand the trade-offs being made, and how this ties into their criteria preferences (Desiderata 4).

3 ARCHITECTURE

To evaluate our approach, a prototype platform has been developed. This platform implements our desiderata from Section 1, whilst providing decision support for train route planning. The system utilises a micro-services architecture shown in Figure 2.

The decision maker operates the decision support system through the user interface. The user inputs details for a planned trip; an origin station, a destination station and a departure time. The user also must specify their preferences with regard to the criteria. This information is sent with a request to open a web-sockets connection to the *Application Controller*. The *Application Controller* holds the state of the train journeys (solutions) within the system. The controller uses the planned trip to build an http request to send to the *Timetable Service*.

Our architecture requires a solution service to generate the initial solution space. The *Timetable Service* is the implementation of the solution service for the train route planning scenario. The service generates a list of train journeys between the requested origin and destination stations at the specified departure time. Initial values are then calculated for all criteria. The *Timetable Service* returns an unranked list of train journeys which are passed from the *Application Controller* to the *Live Train Service*. A streaming component is also required to update the dynamic criteria and to produce a new ranking in real-time. The *Live Train Service* is an implementation of this component for the train scenario. In this case the live train service must update the expected train arrival time. The *Live Train Service* is initialised with a list of train journeys, which are ranked by the *Ranking Service*. A stream of UK wide train updates from *National Rail* is filtered, and matching updates are used to update criteria values. The updated list of train journeys is then re-ranked by the *Ranking Service*. The output stream of ranked train journeys is communicated to the *User Interface* over web-sockets.

The *Ranking Service* accepts a specification of preferences and a list of solutions, to produce a ranking. This ranking is calculated through the application of the Analytic Hierarchy Process, a popular method for multi-criteria decision analysis. The criteria and criteria behaviour are specified through the configuration. For example we specify that price is a criterion and should be minimised. This allows the service to remain generic. The other generic component is the provenance sub-system. The provenance sub-system generates, stores and serves provenance data within the platform. This subsystem is made up of a message queue, a database (*Prov DB*) and two services; one for generating provenance (*Prov Generator Service*), one for serving it (*Prov Provider Service*). The sub-system receives messages from the streaming service which are processed to produce provenance graphs.

3.1 Architecture Components

In this subsection, we provide further details of the components in Figure 2.

Live Train Service. The live train service applies Apache Storm to transform streams of tuples. Apache Storm is an open source SPE which utilises three abstractions; spouts, bolts and topologies. Spouts produce streams. Bolts consume any number of streams to produce new output streams. A topology describes a network of spouts and bolts. Within our streaming component we instrument these operators to extract provenance data. We extend the base classes for bolts and spouts to produce two

Operator	Input	Output
NationalRailSpout	N/A	<timestamp :: Timestamp, id :: trainID, destination :: String, newExpectedArrival :: Timestamp>
DelayBolt	NationalRailSpout	<timestamp :: Timestamp, journeys :: [Journey]>
RankingBolt	DelayBolt	<timestamp :: Timestamp, rankedJourneys :: [<score :: Double, journey :: Journey>] >

Table 2: Input and Output types for each operator

new provenance aware classes; ProvenanceAwareBolt and ProvenanceAwareSpout. An example of a bolt extending this class is shown in Listing 1. *Execute* defines how a bolt processes each tuple and *declareOutputFields* declares the shape of tuples in the output stream. An operator inheriting from these classes will write provenance information concerning its inputs and outputs to the provenance sub-system.

For the train route scenario we have three operators; *NationalRailSpout*, *DelayBolt* and *RankingBolt* . The *NationalRailSpout* produces a stream of delays, the *DelayBolt* applies relevant delays to a list of journeys and the *RankingBolt* interfaces with the *Ranking Service* to calculate a score for each journey. Table 2 shows the input and output tuples for each operator. We instrument all the operators to supply us with provenance regarding the history of solutions, their criteria values and the resulting ranking.

```
public class ExampleBolt extends ProvenanceAwareBolt {
  public void execute(Tuple tuple) {}
  public void declareOutputFields(Declarer declarer) {}
}
```

Listing 1: Code for a provenance aware bolt

Ranking Service. To calculate a recommendation we apply the Analytic Hierarchy Process (AHP) [14]. AHP is a structured technique for organising and analysing complex decisions. AHP consists of an overall goal, a group of options or alternatives for reaching the goal and a group of factors or criteria that relate the alternatives to the goal; the criteria can be further broken down. These criteria generally have different values for different decision makers and so the algorithm requires users to express their preferences. The user preferences are expressed in the form of pairwise comparisons. For instance, a decision maker could express that “Price is more important than Travel Duration”. Pairwise comparisons are easy for a user to express and model the users knowledge within the system. The comparisons are then used to generate weightings for each criteria.

To produce a ranking, criteria values must also be scored. To do this the values are first normalised according to the range of values across all solutions using the following formula:

$$Norm(x) = \frac{x - minX}{maxX - minX}$$

Where minX and maxX are the smallest and largest criteria values respectively. The values are then compared pairwise to generate a comparison matrix. For three solutions S_1, S_2 and S_3 and a criterion X with normalised criteria values x_1, x_2, x_3, we would generate a comparison matrix C.

$$C = \begin{array}{c|ccc} & S_1 & S_2 & S_3 \\ S_1 & 1 & f(x_1,x_2) & f(x_1,x_3) \\ S_2 & f(x_2,x_1) & 1 & f(x_2,x_3) \\ S_3 & f(x_3,x_1) & f(x_3,x_2) & 1 \end{array}$$

We provide two separate formulas for comparing criteria values, depending on whether the values fall along a linear scale (1) or an exponential scale (2). These formulas map two normalised values (x, y) to the fundamental scale proposed by Saaty [14]. For the train route planning scenario we apply the first formula (1), because all criteria form a linear scale. E.g. train prices might be £10, £15, £20 for three alternative routes and not £10, £100, £1000.

$$f(x,y) = |(x - y) \times 8| + 1 \quad (1) \qquad f(x,y) = \frac{e^x}{e^y} \quad (2)$$

The eigenvalues of the comparison matrix for each criterion represent the score for the respective criteria value of each solution. The criteria value scores are then multiplied by the relevant criteria weightings and summed across each solution. This process produces the scores which are used to derive a global ranking.

The normalisation of criteria values can cause some brittleness in the results when we only have a small range. If the algorithm is supplied with two journeys, one costing £50 and another £51 these are seen as the best and worst possible price and so scored accordingly. It would be beneficial for the algorithm to recognise that there is little difference between these two prices. We aim to solve this by allowing those implementing the framework to specify a range of possible values for a criterion.

The decision support component operates over web-sockets. The service requires a configuration file when a connection is opened, providing information about criteria. Critically the configuration indicates the number of criteria and whether numerical criteria should be maximised or minimised. The configuration also allows us to indicate how we should compare non numerical criteria. Once a connection is opened, AHP is applied to a stream of solutions, producing a stream of rankings.

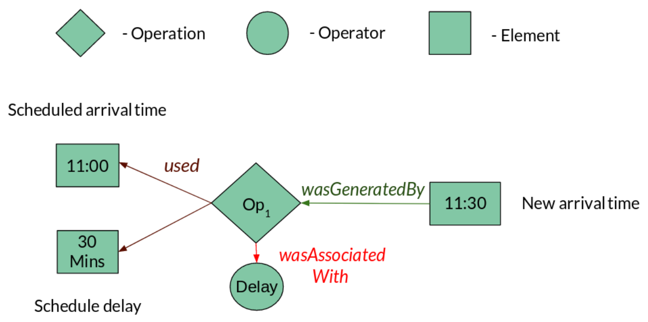


Figure 3: Provenance graph for a train schedule update

Provenance Sub-system. The provenance sub-system processes messages from the streaming system and stores the output in a database for future querying. To store this data we choose to conform to the PROV standard [10]. PROV defines a data model consisting of a set of vertices and edges for modelling provenance as graphs. We adapt a subset of these to map to concepts from data stream analysis. For vertices we use entities, activities and agents. For edges we use *wasGeneratedBy*, *used* and *wasAssociatedWith*.

The PROV data model describes entities as "an immutable piece of state", activities as "dynamic aspects of the world which produce entities" and agents as "parties which take a role in activities". We model stream elements as entities, stream operations as activities and stream operators as agents. Note, we call a set of inputs and outputs a stream operation. The stream operator refers to the operator applied to these inputs to produce the outputs.

Edges describe the relationships between two entities. *wasGeneratedBy* links an entity to the activity which generated it. *used* links an activity to an entity it consumed. *wasAssociatedWith* links an activity to an agent associated with it. We say a stream element was generated by a stream operation. These operations *used* a stream element or window of elements. The operation also *wasAssociatedWith* the operator which was applied. An example provenance graph is shown in Figure 3. This example shows the derivation for an expected train arrival time. The new *arrival time wasGeneratedBy* an operation which *used* the *scheduled arrival time* and the *schedule delay*. The operation *wasAssociatedWith* the delay operator (*DelayBolt*).

3.2 Framework Concepts

In the remainder of this section, we explain what we mean by *explanation* and *uncertainty* and how these concepts surface within our architecture.

Explanation. The AHP algorithm outputs a weight vector for criteria and a score for each solution. Whilst this is useful for constructing a ranking, these values are difficult for a human to interpret. Therefore we require some further explanation of how the system arrived at a recommendation. Fundamentally we describe explanation as a description of how a set of criteria preferences are used by AHP to select a solution from a solution space. Perhaps the most important part, is an explanation of the trade-offs and benefits of a recommendation and how this ties into the specified user preferences. For instance, in the case of train route planning, a user could specify that price is critical to them. Assuming the system recommends *ABC*, the cheapest option, a simple explanation would be that *ABC* is the cheapest train and price is the most important criterion.

Our recommendations are dynamic and so it is important that an explanation can be processed by the user quickly. This lead us towards visual forms of explanation such as bar and spider charts. Spider charts visualise multi-variate data as a shape constructed from three or more quantitative variables across axes stemming from the same point. Typically a chart with a larger area represents a better solution, but these charts can be misleading as the order of criteria can greatly affect the area. For this reason we chose instead to visualise the solution space through bar charts where the values for each criterion and solution are plotted side-by-side. Bar charts are one of the most simple forms of data visualisation, leaving less room for misinterpretation.

Uncertainty. Uncertainty is modelled using cumulative probability density functions (CDFs) drawn from historical data. These functions capture information regarding the potential values of an uncertain criterion for a particular solution. Arrival time is an uncertain criterion for train route planning. We derive a CDF of arrival times for a journey from the historical performance of the trains travelling the same route. These CDFs are a simple model, capturing the distribution of potential criteria values. Through

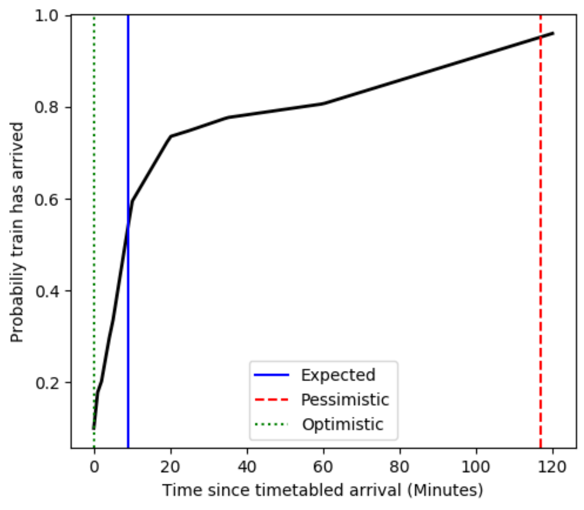


Figure 4: Cumulative Density Function for Arrival Time

this distribution we can view the probability of the potential risks (lateness) for a journey. CDFs serve as alternative to criteria values for uncertain criteria but we require a method of comparing two CDFs. To do this we extract three key values from the distribution; optimistic, expected and pessimistic values. For a CDF f we define optimistic, expected and pessimistic values as x such that $f(x) = 0.05$, $f(x) = 0.5$ and $f(x) = 0.95$ respectively. An example for train arrival times is shown in Figure 4. The user interface allows the decision maker to toggle which of these three values is fed into the ranking algorithm.

4 MOTIVATING EXAMPLE APPLICATION

In this section we explain how the user interacts with the system and how this interface supports the five desiderata from Section 1. The user interface aims to target end-users, rather than decision scientists [16]. The user interface for the train route planner is shown in Figure 5.

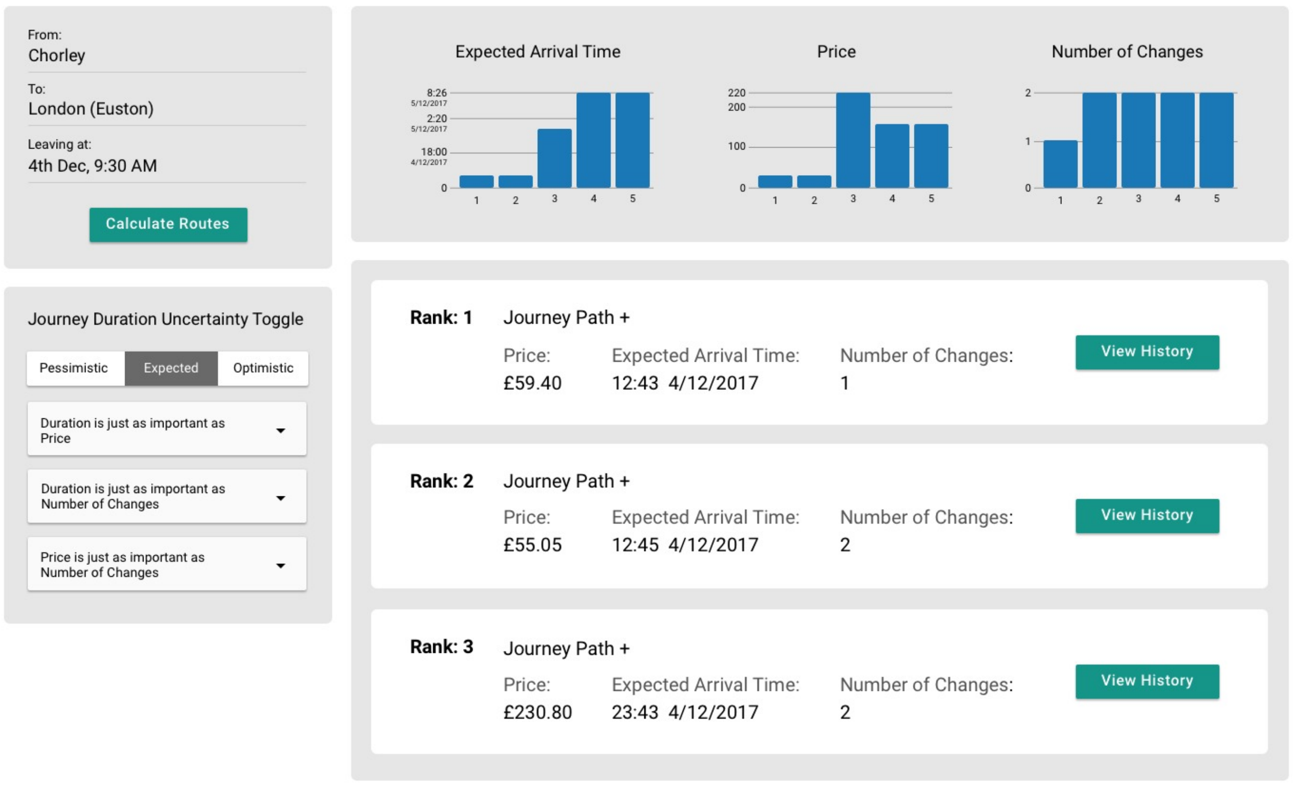
For a decision maker planning a train journey, the first task is to specify the planned trip. The top left corner shows the trip input form, where the user can input where they wish to travel *From* (Origin Station), *To* (Destination Station) and the time they are *Leaving At* (Departure Time). Once these values are set the user can click *Calculate Routes* to generate a set of possible journeys. The next task is for the user to specify their preferences (Desiderata 1). In our user interface these pairwise user preferences are located in the bottom left. In Figure 5 the preferences are set to default, with all criteria equal. Each pair can be set through a drop-down menu one of five potential values;

(1) *X is much more important than Y,*
(2) *X is more important than Y,*
(3) *X is just as important as Y,*
(4) *X is less important than Y,*
(5) *X is much less important than Y.*

These preferences can be changed at any point, triggering the system to re-rank the journeys.

Once the planned trip and preferences have been detailed the user is presented with the top five ranked journeys (the fourth and fifth fall below the fold). Immediately the user can

Figure 5: Route Planning User Interface



From:
Chorley

To:
London (Euston)

Leaving at:
4th Dec, 9:30 AM

Duration is just as important as Price

Duration is just as important as Number of Changes

Price is just as important as Number of Changes

Rank: 1	Journey Path +			
	Price: £59.40	Expected Arrival Time: 12:43 4/12/2017	Number of Changes: 1	View History
Rank: 2	Journey Path +			
	Price: £55.05	Expected Arrival Time: 12:45 4/12/2017	Number of Changes: 2	View History
Rank: 3	Journey Path +			
	Price: £230.80	Expected Arrival Time: 23:43 4/12/2017	Number of Changes: 2	View History

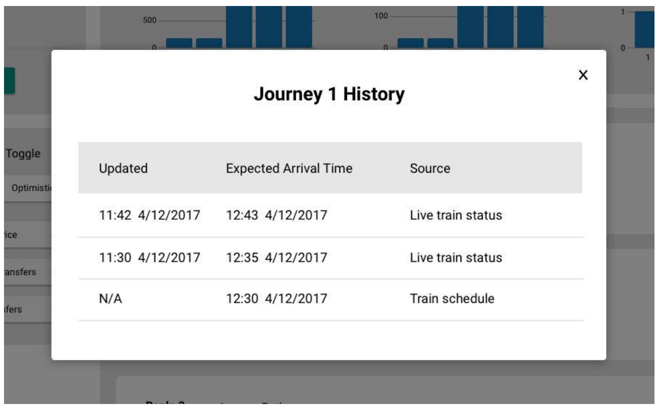
view criteria values of each journey (*Price* , *Arrival Time* and *Transfers*). These values and the resultant ranking are updated continuously once routes have been calculated (Desiderata 2). To prevent information overload some extra details are hidden. Clicking the plus next to *Journey Path* displays the information needed to undertake a journey, including the journey path and the trains of which the journey is composed. Each journey also has a *View Detail* button, which allows the user to view provenance information in a pop-up window (Desiderata 3). The design for this window is shown in Figure 6. Here the user can view the history of values for *Arrival Time* and the data sources.

The values for each of the criteria are shown in the bar charts at the top of Figure 5, with the x-axes ordered according to the ranking. These charts allow the user to visually compare a recommendation (the furthest left value) to the solution space (all other values). The charts are also ordered according to the weighting calculated through AHP, with the most important criteria appearing on the left. This means a user can both understand the trade-offs of a recommendation and how this ties into their specified preferences (Desiderata 4).

Finally the user can toggle between *Pessimistic* , *Expected* and *Optimistic* modes for the predicted arrival time by clicking the corresponding button. These modes simply change the value extracted from the CDF, as described in Section 3.2 (Desiderata 5). *Expected* values are more useful for users making a journey many times (such as commuters) whereas *pessimistic* values would be more important in a scenario where a user is travelling for something more time critical (such as a job interview).

5 RELATED WORK

This paper has proposed an approach for the integration of streaming data with decision support methodologies, with a view



Journey 1 History

Updated	Expected Arrival Time	Source
11:42 4/12/2017	12:43 4/12/2017	Live train status
11:30 4/12/2017	12:35 4/12/2017	Live train status
N/A	12:30 4/12/2017	Train schedule

Figure 6: Provenance Data for an Arrival Time

to enabling users to make decisions that reflect their priorities in the context of a changing physical environment. In this section, we review related work on the intersection of cyber-physical systems (CPS) with decision support, stream data analytics and provenance for data streams.

In relation to CPS, decision support is growing in significance. CPS with key decision support components are being widely adopted in the medical field ([4, 19]). These systems advise doctors in the diagnosis and treatment of patients. Liu *et al.* [7] outlines a framework in the context of command and control; highlighting how decision support can be integrated within a larger CPS and the benefits of doing so. Wang [18] *et al.* make the argument for referring to CPS as cyber-physical-social systems (CPSS). This paper argues the importance of the human aspect within CPS, identifying that users should be more closely integrated within the systems they control. Our architecture fulfils this paradigm by improving extraction of knowledge (pairwise comparisons) and presentation of knowledge (recommendations).

There is a substantial body of work on stream data analyses, often investigating how specific analyses can be carried out efficiently on rapidly streaming data (e.g. [2, 15]). Here the focus has been more on the intersection of streaming and decision support architectures than on algorithms for stream analytics, although this architectural work would benefit from, and presents specific requirements for, efficient multi-dimensional optimization over streams (e.g. [5]).

It has been recognised that multi-criteria decision support systems need to operate in dynamic environments. For example, Benitez *et al.* [1] and Raharjo *et al.* [13] consider making incremental responses to changes in criteria, but there has been less of a focus on responding to changes in criteria values.

It has also been recognized that provenance for data streams is both important for specific streaming applications where decisions may be audited, but also challenging in relation to scalability [9]. Previous work has involved designing generic approaches to collecting and storing provenance data [8, 12]. These systems provide a generic interface for provenance management but no integration with streaming systems. Lim *et al.* have looked at integrating provenance with streaming systems in the context of sensor networks [6], and Blount *et al.* provided provenance for medical event streams [17]. These papers engineer a solution for generating and managing provenance specific to their respective areas rather than seeking to integrate provenance generation into generic SPEs.

6 CONCLUSIONS

Decision support systems use user-specified criteria to compare candidate solutions within a multi-dimensional space of alternatives. This requirement for user-driven comparison of candidate outcomes is widely recognised in decision support, and seems relevant to streaming applications in transport, healthcare, command and control, etc. In this paper we have identified five desiderata for trusted and auditable decision aids over data streams, described an architecture that supports these desiderata, and illustrated its application to an application in journey planning. Future work includes the evaluation of the approach in different applications, scalability of decision support over high-velocity data streams, and investigation of different approaches to uncertainty.

ACKNOWLEDGMENTS

Dominic Duxbury is supported by an EPSRC iCASE award in association with BAE Systems. The authors would also like to recognise Andrew Campbell and Joseph Allen for their assistance in designing the user interface.

REFERENCES

[1] J. BenÃŋtez, X. Delgado-GalvÃąn, J. Izquierdo, and R. PÃľrez-GarcÃŋa. 2012. An approach to AHP decision in a dynamic context. *Decision Support Systems* 53, 3 (2012), 499 – 506. DOI:http://dx.doi.org/10.1016/j.dss.2012.04.015

[2] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. 2005. Mining Data Streams: A Review. *SIGMOD Rec.* 34, 2 (June 2005), 18–26. DOI: http://dx.doi.org/10.1145/1083784.1083789

[3] Salvatore Greco, Matthias Ehrgott, and JoseÌĄ Rui Figueira. 2016. *Multiple Criteria Decision Analysis.* Springer, Springer, New York, NY.

[4] Yu Jiang, Houbing Song, Rui Wang, Ming Gu, Jiaguang Sun, and Lui Sha. 2017. Data-centered runtime verification of wireless medical cyber-physical system. *IEEE Transactions on Industrial Informatics* 13, 4 (aug 2017), 1900–1909. DOI: http://dx.doi.org/10.1109/TII.2016.2573762

[5] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos. 2008. Continuous K-dominant Skyline Computation on Multidimensional Data Streams. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC '08).* ACM, New York, NY, USA, 956–960. DOI:http://dx.doi.org/10.1145/1363686.1363908

[6] Hyo-Sang Lim, Yang-Sae Moon, and Elisa Bertino. 2010. Provenance-based Trustworthiness Assessment in Sensor Networks. In *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks (DMSN '10).* ACM, New York, NY, USA, 2–7. DOI:http://dx.doi.org/10.1145/1858158.1858162

[7] Zhong Liu, Dong Sheng Yang, Ding Wen, Wei Ming Zhang, and Wenji Mao. 2011. Cyber-physical-social systems for command and control. *IEEE Intelligent Systems* 26, 4 (2011), 92–96. DOI:http://dx.doi.org/10.1109/MIS.2011.69

[8] Peter Macko and Margo Seltzer. 2012. A General-purpose Provenance Library. In *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance (TaPP'12).* USENIX Association, Berkeley, CA, USA, 6–6. http://dl.acm.org/citation.cfm?id=2342875.2342881

[9] Archan Misra, Marion Blount, Anastasios Kementsietsidis, Daby Sow, and Min Wang. 2008. Advances and Challenges for Scalable Provenance in Stream Processing Systems. In *Provenance and Annotation of Data and Processes,* Juliana Freire, David Koop, and Luc Moreau (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 253–265.

[10] Luc Moreau, Paolo Missier, James Cheney, and Stian Soiland-Reyes. 2013. *PROV-N: The Provenance Notation.* World Wide Web Consortium, United States.

[11] S. Muthukrishnan. 2005. *Data Streams: Algorithms and Applications.* now, 2600 AD Delft, The Netherlands. https://ieeexplore.ieee.org/document/8186985

[12] Priya. Narasimhan and Peter Triantafillou. 2012. SPADE: support for provenance auditing in distributed environments. In *Proceedings of the 13th International Middleware Conference.* Springer, Springer, New York, NY, 101–120. https://dl.acm.org/citation.cfm?id=2442634

[13] Hendry Raharjo, Min Xie, and Aarnout C. Brombacher. 2009. On modeling dynamic priorities in the analytic hierarchy process using compositional data analysis. *European Journal of Operational Research* 194, 3 (2009), 834 – 846. DOI:http://dx.doi.org/10.1016/j.ejor.2008.01.012

[14] R. W. Saaty. 1987. The analytic hierarchy process-what it is and how it is used. *Mathematical Modelling* 9, 3-5 (1987), 161–176. DOI:http://dx.doi.org/10.1016/0270-0255(87)90473-8

[15] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. 2013. Data Stream Clustering: A Survey. *ACM Comput. Surv.* 46, 1, Article 13 (July 2013), 31 pages. DOI: http://dx.doi.org/10.1145/2522968.2522981

[16] Sajid Siraj, Ludmil Mikhailov, and John A. Keane. 2015. PriEsT: an interactive decision support tool to estimate priorities from pairwise comparison judgments. *ITOR* 22, 2 (2015), 217–235. DOI:http://dx.doi.org/10.1111/itor.12054

[17] Min Wang, Marion Blount, John Davis, Archan Misra, and Daby Sow. 2007. A Time-and-value Centric Provenance Model and Architecture for Medical Event Streams. In *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments (HealthNet '07).* ACM, New York, NY, USA, 95–100. DOI: http://dx.doi.org/10.1145/1248054.1248082

[18] Ying Ming Wang and Kwai Sang Chin. 2011. Fuzzy analytic hierarchy process: A logarithmic fuzzy preference programming methodology. *International Journal of Approximate Reasoning* 52, 4 (2011), 541–553. DOI:http://dx.doi.org/10.1016/j.ijar.2010.12.004

[19] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. 2017. Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data. *IEEE Systems Journal* 11, 1 (mar 2017), 88–95. DOI: http://dx.doi.org/10.1109/JSYST.2015.2460747