The VLDB Journal
https://doi.org/10.1007/s00778-018-0494-9

REGULAR PAPER

Risk-aware path selection with time-varying, uncertain travel costs: a time series approach

Jilin Hu[1] · Bin Yang[1] · Chenjuan Guo[1] · Christian S. Jensen[1]

Received: 23 June 2017 / Revised: 12 December 2017 / Accepted: 4 January 2018

Abstract
We address the problem of choosing the best paths among a set of candidate paths between the same origin–destination pair. This functionality is used extensively when constructing origin–destination matrices in logistics and flex transportation. Because the cost of a path, e.g., travel time, varies over time and is uncertain, there is generally no single best path. We partition time into intervals and represent the cost of a path during an interval as a random variable, resulting in an uncertain time series for each path. When facing uncertainties, users generally have different risk preferences, e.g., risk-loving or risk-averse, and thus prefer different paths. We develop techniques that, for each time interval, are able to find paths with non-dominated lowest costs while taking the users' risk preferences into account. We represent risk by means of utility function categories and show how the use of first-order and two kinds of second-order stochastic dominance relationships among random variables makes it possible to find all paths with non-dominated lowest costs. We report on empirical studies with large uncertain time series collections derived from a 2-year GPS data set. The study offers insight into the performance of the proposed techniques, and it indicates that the best techniques combine to offer an efficient and robust solution.

Keywords Risk preferences · Stochastic dominance · Uncertain time series · Utility functions

1 Introduction

Due to a combination of factors, including developments in autonomous vehicles, the emergence of mobility-on-demand, flex transportation, and increasing needs for more efficient logistics, it is a safe bet that path selection decisions will increasingly be made algorithmically while taking into account time-varying and uncertain travel costs of candidate paths [1].

For example, in logistics, exemplified by PostNord,[1] and in flex transportation, exemplified by FlexDanmark,[2] origin–destination matrices (*OD-matrices*) [2,3] are often used to schedule trips. In particular, a city or a country is partitioned into N zones, e.g., according to administrative districts or simply using a uniform grid, yielding an $N \times N$ matrix where element (i, j) records the "best" path, along with its travel time, from zone i to zone j [4,5].

To obtain an OD-matrix, we need to select the best path for each origin–destination pair. Given such a pair, the best path is often chosen from among a few candidate paths connecting the origin zone and the destination zone. As an example, we may choose among three paths, P_1, P_2, and P_3, that connect zone i to zone j, as shown in Fig. 1.

An intuitive strategy is to choose a path with the minimum average travel time. However, the use of average costs has shortcomings. Specifically, the travel time of a path varies across time, e.g., due to traffic and the weather; and even at a single point in time, different drivers may travel at different speeds, e.g., due to personal preferences and traffic lights. In

✉ Bin Yang
byang@cs.aau.dk

Jilin Hu
hujilin@cs.aau.dk

Chenjuan Guo
cguo@cs.aau.dk

Christian S. Jensen
csj@cs.aau.dk

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark

[1] http://www.postnord.dk/.

[2] https://www.flexdanmark.dk/.

Published online: 24 January 2018

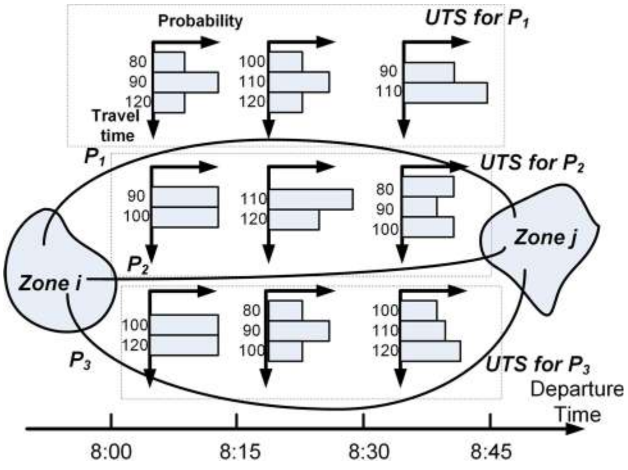

Fig. 1 Motivating example

other words, travel time is time varying [6] and uncertain [7]. Therefore, it is commonplace to model the travel time of a path as an uncertain time series (UTS) [8]: (1) time is first partitioned into fixed-length intervals, e.g., 15-min intervals; (2) a random variable is then recorded for each interval to capture the travel-time distribution of using the path when the departure time falls into the interval.

Figure 1 illustrates the UTSs of our three paths. The UTS of a path contains a random variable that represent the path's travel-time distribution for each departure time interval, e.g., [8:00, 8:15), [8:15, 8:30), and [8:30, 8:45). A random variable is represented as a discrete distribution using a histogram in Fig. 1, but it can also be represented as a continuous distribution using, e.g., Gaussian mixture models [7]. In particular, the histogram modeling the travel time during [8:00, 8:15) for path P_1 indicates the total travel time of path P_1 may be 80, 90, and 120 min with probabilities 0.25, 0.5, and 0.25, respectively.

Since a logistics company generally needs to make deliveries at any time during a day and a flex transportation company generally receives requests for travel at any time during a day, it is useful to compute the best paths for all departure intervals. Thus, the problem becomes one of identifying, for each interval, the "best" path in the interval according to the travel-time distributions of the candidate paths, e.g., P_1, P_2, and P_3 for the example in Fig. 1.

When travel time is uncertain, there may be no unique best path. Table 1 shows the travel times of the three paths for the departure time falls in the interval [8:00, 8:15). The first row states the above-mentioned probabilities of different travel times for path P_1.

Path P_1 has a "wide" distribution, implying that a delivery may arrive at the destination very early, but may also arrive very late. Path P_2 has a "narrow" distribution, thus offering a more predictable delivery time, but also no chance of an early delivery.

Table 1 Uncertain travel times for P_1, P_2, and P_3, [8:00, 8:15)

Travel time (min)	70	80	90	100	110	120
P_1	0	0.25	0.50	0	0	0.25
P_2	0	0	0.50	0.50	0	0
P_3	0	0	0	0.50	0	0.50

A key question is then which path to choose. Some users may prefer to run the risk of making a late delivery in order to have the possibility of making an early delivery, while other users may prefer a more predictable delivery time.

For example, one airport delivery may choose a so-called *risk-loving* path in order to connect with an early flight among several options, while another delivery may choose a *risk-averse* path in order to make sure to catch the last flight of the day. Further, there is evidence that emergency services such as ambulances may choose risk-loving paths [9,10] and that transports involving perishables may prefer risk-averse paths [11]. It is important to provide integrated support for different risk preferences in a framework such as the one proposed in this paper.

In the example in Table 1, path P_1 is risk-loving while path P_2 is risk-averse. A *risk-neutral* user may choose either P_1 or P_2. Path P_3 is not interesting to any user, risk-loving, risk-averse, or risk-neutral, as it offers no benefits over paths P_1 and P_2.

The problem now becomes the following: given a set of candidate paths, a time horizon of interest, e.g., a day, a week, or a month, and a risk preference, e.g., risk-loving, risk-averse, or risk-neutral, identify, for each interval in the time horizon, the "best" path w.r.t. the given risk preference.

We show how different risk preferences can be modeled by means of utility function categories when assuming that the paths with the highest utility are preferred. Further, we provide a connection between stochastic dominance relationships and utility function categories. In short, having chosen a risk preference, and thus a utility function category, if one random variable dominates another random variable, this means that the former random variable must have at least the same expected utility as the latter, no matter which specific utility function in the category is considered. Thus, the path represented by the dominated random variable is uninteresting to any user whose risk preference is captured by the utility function category.

Based on the above, we provide a generic framework that aims to maximally help users with different risk preferences choose paths. In short, the framework compares the random variables of paths and returns exactly the paths with non-dominated random variables. When a user has a risk-loving or a risk-averse preference, thus having a convex or a concave utility function, we compare the random variables of paths using the notions of *second convex order* and *second concave* *order* stochastic dominance to compare random variables. Further, if a user has no clear preference, i.e., has a risk-neural preference, the user may use any monotonous utility function. In this case, we use *first-order* stochastic dominance to compare the random variables of paths.

We provide all necessary techniques for checking first order, second convex order, and second concave order stochastic dominance between two random variables. These are then used as building blocks for techniques that enable the checking of dominance for each interval among different uncertain time series. Further, with the goal of improving efficiency, we present a grouping strategy that groups similar random variables from different intervals, and we provide an optimized group dominance checking technique. If one group dominates another group, all random variables in the former group dominate all random variables in the latter group. When domination between groups occurs, fewer random variables have to be compared.

To the best of our knowledge, this is the first study that provides a comprehensive analysis of the relationship between decision making under uncertainty with utility functions that represents different risk preferences, on the one hand, and different-order stochastic dominance, on the other hand, and it is the first study that enables temporal dominance checking among multiple uncertain time series. In particular, we make four contributions. First, we provide a comprehensive analysis of the relationships among risk preferences, utility functions, and stochastic dominance, and we formulate a novel temporal dominance query on uncertain time series. Second, we present a complete set of techniques for checking first-order, second convex order, and second concave order dominance between two random variables. Third, we present a general grouping strategy and techniques for checking dominance between random variable groups. Fourth, we report on empirical studies based on two large uncertain time series collections that indicate that the proposed techniques are efficient.

The remainder of the paper is organized as follows. Section 2 defines the setting and formalizes the problem. Section 3 covers stochastic dominance relations. Sections 4 and 5 detail the algorithms for checking dominance between random variables and between random variable groups, respectively. Section 6 reports on the empirical study. Related work is coved in Sects. 7 and 8 concludes.

2 Preliminaries

2.1 Uncertain time series

A *road network* is modeled as a graph where *vertices* represent road intersections or road ends and *edges* represent road segments. A *path* is a sequence of adjacent edges that represent connected road segments.

We model the travel time of a path as a *uncertain time series (UTS)*: the time horizon is partitioned into intervals, and each interval is associated with a random variable that represents the path's total travel time when the departure time belongs to the interval. We denote path P_i's UTS as

$$T_i = \left\langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)} \right\rangle,$$

where $X_i^{(j)}$ is the random variable associated with the jth interval. The length of UTS T_i, denote as $|T_i|$, is the number of random variables, or equivalently, the number of intervals in the UTS, i.e., $|T_i| = N$.

Consider the example shown in Fig. 1, where three candidate paths exist that connect a source zone and a destination zone. Thus, we consider three UTSs, T_1, T_2, and T_3, representing the time-varying uncertain travel times of paths P_1, P_2, and P_3, respectively. Assuming that the time horizon is partitioned into 15-min intervals and we consider a time horizon of a week, each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(672)} \rangle$, $1 \leqslant i \leqslant 3$, has length $7 * 24 * 4 = 672$, or equivalently, has 672 different departure time intervals. In particular, random variable $X_i^{(j)}$, $1 \leqslant j \leqslant 672$, represents the travel-time distribution of path P_i when the departure time falls into the jth interval.

The random variables in an UTS may be *dependent* and may also be *independent* of each other. To achieve a generic solution that cover both cases, the paper's solution does not make any assumptions on the dependency among the random variables in an UTS, e.g., the independence or Markov assumption. This has the effect that the proposed solution works both when the random variables are independent or dependent of each other.

2.2 Constructing UTSs from trajectory data

We use map-matched [12,13] GPS trajectories to derive UTSs for paths, as sketched below.

After map-matching, a GPS trajectory, $\langle r_1, r_2, \ldots, r_n \rangle$, is represented as a sequence of *trajectory records* that capture a trip made by a vehicle. A trajectory record $r_i = (t_i, e_i, m_i)$ indicates a trip on edge e_i that started at time t_i and took travel time m_i.

Given an interval I and an edge e, we are able to obtain a multiset of travel-time measurements $M_{I,e} = \{r_i.m_i | r_i.e_i = e \wedge r_i.t_i \in I\}$ from the trajectory records that occurred on edge e during interval I. Next, a random variable that represents the travel-time distribution on edge e during interval I can be learned from the travel-time measurements in $M_{I,e}$ [7,14,15]. For example, if $M_{[8:00,8:15),e_i} = \{80, 90, 120, 90, 120, 90, 80, 90\}$, we are able to learn a discrete random variable {(80, 0.25), (90, 0.50), (120, 0.25)} that then represents the travel-time distribution for edge e_i when the departure time falls into interval [8:00, 8:15). It is also possible to learn a continuous random variable [7]. The techniques we propose later accommodate both discrete and continuous random variables.

Based on the above, for each edge and each interval, we are able to derive a random variable, thus obtaining a UTS for each edge. Next, given a path, we are able to construct a random variable for each interval by "summing up" the random variables of the edges in the path, thus obtaining a UTS for the path. Specifically, the travel-time distribution of path $P_i = \langle e_1, e_2, \ldots, e_M \rangle$ during departure time interval I_j is given by $\odot_{k=1}^{M} \mathrm{RV}(e_k, I_{e_k})$, where $\odot$ denotes convolution of two distributions and $\mathrm{RV}(e_k, I_{e_k})$ denotes the travel-time distribution of edge e_k during interval I_{e_k}, where I_{e_k} is the departure time interval on edge e_k, which may differ from the departure time interval of the path I_j and needs to be progressively updated according to the travel times of e_k's predecessor edges. The details of constructing random variables for paths are covered elsewhere [7,16–18].

2.3 Problem definition

Recall the problem examplified in the introduction. Given a set of candidate paths, a time horizon of interest, e.g., a day, a week, or a month, and a risk preference, e.g., risk-loving, risk-averse, or risk-neutral, we aim at identifying, for each interval in the time horizon of interest, the "best" path w.r.t. the given risk preference.

Since the travel time of a path is represented by UTS, we consider a UTS collection $TS = \{T_1, T_2, \ldots, T_k\}$, where each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)} \rangle$, $1 \leqslant i \leqslant k$, corresponds to a path. The time horizon of interest can be mapped to an interval range $R = [s, e]$ from the sth interval to the eth interval.

Next, let $\mathrm{RVS}^{(j)} = \{X_1^{(j)}, X_2^{(j)}, \ldots, X_k^{(j)}\}$ denote the set of the random variables during the jth interval in TS. Given a user risk preference, we aim at identifying, for each interval j, where $s \leqslant j \leqslant e$, the "optimal" random variables among the random variables in $\mathrm{RVS}^{(j)}$ for the given risk preference. Thus, the path that corresponds to the optimal random variable is chosen as the best path for the jth interval. For example, if X_2^5 is the optimal random variable, then P_2 is chosen as the best path in the 5th interval in the OD-matrix.

Shortly, in Sect. 3, we explain (1) how different user risk preferences can be connected to different-order stochastic dominance relationships and (2) how optimal random variables can be defined under a specific stochastic dominance relationship.

Based on the above, we study the *temporal dominance query* $Q(x, R, \mathrm{TS})$ that takes as input a stochastic dominance relationship x that corresponds to a user risk preference, an interval range $R = [s, e]$, and a UTS collection TS. The query returns a sequence of optimal random variable sets $q = \langle q^{(s)}, q^{(s+1)}, \ldots, q^{(e)} \rangle$, where $q^{(j)}$ is a set of optimal random variables w.r.t. the given stochastic dominance relationship x among all random variables in $RVS^{(j)}$, where $s \leqslant j \leqslant e$. A formal definition of the temporal dominance query is given in Sect. 3.3 after we define different orders of stochastic dominance.

Table 2 Notation

Notation	Definition
X_i, X_j	Random variables
$E(X_i)$	The expectation of X_i
$X_i.\mathrm{min}$	The minimum value in the range of X_i
$X_i.\mathrm{max}$	The maximum value in the range of X_i
$u(a)$	A non-increasing utility function
$u'(a)$	The first derivative of function $u(a)$
$u''(a)$	The second derivative of function $u(a)$
$E_{\mathrm{U}}(X_i)$	The expected utility of X_i
$f_{X_i}(x)$	The probability density function (pdf) of X_i
$F_{X_i}(x)$	The cumulative distribution function (cdf) of X_i
$\widehat{F}_{X_i}(x)$	$\int_0^x F_{X_i}(t)\mathrm{d}t$, the integral of F_{X_i} from 0 to x
$\widetilde{F}_{X_i}(x)$	$\int_x^{+\infty} F_{X_i}(t)\mathrm{d}t$, the integral of F_{X_i} from x to $+\infty$
$X_i \succ_{\mathrm{fsd}} X_j$	X_i first-order dominates X_j
$X_i \succ_{\mathrm{ssd}} X_j$	X_i second convex order dominates X_j
$X_i \succ_{\mathrm{scsd}} X_j$	X_i second concave order dominates X_j
TS	A collection of uncertain time series
T_i	An uncertain time series
RVS	A set of random variables
$\mathrm{RVS}^{(j)}$	A set of the random variables in the jth interval in TS
$O_{\mathrm{fsd}}(\mathrm{RVS})$	The first-order optimal random variable set of RVS
$O_{\mathrm{ssd}}(\mathrm{RVS})$	The second convex order optimal random variable set of RVS
$O_{\mathrm{scsd}}(\mathrm{RVS})$	The second concave order optimal random variable set of RVS

Frequently used notation is listed in Table 2.

3 Stochastic optimality

We explain how user risk preferences, utility functions, and stochastic dominance are connected and define the notion of optimal random variable.

3.1 Decision making under uncertainty

3.1.1 Utility functions

We model decision making as a utility maximization problem. Utility is computed by a utility function that takes *measurements* as input. Without loss of generality, we assume that smaller measurements (e.g., smaller travel times) are preferred. Thus, we consider *non-increasing* utility functions.

Following the example in Sect. 1, we first consider a simple, deterministic case where the average travel time of paths P_1, P_2, and P_3 are 100, 110, and 120 min. Next we consider the utility function $u(x) = 120 - x$, where x is a path's travel time. Thus, the utilities of the three paths are 20, 10, and 0, respectively. Therefore, P_1 has the highest utility and is the optimal choice.

3.1.2 Expected utility principle

Next, to accommodate uncertain measurements, we apply the *expected utility principle* [19,20]. An uncertain measurement is represented as a random variable X_i, and the expected utility of random variable X_i is employed as the utility of the uncertain measurement: $E_{\mathrm{U}}(X_i) = \int_{X_i.\min}^{X_i.\max} u(x) \cdot f_{X_i}(x)\mathrm{d}x$.

Given the utility function $u(x) = 120 - x$ and the uncertain travel times of P_1, P_2, and P_3 as shown in Table 1, the expected utility of P_1, P_2, and P_3 can be calculated as follows: $E_{\mathrm{U}}(P_1) = 0.25 \cdot (120-80) + 0.5 \cdot (120-90) + 0.25 \cdot (120-120) = 25$, $E_{\mathrm{U}}(P_2) = 25$, and $E_{\mathrm{U}}(P_3) = 10$. Therefore, P_1 or P_2 is optimal as both have the highest expected utility.

3.1.3 Risk preferences and utility functions

When making decisions under uncertainty, different users may have different preferences—*risk-loving*, *risk-averse* or *risk-neutral*. A risk-loving user, e.g., an ambulance that transports a cardiac arrest patient, prefers P_1 that offers the possibility of arriving very early although there is also a risk of arriving very late. A risk-averse user, e.g., a vehicle that transports perishables, prefers P_2 that guarantees arrival within 100 min. A risk-neutral user has no clear preference for P_1 versus P_2 and may choose either P_1 or P_2. However, path P_3 is not interesting to any user since it can neither make a user arrive very early nor guarantee arrival within 100 min.

Different risk preferences can be represented by different categories of utility functions. Specifically, we categorize utility functions into *concave*, *convex*, and *other* functions. Recall that we only consider *non-increasing* functions since smaller measurements (e.g., smaller travel times) are always preferred (see Sect. 3.1.1). Figure 2a–c shows a convex function, a concave function, and a function that is neither convex nor concave.

Next, the relationships between risk preferences and utility function categories are shown in Table 3. All users use non-increasing utility functions. The preferences of risk-loving and risk-averse users can be captured by convex and concave functions, respectively. The last column will be explained later in Sect. 3.2.3.

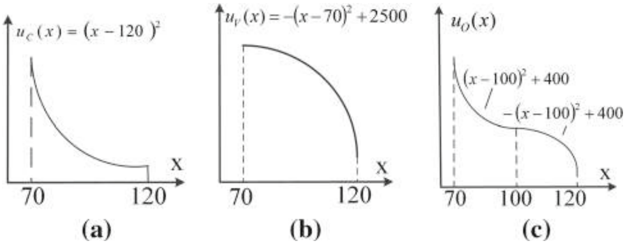

Fig. 2 Categorization of utility functions. **a** Convex, **b** concave and **c** other

Table 3 Risk preferences, utility functions, and stochastic dominance

Risk attitudes	Utility functions	Stochastic dominance
Risk-neutral	Non-increasing	First order
Risk-loving	Non-incr., convex	Second convex order
Risk-averse	Non-incr., concave	Second concave order

Table 4 Expected utilities for paths P_1, P_2, and P_3

	P_1	P_2	P_3	Optimal
u_{C}	**850**	650	200	P_1
u_{V}	1650	**1850**	800	P_2
u_{O}	**450**	**450**	200	P_1, P_2

To further elaborate on the connection between risk preferences and utility functions, we consider the three functions in Fig. 2 and the three paths in Table 1. The expected utilities when using the three utility functions are shown in Table 4, where the highest expected utilities for each utility function are highlighted in bold.

We have already seen that P_1 and P_2 are optimal for risk-loving and risk-averse users, respectively. Table 4 shows that P_1 and P_2 have the largest expected utilities for the convex utility function $u_{\mathrm{C}}(\cdot)$ and the concave utility function $u_{\mathrm{V}}(\cdot)$, which is consistent with the relationship in Table 3. Further, when a user has no clear risk-loving or risk-averse attitude, P_1 and P_2 may be optimal. However, P_3 is not of interest to any user.

3.2 Stochastic dominance

Stochastic dominance enables comparison of two random variables. We introduce three different orders of stochastic dominance and describe how these notations of dominance apply to decision making with different categories of utility functions.

3.2.1 First-order stochastic dominance

Definition 1 First-order stochastic dominance (FSD). Given two random variables X_1 and X_2, if $\forall\ a \in \mathbb{R}^+$, $F_{X_1}(a) \geqslant$

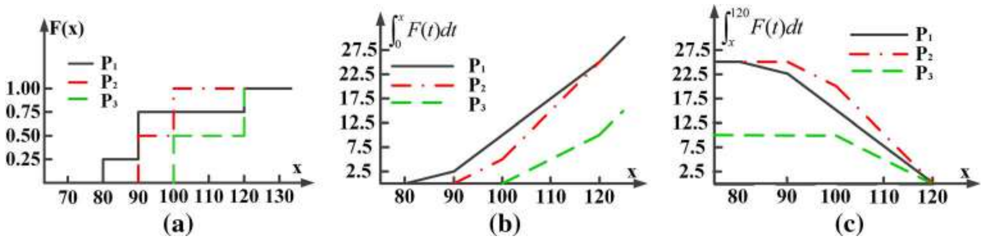

Fig. 3 Distributions of P_1, P_2, and P_3. **a** cdf, $F_{X_i}(x)$, **b** integral of cdf, $\widehat{F}_{X_i}(x)$ and **c** integral of cdf, $\widetilde{F}_{X_i}(x)$

$F_{X_2}(a)$, X_1 first order stochastically dominates X_2, denoted by $X_1 \succ_{\text{fsd}} X_2$.

Following the example in Table 1, the cdfs of the three paths are shown in Fig. 3a. Here, P_1's travel time random variable first order stochastically dominates that of P_3 since for every possible travel time $a \in [70, 120]$, P_1's cdf is no smaller than that of P_3. Similarly, P_2's travel time random variable dominates that of P_3. However, P_1 and P_2 do not dominate each other because P_1 has a larger cumulative probability before 100 min and P_2 has a larger cumulative probability after 100 min.

Theorem 1 *Given a non-increasing utility function $u(a)$, $a \in R^+$, if $X_1 \succ_{fsd} X_2$ then the expected utility of X_1 is no smaller than that of X_2, i.e., $E_U(X_1) \geqslant E_U(X_2)$.*

Proof We prove that $\Delta = E_{\text{U}}(X_1) - E_{\text{U}}(X_2) \geqslant 0$. For readability, we define $X_{\max} = \max(X_1.\max, X_2.\max)$, and $X_{\min} = \min(X_1.\min, X_2.\min) - \epsilon$, where $\epsilon > 0$. Then:

$$\begin{aligned}
\Delta &= \int_{X_1.\min}^{X_1.\max} u(a) f_{X_1}(a)\mathrm{d}a - \int_{X_2.\min}^{X_2.\max} u(a) f_{X_2}(a)\mathrm{d}a \\
&= \int_{X_{\min}}^{X_{\max}} u(a) f_{X_1}(a)\mathrm{d}a - \int_{X_{\min}}^{X_{\max}} u(a) f_{X_2}(a)\mathrm{d}a \\
&= \int_{X_{\min}}^{X_{\max}} u(a) \cdot (f_{X_1}(a) - f_{X_2}(a))\mathrm{d}a \\
&= \int_{X_{\min}}^{X_{\max}} u(a) d(F_{X_1}(a) - F_{X_2}(a)) \\
&= \left[u(a)(F_{X_1}(a) - F_{X_2}(a))\right]_{X_{\min}}^{X_{\max}} \\
&\quad - \int_{X_{\min}}^{X_{\max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))\mathrm{d}a \\
&= - \int_{X_{\min}}^{X_{\max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))\mathrm{d}a
\end{aligned} \tag{1}$$

From line 3 to line 6, we use *integration by parts* [21]. At $X_{\max}$, we have $F_{X_1}(X_{\max}) = 1$ and $F_{X_2}(X_{\max}) = 1$; and at $X_{\min}$, we have $F_{X_1}(X_{\min}) = 0$ and $F_{X_2}(X_{\min}) = 0$. Thus, we have $[u(a)(F_{X_1}(a) - F_{X_2}(a))]_{X_{\min}}^{X_{\max}} = 0$, which explains the final transformation.

Since X_1 first order stochastically dominates X_2, we have $F_{X_1}(a) \geqslant F_{X_2}(a)$ and thus $F_{X_1}(a) - F_{X_2}(a) \geqslant 0$. Since utility function $u(a)$ is non-increasing, we have $u'(a) \leqslant 0$. Thus, $\Delta \geqslant 0$. □

Theorem 1 implies that if a random variable X_1 first order stochastically dominates a random variable X_2, the object represented by X_2 is not interesting because the expected utility of X_1 is always no smaller than the expected utility of X_2. In our example, P_1 and P_2 may be optimal paths, but P_3 is uninteresting regardless of the utility function.

3.2.2 Second-order stochastic dominance

The second-order stochastic dominance between two random variables is defined based on two different forms of integrals of the two random variables' cumulative distribution functions, $\widehat{F}_{X_i}(\cdot)$ and $\widetilde{F}_{X_i}(\cdot)$. In particular, $\widehat{F}_{X_i}(\cdot)$ denotes the integral of the cdf $F_{X_i}(\cdot)$ from 0, shown in Eq. 2.

$$\widehat{F}_{X_i}(a) = \int_0^a F_{X_i}(t)\mathrm{d}t \tag{2}$$

And $\widetilde{F}_{X_i}(\cdot)$ denotes the integral of the cdf $F_{X_i}(\cdot)$ until $+\infty$, shown in Eq. 3.

$$\widetilde{F}_{X_i}(a) = \int_a^{+\infty} F_{X_i}(t)\mathrm{d}t \tag{3}$$

In the two definitions that follow, we distinguish two cases—second convex order stochastic dominance, defined based on $\widehat{F}_{X_i}(\cdot)$, and second concave order stochastic dominance, defined based on $\widetilde{F}_{X_i}(\cdot)$.

Definition 2 Second convex order stochastic dominance (SSD). Given random variables X_1 and X_2, if $\forall\ a \in \mathbb{R}^+$, $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$, X_1 second convex order stochastically dominates X_2 , denoted by $X_1 \succ_{\text{ssd}} X_2$.

Figure 3b shows the integrals of the cdfs from 0 for the three paths. We have $P_1 \succ_{\text{ssd}} P_2$, $P_2 \succ_{\text{ssd}} P_3$, and $P_1 \succ_{\text{ssd}} P_3$.

Theorem 2 *Given a non-increasing and convex utility function $u(a)$, $a \in R^+$, if $X_1 \succ_{ssd} X_2$ then $E_U(X_1) \geqslant E_U(X_2)$.*

Proof We prove $\varDelta = E_{\mathrm{U}}(X_1) - E_{\mathrm{U}}(X_2) \geqslant 0$. According to Eq. 1,

$$\varDelta = -\int_{X_{\min}}^{X_{\max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))\mathrm{d}a$$

Next, we construct a helper function as follows.

$$I(a) = \int_{X_{\min}}^{a} (F_{X_1}(t) - F_{X_2}(t))\mathrm{d}t = \widehat{F}_{X_1}(a) - \widehat{F}_{X_2}(a)$$

Then, we have

$$\begin{aligned}\varDelta &= -\int_{X_{\min}}^{X_{\max}} u'(a)\mathrm{d}I(a)\\ &= -u'(a)I(a)\Big|_{X_{\min}}^{X_{\max}} + \int_{X_{\min}}^{X_{\max}} I(a)u''(a)\mathrm{d}a\\ &= -u'(X_{\max})I(X_{\max}) + u'(X_{\min})I(X_{\min})\\ &\quad + \int_{X_{\min}}^{X_{\max}} I(a)u''(a)\mathrm{d}a \end{aligned} \tag{4}$$

Consider the first term on the right hand side in Eq. 4. Since utility function u is non-increasing, $u'(X_{\max}) \leqslant 0$. Because random variable X_1 second convex order stochastically dominates X_2, we have $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ and thus $I(X_{\max}) = \widehat{F}_{X_1}(X_{\max}) - \widehat{F}_{X_2}(X_{\max}) \geqslant 0$. Thus, the first term is non-negative.

Next, as $I(X_{\min}) = \widehat{F}_{X_1}(X_{\min}) - \widehat{F}_{X_2}(X_{\min}) = 0 - 0 = 0$, the second term is zero.

Finally, we use the property that the second derivative of a *convex* function is *non-negative*. Since utility function u is *convex*, $u''(a) \geqslant 0$. Since random variable X_1 second convex order stochastically dominates X_2, $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ and thus $I(a) = \widehat{F}_{X_1}(a) - \widehat{F}_{X_2}(a) \geqslant 0$. Thus, the third term is non-negative. □

Theorem 2 implies that when random variable X_1 second convex order stochastically dominates random variable X_2, and any risk-loving utility function, i.e., a convex function, is used, the choice represented by X_2 is not interesting because X_1's expected utility is always no smaller than that of X_2.

In our example, $P_1 \succ_{\mathrm{ssd}} P_2$ and $P_1 \succ_{\mathrm{ssd}} P_3$, leaving P_1 as the only optimal choice for a risk-loving users. Recall also that when a user has no clear risk-loving or risk-averse preference, both P_1 and P_2 can be optimal choices. This illustrates that when a user's preference becomes more specific, we can narrow down the optimal choices for the user using second convex order stochastic dominance.

Definition 3 Second concave order stochastic dominance (SCSD). Given random variables X_1 and X_2, if $\forall\ a \in \mathbb{R}^+$, $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$, X_1 second concave order stochastically dominates X_2, denoted by $X_1 \succ_{\mathrm{scsd}} X_2$.

Figure 3c shows the integrals of the cdfs until $+\infty$ for the three paths in our example. According to Definition 3, we have $P_2 \succ_{\mathrm{scsd}} P_1$, $P_2 \succ_{\mathrm{scsd}} P_3$, and $P_1 \succ_{\mathrm{scsd}} P_3$.

Theorem 3 *Given a non-increasing and concave utility function $u(a),\ a \in R^+$, if $X_1 \succ_{scsd} X_2$, then $E_U(X_1) \geqslant E_U(X_2)$.*

Proof We prove $\varDelta = E_{\mathrm{U}}(X_1) - E_{\mathrm{U}}(X_2) \geqslant 0$. According to Eq. 1,

$$\varDelta = -\int_{X_{\min}}^{X_{\max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))\mathrm{d}a$$

Next, we construct a helper function as follows.

$$\begin{aligned} I(a) &= \widetilde{F}_{X_1}(a) - \widetilde{F}_{X_2}(a)\\ &= \int_{a}^{X_{\max}} (F_{X_1}(t) - F_{X_2}(t))\mathrm{d}t\\ &\quad + \int_{X_{\max}}^{+\infty} (F_{X_1}(t) - F_{X_2}(t))\mathrm{d}t\\ &= \int_{a}^{X_{\max}} (F_{X_1}(t) - F_{X_2}(t))\mathrm{d}t \end{aligned} \tag{5}$$

This holds because $F_{X_1}(t) = F_{X_2}(t) = 1$ if $t > X_{\max}$.

Then, we have

$$\begin{aligned}\varDelta &= \int_{X_{\min}}^{X_{\max}} u'(a)\mathrm{d}I(a)\\ &= u'(a)I(a)\Big|_{X_{\min}}^{X_{\max}} - \int_{X_{\min}}^{X_{\max}} I(a)u''(a)\mathrm{d}a\\ &= u'(X_{\max})I(X_{\max}) - u'(X_{\min})I(X_{\min})\\ &\quad - \int_{X_{\min}}^{X_{\max}} I(a)u''(a)\mathrm{d}a \end{aligned} \tag{6}$$

Since $I(X_{\max}) = 0$, the first term in Eq. 6 is 0.

Next, as utility function u is non-increasing, we have $u'(X_{\min}) \leqslant 0$. Further, X_1 second concave order stochastically dominates X_2, so $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$ and thus $I(X_{\min}) \geqslant 0$. Thus, the second term is non-negative.

Finally, we use the property that the second derivative of a *concave* function is *non-positive*. Because utility function u is *concave*, we have $u''(a) \leqslant 0$. Since X_1 second concave order stochastically dominates X_2, we have $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$ and thus $I(a) = \widetilde{F}_{X_1}(a) - \widetilde{F}_{X_2}(a) \geqslant 0$. As a result, the third term in Eq. 4 is non-negative. □

Theorem 3 implies that a user with a risk-averse utility function, i.e., a concave function, is not interested in choosing X_2 if $X_1 \succ_{\mathrm{scsd}} X_2$, no matter the specific form of the concave function. In our example, P_2 is the optimal choice for risk-averse users.

3.2.3 Optimal random variable set

Given a set of random variables RVS, if a random variable is not dominated by any other random variable, it is an optimal random variable, where dominance can be based on first order, second convex order, or second concave order dominance. For example, the first-order optimal random variable set is defined as follows. $O_{\text{fsd}}(\text{RVS}) = \{X \in \text{RVS} | \nexists X' \in \text{RVS}(X' \succ_{\text{fsd}} X \wedge X' \neq X)\}$. The second convex and concave order optimal random variable sets $O_{\text{ssd}}(\text{RVS})$ and $O_{\text{scsd}}(\text{RVS})$ are defined similarly.

To help users make decisions, only the optimal random variables should be returned. In particular, if a user is risk-loving, only the optimal random variable set w.r.t. second convex order dominance, i.e., $O_{\text{ssd}}(\text{RVS})$, should be returned; and if a user is risk-averse, only the optimal random variable set w.r.t. second concave order dominance, i.e., $O_{\text{scsd}}(\text{RVS})$, should be returned; further, if a user is risk-neural, the optimal random variable set w.r.t. first order dominance, i.e., $O_{\text{fsd}}(\text{RVS})$, should be returned. This explains the last column in Table 3.

3.3 Temporal dominance query

Having defined the necessary concepts, we are able to define formally the *temporal dominance query* $Q(x, R, \text{TS})$.

The query takes as input a stochastic dominance relationship parameter x, which can be fsd, ssd, or scsd; a interval range $R = [s, e]$ that represent a time horizon of interest, and a UTS collection $\text{TS} = \{T_1, T_2, \ldots, T_k\}$, where each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)} \rangle$.

The query returns a sequence of optimal random variable sets $q = \langle q^{(s)}, q^{(s+1)}, \ldots, q^{(e)} \rangle$, where $q^{(j)} = O_x(\text{RVS}^{(j)})$ and $\text{RVS}^{(j)} = \{X_1^{(j)}, X_2^{(j)}, \ldots, X_k^{(j)}\}$ denotes the set of the random variables during the jth interval in TS where $s \leqslant j \leqslant e$, and x is fsd, ssd, or scsd.

Temporal dominance queries provide a generic solution to users with varying risk preferences in the sense that:

(1) if a user does not have a clear risk preference, fsd should be used, and the user can choose any object in $O_{\text{fsd}}(\text{RVS})$;
(2) if a user has either a risk-loving or risk-averse preference, ssd or scsd should be used, and the user can choose any object in $O_{\text{ssd}}(\text{RVS})$ or $O_{\text{scsd}}(\text{RVS})$;
(3) if a user provides a specific utility function, we first categorize it as convex, concave, or other; then we only need to compute expected utilities for the objects in $O_{\text{fsd}}(\text{RVS})$, $O_{\text{ssd}}(\text{RVS})$, or $O_{\text{scsd}}(\text{RVS})$, not for other dominated objects, and finally, and return the object with the highest expected utility.

4 Checking stochastic dominance

To efficiently compute temporal dominance queries on UTSs, we first need efficient means of checking stochastic dominance between two random variables. In this section, we present algorithms for checking these three kinds of dominance considered between two random variables.

For a random variable X, we denote the expectation of the random variable as $E(X)$ and the minimum and maximum value of the random variable as $X.\text{min}$ and $X.\text{max}$, respectively. Next, we introduce $X.\text{min}^- = X.\text{min} - \rho$ where ρ is a small positive value. When considering travel time, ρ represents a time that is shorter than the finest time granularity ϵ. For example, if the finest time granularity ϵ is a second, ρ can be half second or a millisecond.

Recall that we use $F_X(\cdot)$ to denote the cdf of random variable X. Thus, we have $F_X(X.\text{min}^-) = 0$; $\forall X.\text{min} < x < X.\text{max}, 0 < F_X(x) < 1$; and $\forall x \geqslant X.\text{max}, F_X(x) = 1$.

4.1 Checking FSD

We first consider a naive algorithm based on the definition of FSD. Next, we propose an initial check strategy to improve the naive algorithm. Finally, we propose a speedup algorithm.

4.1.1 Naive algorithm

Given two random variables X_1 and X_2, a naive algorithm for checking whether X_1 first-order dominates X_2 is to use Definition 1. For each value $a \in Dom$ where $Dom = [\min(X_1.\text{min}, X_2.\text{min}), \max(X_1.\text{max}, X_2.\text{max})]$, we check whether $F_{X_1}(a) \geqslant F_{X_2}(a)$ always holds. If yes, X_1 first order dominates X_2. Otherwise, X_1 does not first order dominate X_2.

Assume that the finest granularity of travel-time measurements is ϵ, e.g., 1 s or 1 min. We have $N = \frac{Dom}{\epsilon}$ possible values to check. Thus, the complexity of the naive algorithm is linear in N, i.e., the number of *comparisons* between two random variables' CDFs. The pseudocode of the naive algorithm is shown in Algorithm 1.

Algorithm 1 NaiveFSD

```
Input:
    Random variables: $X_1$ and $X_2$;
Output:
    Dominance relationship between $X_1$ and $X_2$ w.r.t. FSD;
for each $a \in [\min(X_1.min, X_2.min), \max(X_1.max, X_2.max)]$ do
    if $F_{X_1}(a) < F_{X_2}(a)$ then
        return $X_1$ does not dominate $X_2$;
    end if
end for
return $X_1$ dominates $X_2$;
```

4.1.2 Naive algorithm with initial check

The naive algorithms is inefficient as it needs to check every $a \in [\min(X_1.\text{min}, X_2.\text{min}), \max(X_1.\text{max}, X_2.\text{max})]$. We present an improved algorithm with an initial check. If X_1 and X_2 fail this check, X_1 does not dominate X_2. The initial check is based on Lemma 1.

Lemma 1 *Given two random variables X_1 and X_2, if $X_1 \succ_{fsd} X_2$ then $X_1.min \leqslant X_2.min$ and $X_1.max \leqslant X_2.max$, and $E(X_1) \leqslant E(X_2)$.*

Proof We prove by contradiction that X_1.min $\leqslant X_2$.min and X_1.max $\leqslant X_2$.max.

First, assume that X_1.min $> X_2$.min. Based on the assumption and the definition of X.min$^-$, we get X_1.min $> X_1$.min$^- > X_2$.min. Consequently, we have $F_{X_2}(X_1.\text{min}^-) > 0 = F_{X_1}(X_1.\text{min}^-)$. However, since $X_1 \succ_{\text{fsd}} X_2$, we have $F_{X_1}(x) \geqslant F_{X_2}(x)$ for any $x \in (-\infty, +\infty)$, including the case when $x = X_1.\text{min}^-$. This results in a contradiction. Thus, the assumption is invalid, and we must have X_1.min $\leqslant X_2$.min.

Second, assuming that X_1.max $> X_2$.max, we have that F_{X_1} $(X_2.\text{max}) < 1$ and $F_{X_2}(X_2.\text{max}) = 1$. Thus, $F_{X_1}(X_2.\text{max}) < F_{X_2}(X_2.\text{max})$. However, since $X_1 \succ_{\text{fsd}} X_2$, we have $F_{X_1}(X_2.\text{max}) \geqslant F_{X_2}(X_2.\text{max})$, which also results in a contradiction. Thus, the assumption is invalid, and we must have X_1.max $\leqslant X_2$.max.

Finally, we apply Theorem 1 to prove $E(X_1) < E(X_2)$. Consider a non-increasing utility function $u(x) = -x$. According to Theorem 1, we have $E_{\text{U}}(X_1) \geqslant E_{\text{U}}(X_2)$. Since $u(x) = -x$, we have $E_{\text{U}}(X_1) = -E(X_1)$ and $E_{\text{U}}(X_2) = -E(X_2)$. Thus, we have $E(X_1) \leqslant E(X_2)$. □

Based on Lemma 1, we first check if the three conditions, i.e., X_1.min $\leqslant X_2$.min, X_1.max $\leqslant X_2$.max, and $E(X_1) \leqslant E(X_2)$, hold. If yes, X_1 may stochastically dominate X_2, and then we call the naive algorithm to further check if X_1 dominates X_2; otherwise, X_1 does not dominate X_2. Hence, we propose an *Initial Check* method before calling the naive FSD algorithm, which is shown in Algorithm 2.

4.1.3 Speedup algorithm

We propose a speedup algorithm to verify the first-order stochastic dominance relationship between two random variables, which can significantly reduce the number of comparisons. We first apply the initial check specified in Lemma 1 to filter the cases where two random variables cannot dominate each other. We then assume that X_1.min $\leqslant X_2$.min and X_1.max $\leqslant X_2$.max. Then, the speedup algorithm needs to check whether $F_{X_1}(a) \geqslant F_{X_2}(a)$ for each $a \in [X_2.\text{min}, X_1.\text{max}]$.

Algorithm 2 NaiveFSDInitialCheck

```
Input:
  Random variables: $X_1$ and $X_2$;
Output:
  Dominance relationship between $X_1$ and $X_2$ w.r.t. FSD;
if $E(X_1) \leqslant E(X_2) \wedge X_1.min \leqslant X_2.min \wedge X_1.max \leqslant X_2.max$ then
    Call the naive FSD algorithm on $X_1$ and $X_2$;
else if $E(X_2) \leqslant E(X_1) \wedge X_2.min \leqslant X_1.min \wedge X_2.max \leqslant X_1.max$ then
    Call the naive FSD algorithm on $X_2$ and $X_1$;
else
    return no dominance;
end if
```

The speedup algorithm does not only rely on the two random variables' cdfs, i.e., $F_{X_1}(\cdot)$ and $F_{X_2}(\cdot)$, but also on the integral of cdfs, i.e., $\widehat{F}_{X_1}(\cdot)$ and $\widehat{F}_{X_2}(\cdot)$. This means that $\widehat{F}'_{X_1}(a) = F_{X_1}(a)$ and $\widehat{F}'_{X_2}(a) = F_{X_2}(a)$ for any a. Figure 4a shows an example of the integrals of the cdfs of two random variables X_1 and X_2. Recall that at a point on the curve of an integral of a cdf, the slope of the point is the point's corresponding cumulative distribution.

The speedup algorithm *SpeedupFSD*(X_1, X_2, s, e) follows a divide-and-conquer approach. It takes as input two random variables X_1 and X_2 and a range $[s, e]$, and it returns a Boolean value indicating whether $F_{X_1}(x) \geqslant F_{X_2}(x)$ for any $x \in [s, e]$. The pseudocode is shown in Algorithm 3. First, we call *SpeedupFSD*$(X_1, X_2, X_2.min, X_1.\text{max})$.

Algorithm 3 first checks whether $F_{X_1}(x) \geqslant F_{X_2}(x)$ when x equals to the two boundary values s and e, i.e., whether $F_{X_1}(s) \geqslant F_{X_2}(s)$ and $F_{X_1}(e) \geqslant F_{X_2}(e)$ (lines 1–2). If not, X_1 cannot dominate X_2, and the algorithm returns *False*. Next, Algorithm 3 uses Lemma 2 to check whether the dominance relationship between X_1 and X_2 can be identified by using their cdfs at s and e (lines 4–5).

Lemma 2 *Given random variables X_1 and X_2 and a range $[s, e]$, if $F_{X_1}(s) \geqslant F_{X_2}(e)$ then $F_{X_1}(x) \geqslant F_{X_2}(x) \forall\ x \in [s, e]$.*

Proof Since $F_X(x)$ is non-decreasing, we have $F_X(s) \leqslant F_X(x) \leqslant F_X(e), \forall\ x \in [s, e]$. Thus, we obtain $F_{X_1}(x) \geqslant F_{X_1}(s) \geqslant F_{X_2}(e) \geqslant F_{X_2}(x), \forall x \in [s, e]$. □

To better explain the next lines in Algorithm 3, we introduce four important points: $A = (s, \widehat{F}_{X_1}(s))$, $B = (e, \widehat{F}_{X_1}(e))$, $C = (s, \widehat{F}_{X_2}(s))$, and $D = (e, \widehat{F}_{X_2}(e))$, where points A and C are the left endpoints of the curves of $\widehat{F}_{X_1}$ and $\widehat{F}_{X_2}$ in the range $[s, e]$, and points B and D are the right endpoints. Examples of the four points are shown in Fig. 4a, b.

Next, we construct a line segment l_0 by connecting A and B, and we compute the slope k_{AB} of line segment l_0. At point B, we compute the corresponding slope k_B on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of e on X_1,

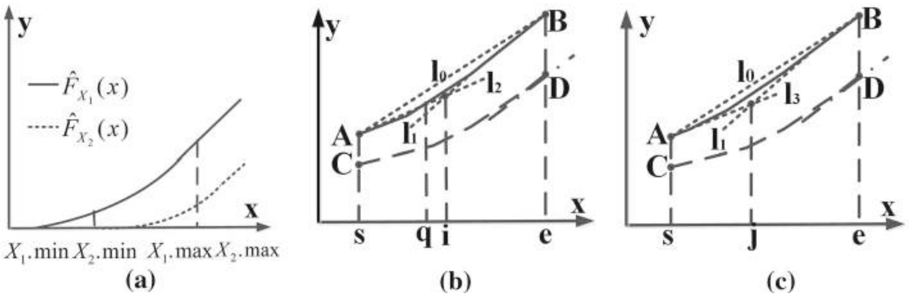

Fig. 4 Speedup algorithm for checking FSD. **a** $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$, **b** Case 1 and **c** Case 2

Algorithm 3 SpeedupFSD (X_1, X_2, s, e)

```
Input:
    Random variables $X_1$ and $X_2$; a range $[s, e]$
Output:
    True if $X_1 \succ_{fsd} X_2$; False, otherwise.
1:  if $F_{X_1}(s) < F_{X_2}(s)$ or $F_{X_1}(e) < F_{X_2}(e)$ then
2:      return False;
3:  end if
4:  if $F_{X_1}(s) \geqslant F_{X_2}(e)$ then
5:      return True; //acc. to Lemma 2
6:  end if
7:  $k_{AB} \leftarrow$ the slope of the line segment $AB$;
8:  $k_A \leftarrow F_{X_1}(s)$; $k_B \leftarrow F_{X_1}(e)$; $k_D \leftarrow F_{X_2}(e)$;
9:  Construct $l_1$ through point $B$ with slope $k_B$;
10: if $k_D < k_{AB}$ then
11:     Construct $l_2$ through point $A$ with slope $k_D$;
12:     $i \leftarrow x$-coordinate of the intersection between $l_1$ and $l_2$;
13:     return SpeedupFSD($X_1, X_2, s, i$) //acc. to Lemma 3
14: else
15:     Construct $l_3$ from point $A$ with slope $k_A$;
16:     $j \leftarrow x$-coordinate of the intersection between $l_1$ and $l_3$;
17:     Boolean $b1 \leftarrow$ SpeedupFSD($X_1, X_2, s, j$);
18:     Boolean $b2 \leftarrow$ SpeedupFSD($X_1, X_2, j, e$);
19:     return $b1 \wedge b2$;
20: end if
```

i.e., $k_B = \widehat{F}'_{X_1} = F_{X_1}(e)$. We have $k_B \geqslant k_{AB}$. We construct a straight line l_1 that goes through point B with slope k_B. At point D, we compute the corresponding slope k_D on the curve of $\widehat{F}_{X_2}$, which equals the cumulative distribution of e on X_2, i.e., $k_D = \widehat{F}'_{X_2} = F_{X_2}(e)$. Then we distinguish two cases depending on the ordering of k_D and k_{AB}.

4.1.4 Case 1 (lines 10–13)

$k_D < k_{AB}$ (see Fig. 4b). We make a straight line l_2 through point A with slope k_D. Since $k_D < k_{AB}$, l_2 and l_1 must intersect, and the x-coordinate of the intersection i must be between s and e.

According to Lemma 3, we have $F_{X_1}(x) > F_{X_2}(x), \forall x \in [i, e]$. Therefore, range $[i, e]$ can be pruned safely and we only need to check if $F_{X_1}(x) \geqslant F_{X_2}(x), \forall x \in [s, i]$. To do this, we recursively call *SpeedupFSD*(X_1, X_2, s, i).

Lemma 3 *If* $k_D < k_{AB}$*, then* $F_{X_1}(x) \geqslant F_{X_2}(x), \forall x \in [i, e]$.

Proof Let the x-coordinate of the intersection between l_2 and $\widehat{F}_{X_1}(x)$ be q (see Fig. 4b). Further, we have $\widehat{F}'_{X_1}(q) = F_{X_1}(q) = k_D$. Since $\widehat{F}_{X_1}(x)$ is convex and increasing, we have (1) $F_{X_1}(a) \geqslant k_D$ if $a \geqslant q$ and (2) $q \leqslant i$. Based on the above, we have $\forall x \in [i, e]$, $F_{X_1}(x) \geqslant F_{X_1}(q) = k_D = F_{X_2}(e) \geqslant F_{X_2}(x)$. □

4.1.5 Case 2 (lines 14–19)

$k_D \geqslant k_{AB}$ (see Fig. 4c). We make a straight line l_3 through point A with slope $k_A = F_{X_1}(s)$. Since k_A must be no larger than k_{AB}, l_3 and l_1 must intersect, and the x-coordinate of the intersection j must be between s and e. In this case, we cannot guarantee $F_{X_1}(a) \geqslant F_{X_2}(a)$ for neither sub-range $[s, j]$ nor sub-range $[j, e]$. Therefore, we recursively call *SpeedupFSD*(X_1, X_2, s, j) and *SpeedupFSD*(X_1, X_2, j, e) to check both sub-ranges.

4.2 Checking SSD

We present a naive algorithm, a naive algorithm with an initial check, and a speedup algorithm to check second convex order stochastic dominance (SSD) between two random variables.

4.2.1 Naive algorithm for SSD

Similar to the naive algorithm for FSD checking, the naive algorithm for SSD checking applies the definition of SSD, i.e., Definition 2. For each value $a \in [\min(X_1.\min, X_2.\min), \max(X_1.\max, X_2.\max)]$, we check whether $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ always holds. If so, X_1 dominates X_2; otherwise, X_1 does not dominate X_2. The pseudocode is shown as follows.

4.2.2 Naive algorithm with initial check for SSD

We present an initial check based on Lemma 4, which is similar to the initial check of FSD. If random variables X_1 and X_2 do not pass the check, X_1 does not dominate X_2 w.r.t. SSD. The proof of Lemma 4 and the pseudocode of Algorithm 5 with the initial check according to Lemma 4 follow.

Algorithm 4 NaiveSSD

```
Input:
    Random variables: $X_1$ and $X_2$;
Output:
    Dominance relationship between $X_1$ and $X_2$ w.r.t. SSD;
for each $a \in [\min(X_1.min, X_2.min), \max(X_i.max, X_i.max)]$ do
    if $\widehat{F}_{X_1}(a) < \widehat{F}_{X_2}(a)$ then
        return $X_1$ does not dominate $X_2$;
    end if
end for
return $X_1$ dominates $X_2$;
```

Algorithm 5 NaiveSSDInitialCheck

```
Input:
    Random variables $X_1$ and $X_2$;
Output:
    Dominance relationship between $X_1$ and $X_2$ w.r.t. SSD;
if $E(X_1) \leqslant E(X_2) \wedge X_1.min \leqslant X_2.min$ then
    Call the naive SSD algorithm on $X_1$ and $X_2$;
else if $E(X_2) \leqslant E(X_1) \wedge X_2.min \leqslant X_1.min$ then
    Call the naive SSD algorithm on $X_2$ and $X_1$;
else
    return no dominance.
end if
```

Lemma 4 *If $X_1 \succ_{ssd} X_2$ then $X_1.min \leqslant X_2.min$ and $E(X_1) \leqslant E(X_2)$.*

Proof We prove X_1.min $\leqslant$ X_2.min by contradiction. If X_1.min $>$ X_2.min, we have $\widehat{F}_{X_1}(X_1.\text{min}) = 0$ and $\widehat{F}_{X_2}(X_1.\text{min}) > 0$. Thus, we have $\widehat{F}_{X_2}(X_1.\text{min}) > \widehat{F}_{X_1}(X_1.\text{min})$. According to the definition of SSD, when $X_1 \succ_{\text{ssd}} X_2$, we have $\widehat{F}_{X_1}(X_1.\text{min}) \geqslant \widehat{F}_{X_2}(X_1.\text{min})$. This yields a contradiction, and we must have X_1.min $\leqslant$ X_2.min.

We apply Theorem 2 to prove $E(X_1) \leqslant E(X_2)$. Consider a non-increasing convex utility function $u(x) = -x$. According to Theorem 2, we have $E_{\text{U}}(X_1) \geqslant E_{\text{U}}(X_2)$. Since $u(x) = -x$, we have $E_{\text{U}}(X_1) = -E(X_1)$ and $E_{\text{U}}(X_2) = -E(X_2)$. Thus, $E(X_1) \leqslant E(X_2)$. □

4.2.3 Speedup algorithm for SSD

We propose a speedup algorithm for SSD checking between two random variables. We first utilize the initial check specified in Lemma 4 to filter cases where two random variables cannot dominate each other. We can then assume that X_1.min $\leqslant$ X_2.min, and the algorithm needs to check if X_1 dominates X_2 w.r.t. SSD, i.e., whether $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a), \forall a \in [X_2.\text{min}, \max(X_1.\text{max}, X_2.\text{max})]$.

The speedup algorithm also employs the two random variables' cdfs, i.e., $F_{X_1}(\cdot)$ and $F_{X_2}(\cdot)$, and the integrals of the cdfs, i.e., $\widehat{F}_{X_1}(\cdot)$ and $\widehat{F}_{X_2}(\cdot)$.

Algorithm *SpeedupSSD*(X_1, X_2, s, e) follows a divide-and-conquer approach. It takes as input two random variables X_1 and X_2 and a range $[s, e]$, and it returns a Boolean value indicating whether $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ for each $a \in [s, e]$.

Algorithm 6 SpeedupSSD (X_1, X_2, s, e)

```
Input:
    Random variables $X_1$ and $X_2$; a range $[s, e]$;
Output:
    True, if $X_1 \succ_{dcx} X_2$; False, otherwise;
if $\widehat{F}_{X_1}(s) < \widehat{F}_{X_2}(s) \vee \widehat{F}_{X_1}(e) < \widehat{F}_{X_2}(e)$ then
    return False
end if
if $\widehat{F}_{X_1}(s) \geqslant \widehat{F}_{X_2}(e)$ then
    return True; //acc. to Lemma 4
end if
$k_A \leftarrow F_{X_1}(s)$; $k_B \leftarrow F_{X_1}(e)$;
Construct line $l_A$ through $A$ with the slope $k_A$;
Construct line $l_B$ through $B$ with the slope $k_B$;
Point $E$ ← intersection of lines $l_A$ and $l_B$;
Point $M$ ← the point on curve of $\widehat{F}_{X_2}$ whose $x$-coordinate is $x_E$;
if $E$ is not below $M$ then
    return True; //acc. to Lemma 5
else
    $b1$ ← SpeedupSSD($X_1, X_2, s, x_E$);
    $b2$ ← SpeedupSSD($X_1, X_2, x_E, e$);
    return $b1 \wedge b2$;
end if
```

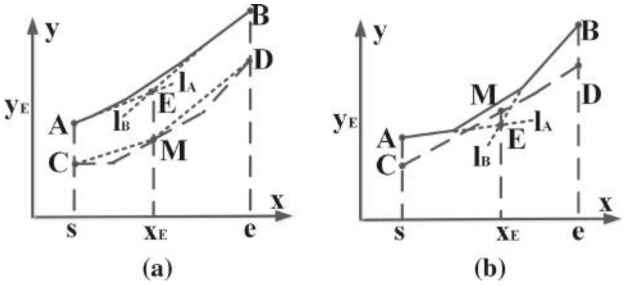

Fig. 5 Speedup algorithm for checking SSD. **a** Case 1 and **b** Case 2

The pseudocode is shown in Algorithm 6. First, we call *SpeedupSSD*$(X_1, X_2, X_2.\text{min}, \max(X_1.\text{max}, X_2.\text{max}))$.

As for SSD checking, we make use of the four important points, where points A and C are the left endpoints and points B and D are the right endpoints of the curves of $\widehat{F}_{X_1}$ and $\widehat{F}_{X_2}$ in the range $[s, e]$. Examples of the four points are shown in Fig. 5a, b.

At point A, we compute the corresponding slope k_A on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of s on X_1, i.e., $k_A = \widehat{F}'_{X_1}(s) = F_{X_1}(s)$. Then we construct a line l_A through A with slope k_A. Next, at point B, we compute the corresponding slope k_B on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of e on X_1, i.e., $k_B = \widehat{F}'_{X_1}(e) = F_{X_1}(e)$. Then we construct a line l_B through B with slope k_B. Assume that l_A and l_B intersect at point $E = (x_E, y_E)$ where the x- and y-coordinates of point E are x_E and y_E, respectively. We denote the point on curve of $\widehat{F}_{X_2}$ whose x-coordinate is x_E as $M = (x_E, \widehat{F}_{X_2}(x_E))$.

We proceed to consider two cases.

Case 1: *Point E is not below point M* (see Fig. 5a). According to Lemma 5, we have $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, e]$. Thus, we return *True*.

Lemma 5 *If* $y_E \geqslant \widehat{F}_{X_2}(x_E)$ *then* $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, e]$.

Proof Since $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$ are convex, the tangent line l_A at point A of $\widehat{F}_{X_1}(x)$ satisfies $\widehat{F}_{X_1}(x) \geqslant l_A(x), \forall x \in [s, x_E]$; and the chord line l_{CM} that connects points C and M of $\widehat{F}_{X_2}(x)$ satisfies $l_{CM}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, x_E]$.

Moreover, we have $l_A(s) = \widehat{F}_{X_1}(s) \geqslant \widehat{F}_{X_2}(s) = l_{CM}(s)$, meaning that the left end point of l_A is not below the left end point of l_{CM}; and $l_A(x_E) = y_E \geqslant \widehat{F}_{X_2}(x_E) = l_{CM}(x_E)$, meaning that the right end point of l_A is not below the right end point of l_{CM}. Thus, line l_A is not below line l_{CM} between s and x_E, i.e., $l_A(x) \geqslant l_{CM}(x), \forall x \in [s, x_E]$. Thus, we have $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, x_E]$.

We are able to derive the same conclusion for range $[x_E, e]$. Therefore, the lemma holds. □

Case 2: *Point E is below point M* (see Fig. 5b). In this case, we cannot give any guarantee, and the range $[s, e]$ is divided into two sub-ranges $[s, x_E]$ and $[x_E, e]$ based on the divided point, here x_E. We then recursively call *SpeedupSSD* on $[s, x_E]$ and $[x_E, e]$, respectively.

4.3 Checking SCSD

Similar to the case for checking SSD, the naive algorithm is based on the definition of SCSD. Lemma 6 enables an initial check for SCSD.

Lemma 6 *If* $X_1 \succ_{scsd} X_2$*, then* $X_1.\max \leqslant X_2.max$ *and* $E(X_1) \leqslant E(X_2)$.

Proof We prove $X_1.\max \leqslant X_2.\max$. Assuming that $X_1.\max > X_2.\max$ by contradiction, we have $\widetilde{F}_{X_1}(X_1.\max) = 0$ and $\widetilde{F}_{X_2}(X_1.\max) > 0$. Thus, $\widetilde{F}_{X_2}(X_1.\max) < \widetilde{F}_{X_1}(X_1.\max)$. According to the definition of SCSD, when $X_1 \succ_{\text{scsd}} X_2$, we have $\widetilde{F}_{X_1}(X_1.\max) \geqslant \widetilde{F}_{X_2}(X_1.\max)$. This yields a contradiction. Thus, the assumption is invalid, and $X_1.\max \leqslant X_2.\max$.

We apply Theorem 3 to prove $E(X_1) \leqslant E(X_2)$. Consider a non-increasing convex utility function $u(x) = -x$. According to Theorem 3, we have $E_{\text{U}}(X_1) \geqslant E_{\text{U}}(X_2)$. Since $u(x) = -x$, we have $E_{\text{U}}(X_1) = -E(X_1)$ and $E_{\text{U}}(X_2) = -E(X_2)$. Thus, we have $E(X_1) \leqslant E(X_2)$. □

Finally, we present a speedup algorithm for checking SCSD. The speedup algorithm for check SCSD is similar to the speedup algorithm for checking SSD. The only difference is that we need to check if point E is not above M, while when checking for SSD, we check if point E is not below M. We give an intuitive idea of how to perform speedup algorithm for SCSD in Fig. 6.

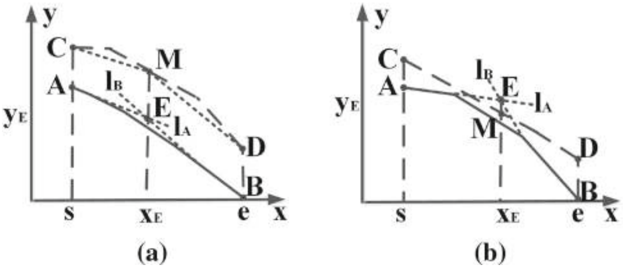

Fig. 6 Speedup algorithm for checking SCSD. **a** Case 1 and **b** Case 2

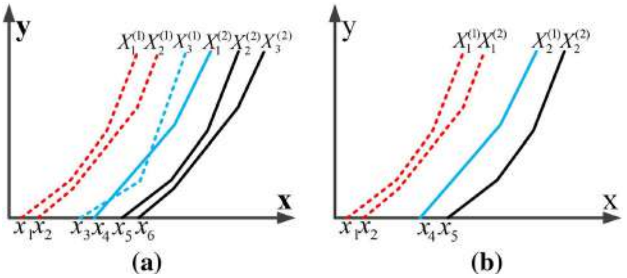

Fig. 7 Intuition for the grouping strategy. **a** Naive and **b** grouping

5 Finding temporal dominance

We proceed to provide efficient algorithms for checking temporal dominance among UTSs. We first solve the case of checking temporal dominance between two UTSs, and then the case of checking temporal dominance among multiple UTSs.

5.1 Naive algorithm

Given two uncertain time series $T_1 = \langle X_1^{(1)}, X_1^{(2)}, \ldots, X_1^{(N)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, \ldots, X_2^{(N)} \rangle$, we would like to find the temporal dominance between T_1 and T_2. In particular, during each interval j, we want to determine the dominance relationship between $X_1^{(j)}$ and $X_2^{(j)}$ w.r.t. a given order: FSD, SSD, or SCSD.

A naive algorithm checks, for each interval j, the relationship between $X_1^{(j)}$ and $X_2^{(j)}$ using the proposed speedup algorithms. For example, consider two UTSs, where each has 3 intervals $T_1 = \langle X_1^{(1)}, X_1^{(2)}, X_1^{(3)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, X_2^{(3)} \rangle$, and assume that we consider FSD. The integral cdfs of the random variables in the UTSs are shown in Fig. 7. The naive algorithm needs to determine the dominance relationship between three pairs of random variables.

This can be very inefficient, especially when long UTSs have many intervals. To contend better with long UTSs, we group the random variables in the UTSs and compare groups. If one group dominates another, any random variable in the former group dominates every random variable in the latter group. This grouping can significantly reduce the cost of dominance checking.

For example, we form G_1 from $X_1^{(1)}$ and $X_1^{(2)}$, and form G_2 from $X_2^{(1)}$ and $X_2^{(2)}$, as shown in Fig. 7b. Since the lower boundary curve of G_1 is always above the upper boundary curve of G_2, the random variables in G_1 dominates the random variables in G_2. Thus, we determine the dominance relationships for two intervals, by comparing one group pair. Likewise, when groups have multiple random variables, if one group dominates another, we are able to determine dominance relationships for multiple intervals, thus improving efficiency. On the other hand, large groups are less likely to dominate each other even if there is dominance between pairs of variables in them. Thus, we proceed to design a general grouping framework that is able to identify groups with appropriate sizes.

5.2 A general grouping framework

We choose the *expectations* of random variables as the *grouping criterion*. This is because $E(X_1) \leqslant E(X_2)$ is a necessary condition for $X_1 \succ_{\text{fsd}} X_2$, $X_1 \succ_{\text{ssd}} X_2$, and $X_1 \succ_{\text{scsd}} X_2$ according to Lemmas 1, 4, and 6.

Analogous to the ideas of the initial check covered in Sect. 4, we propose a grouping strategy to produce groups such that for each group, the random variables from a UTS, say T_1, have smaller expectations and the random variables from the other UTS, say T_2, have larger expectations. This way, the random variables from T_2 with larger expectations cannot dominate the random variables in T_1 with smaller expectations, and thus we only need to check if the random variables in T_1 dominate the random variables in T_2. The resulting strategy is shown in Algorithm 7. We illustrate the strategy using the example in Table 5 that lists the expectations of two UTSs with 8 intervals.

We choose the UTS with the smaller average expectation as the *master* and call the other UTS the *slave*. In the example, T_2 is the master and T_1 is the slave.

The algorithm identifies appropriate groups iteratively. In each iteration, it identifies the smallest expectation $E_{\min}$ among the expectations of non-grouped intervals in the master. In the first iteration in the example, $E_{\min} = 40$ because 40 is the smallest expectation in T_2. The algorithms stops when all intervals are grouped (line 4). In each iteration, it distinguishes between two cases.

Case 1: The slave has intervals with expectations that are smaller than $E_{\min}$. In this case, these intervals form a group (lines 7–15).

In the example, the first iteration belongs to case 1 because intervals 1, 2, 5, and 6 in T_1 have expectations 20, 39, 30, and 35, which are all smaller than $E_{\min} = 40$. Thus $G_1 = \{1, 2, 5, 6\}$ is the first group.

Algorithm 7 GroupingBasedOnExpectations(T_1, T_2)

```
Input:
   UTSs: $T_1$ and $T_2$.
Output:
   A sequence of optimal random variables, $q$;
1: Identify the master UTS $T_m$ and the slave UTS $T_s$;
2: Make an empty sequence $q \leftarrow \emptyset$;
3: Non-grouped interval set $NGIS \leftarrow$ all intervals;
4: while $NGIS \neq \emptyset$ do
5:    $E_{min} \leftarrow$ the smallest expectation in $NGIS$ in $T_m$;
6:    $E'_{min} \leftarrow$ the smallest expectation in $NGIS$ in $T_s$;
7:    if $E'_{min} < E_{min}$ then
8:       Make an empty group $G$;
9:       for each interval $j \in NGIS$ do
10:          if the expectation of the $j$th interval in $T_s$ is no larger than
             $E_{min}$ then
11:             $G \leftarrow G \cup \{j\}$; Remove $j$ from $NGIS$;
12:          end if
13:       end for
14:       $I \leftarrow$ GroupCheck($T_s$, $T_m$, $G$, min($G$), max($G$));
15:       Update($q$, $I$, $T_s$, $T_m$);
16:    else
17:       $E_{min} \leftarrow E'_{min}$;
18:       Make a new empty group $G$;
19:       for each interval $j \in NGIS$ do
20:          if the expectation of the $j$th interval in $T_m$ is no larger than
             $E_{min}$ then
21:             $G \leftarrow G \cup \{j\}$; Remove $j$ from $NGIS$;
22:          end if
23:       end for
24:       $I \leftarrow$ GroupCheck($T_m$, $T_s$, $G$, min($G$), max($G$));
25:       Update($q$, $I$, $T_m$, $T_s$);
26:    end if
27: end while
28: return $q$;
```

Table 5 Expectations for UTSs T_1 and T_2

Interval j	1	2	3	4	5	6	7	8
$E(X_1^{(j)})$ in T_1	20	39	60	80	30	35	55	75
$E(X_2^{(j)})$ in T_2	40	40	45	54	40	40	50	54

The intuition is that when considering the random variables in G_1, the random variables from T_1 may dominate the random variables from T_2, while the random variables in T_2 cannot dominate the random variables from T_1. Then, we only need to further check if the random variables in T_1 dominate the random variables in T_2 by calling algorithm *GroupCheck*$(T_s, T_m, G, \min(G), \max(G))$ (line 14), where $\min(G)$ and $\max(G)$ return the minimum and maximum values of the random variables in the intervals that belong to group G from both master T_m and slave T_s, respectively. Algorithm *GroupCheck* returns the intervals in G for which the corresponding RVs in T_s dominate those in T_m. Thus, for the removed intervals in G, the RVs in T_s and T_m do not dominate each other. Based on this, a helper function *update* is called to update the result q.

Case 2: The slave has no intervals with expectations that are smaller than $E_{\min}$. In this case, we update $E_{\min}$ to be the smallest expectation among the expectations in the non-grouped intervals in the slave. Then, in the master, the intervals whose expectations are no larger than the updated $E_{\min}$ are grouped (lines 16–26).

In the example, the second iteration belongs to case 2. After the first iteration, the non-grouped intervals are 3, 4, 7, and 8. The minimum expectation among non-grouped intervals in the master is $E_{\min}$ with value 45. In the slave, intervals 3, 4, 7, and 8 have expectations 60, 80, 55, and 75, all of which exceed 45. We then set $E_{\min}$ to 55, i.e., the minimum expectation of non-grouped intervals in the slave. Since the expectations of the non-grouped intervals in the master are 45, 54, 50, 54, which are all no larger than 55, we form the group $G_2 = \{3, 4, 7, 8\}$.

The intuition is that when considering the random variables in G_2, the random variables from the master may dominate the random variables from the slave, while the random variables in the slave cannot dominate random variables in the master. This is because the random variables in T_2 have smaller expectations than the expectations of the random variables in T_1. We then call *GroupCheck*$(T_m, T_s, G, \min(G), \max(G))$ (line 24).

5.3 Dominance checking for groups

We provide a detailed discussion only for the case of SSD checking. Since checking of FSD and SCSD for groups can be done in a similar manner, we omit the details.

Dominance checking for groups is achieved by calling function *GroupCheck*(T_1, T_2, G, min, max), shown in Algorithm 8. In the following discussion, we only consider random variables during intervals in G. Recall that when groups are formed using Algorithm 7, we ensure that T_1's random variables have smaller expectations, meaning that T_2's random variables cannot dominate T_1's random variables. Thus, we only need to check whether T_1's random variables dominate T_2's random variables.

We use $G = \{1, 2, 5, 6\}$, i.e., the first group found in Sect. 5.2, as an example. The integrals of the cdfs of the random variables in G from T_1 and T_2 are shown in Fig. 8a.

5.3.1 Boundary condition

We start by checking a boundary condition (line 2 in Algorithm 8). Consider the random variables (RVs) in G. If the RVs in T_1 dominate the RVs in T_2, the curves of the integral of the cdfs of the RVs in T_1 should appear above the corresponding curves for T_2. The boundary condition checks if this is true on the left and right boundaries. If this is not true on either boundary, T_1's curves cannot always appear above T_2's curves, and thus we cannot guarantee that T_1's RVs dominate T_2's RVs. Thus, we need to partition G into sub-groups and further check for each sub-group whether T_1's curves can always appear above T_2's curves.

Algorithm 8 GroupCheck(T_1, T_2, G, min, max)

```
Input:
  UTSs: $T_1$ and $T_2$; Group $G$; Double: $min$ and $max$;
Output:
  Interval set $IS$ of intervals where $T_1$ dominates $T_2$;
1: $IS \leftarrow \emptyset$;
2: if $\neg$ BoundaryCondition($T_1, T_2, G, min, max$) then
3:     Make sub-groups from $G$;
4:     for each sub-group $sg$ do
5:         $IS \leftarrow IS \cup$ GroupCheck($T_1, T_2, sg, min, max$);
6:     end for
7: else
8:     $s \leftarrow min(T_2, G)$; $e \leftarrow max(T_1, G)$;
9:     $k_A \leftarrow min(T_1, G, s)$; $k_B \leftarrow max(T_1, G, e)$;
10:    Construct line $l_A$ from $A$ with the slope $k_A$;
11:    Construct line $l_B$ from $B$ with the slope $k_B$;
12:    Point $E \leftarrow$ intersection of lines $l_A$ and $l_B$;
13:    Point $M \leftarrow$ the point on the top-most curve among the curves
       of $T_2$'s random variables whose $x$-coordinate is $x_E$;
14:    if $E$ is not below $M$ then
15:        $IS \leftarrow G$;
16:    else
17:        $G' \leftarrow$ GroupCheck($T_1, T_2, G, s, i$);
18:        $IS \leftarrow$ GroupCheck($T_1, T_2, G', i, e$);
19:    end if
20: end if
21: return $IS$;
```

We show how the boundary condition works for the left boundary. It works similarly for the right boundary. We first identify the left boundary lb as the smallest minimum values of the RVs in T_2. For each interval $i \in G$, we consider two copies, one copy for the random variable in interval i in T_1, represented as a square, and one copy for the random variable in interval i in T_2, represented as a circle—see Fig. 8b.

Next, we associate with each interval copy a value that equals its random variable's integral of cdf at lb. In particular, for interval copy $\boxed{5}$, which corresponds to random variable $X_1^{(5)}$ from T_1, the associated value is $\widehat{F}_{X_1^{(5)}}(lb)$. For interval copy $\textcircled{1}$, which corresponds to random variable $X_2^{(1)}$ from T_2, and the associated value is $\widehat{F}_{X_2^{(1)}}(lb)$. Next, we order the interval copies according to their associated values. Figure 8b shows such ordered interval copies. Specifically, since RVs in the 1st and 5th intervals from T_1 have nonzero values y_1 and y_2, and all the remaining RVs have 0s, $\boxed{1}$ and $\boxed{5}$ appear at the top, and the remaining interval copies follow.

Next, we identify the interval copies from T_2 that appear at the top. In our example, it is interval 1 from T_2, i.e., $\textcircled{1}$. All interval copies from T_1 that appear above $\textcircled{1}$ form a sub-group. In the example, intervals 1 and 5 form a sub-group. This is so because, on the left boundary and during intervals 1 and 5, the RVs from T_1 appear above the RVs from T_2, meaning that it is possible that the RVs during intervals 1 and

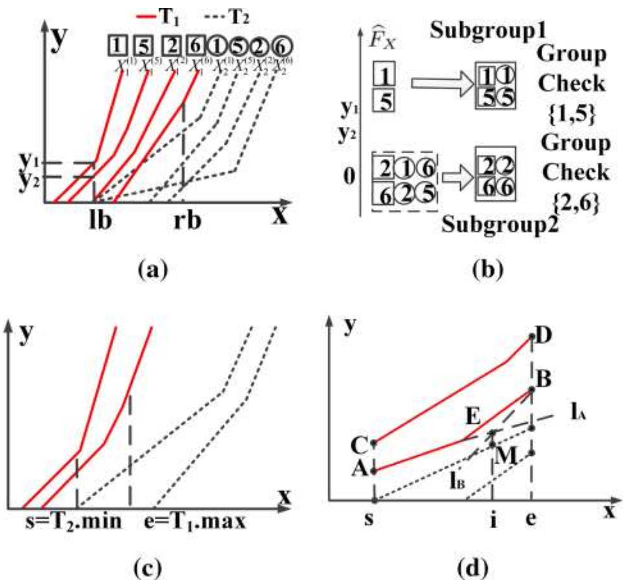

Fig. 8 Group-based dominance checking. **a** $\widehat{F}_X(\cdot)$ of RVs in G, **b** boundary condition, **c** sub-group1 = {1, 5} and **d** group checking

5 from T_1 dominates those from T_2. The remaining intervals form another sub-group.

If one of the sub-groups is empty, this means that it is not necessary to make further sub-groups at boundaries and that group G passes the boundary condition checking. Otherwise, we check dominance for each sub-group by recursively calling *GroupCheck* on each sub-group (lines 3–5 in Algorithm 8).

5.3.2 Group check with lower and upper bounds

We follow an idea similar to that used in the speedup algorithm for checking SSD. We first identify s as the smallest minimum value of the RVs in T_2 and e as the largest maximum value of the RVs in T_1 (line 8), see Fig. 8c. In order to better explain the idea of group check, we give several definitions and lemmas.

Definition 4 *Lower Bound.* We use H to denote the integral cdfs of a set of RVs. Given a range $[s, e]$ on the x-axis, the minimum values of H at s and e are denoted as $H_{s,\min} = \min_{\widehat{F}_i\in H}\widehat{F}_i(s)$ and $H_{e,\min} = \min_{\widehat{F}_i\in H}\widehat{F}_i(e)$, respectively. The minimum value of the derivatives of H at s is denoted as $H'_{s,\min} = \min_{\widehat{F}_i\in H}\widehat{F}'_i(s)$, and the maximum value of the derivatives of H at e is denoted as $H'_{e,\max} = \max_{\widehat{F}_i\in H}\widehat{F}'_i(s)$. The lower bound in range $[s, e]$, denoted as $lb(H, s, e)$, consists of the following two line segments.

$$\begin{cases} y = H'_{s,\min}\cdot(x-s)+H_{s,\min}, & \text{when } x\in[s,i] \\ y = H'_{e,\max}\cdot(x-e)+H_{e,\min}, & \text{when } x\in(i,e] \end{cases}$$

Here, i is the x-coordinate of the intersection of the two line segments.

For example, consider the RVs in T_1, whose integral cdfs are shown in Fig. 8d. Points A and B hold the minimum values of the integral cdfs of the RVs at s and e, respectively. This means that point A's y-coordinate is $H_{s,\min}$ and point B's y-coordinate is $H_{e,\min}$. The slopes of lines l_A and l_B equal $H'_{s,\min}$ and $H'_{e,\max}$, respectively. Lines l_A and l_B intersect at point E, whose x-coordinate is i. Thus, the lower bound of the RVs in T_1 consists of line segment AE when $x\in[s,i]$ and line segment EB when $x\in[i,e]$.

Lemma 7 *For any value $x\in[s,e]$ on the x-axis, and any integral cdf $\widehat{F}_i\in H$, we have $lb(H,s,e)(x)\leqslant\widehat{F}_i(x)$, where $lb(H, s, e)(x)$ returns the y-coordinate of the point that is on the lower bound and whose x-coordinate is x.*

Proof We use proof by contradiction. We start proving on the left range $[s, i]$. Assume that there exists a value $x'\in[s,i]$ and an integral cdf $\widehat{F}_i\in H$ such that $\widehat{F}_i(x') < lb(H,s,e)(x')$. If so, the slope of the chord line that connects point $(s, \widehat{F}_i(s))$ and point $(x', \widehat{F}_i(x'))$ is smaller than $H'_{s,\min}$. However, since any integral cdf is convex, the slope of the chord line of any integral cdf in H on range $[s, i]$ should be no smaller than $H'_{s,\min}$. This yields a contradiction. The proof for the right range $[i, e]$ is similar. Thus, the lemma holds. □

Definition 5 *Upper Bound.* We use H to denote the integral cdfs of a set of RVs. Given a range $[s, e]$ on the x-axis, the maximum values of H at s and e are denoted as $H_{s,\max} = \max_{\widehat{F}_i\in H}\widehat{F}_i(s)$ and $H_{e,\max} = \max_{\widehat{F}_i\in H}\widehat{F}_i(e)$, respectively. For any $d\in[s,e]$, we define $H_{d,\max}$ in a similar way. Given $d\in[s,e]$, the upper bound of H, denoted as $ub(H, s, e, d)$, consists of the following two line segments.

$$\begin{cases} y = \frac{H_{d,}\text{max}-H_{s,}\text{max}}{d-s}\cdot(x-s)+H_{s,\max}, & \text{when } x\in[s,d] \\ y = \frac{H_{e,}\text{max}-H_{d,}\text{max}}{e-d}\cdot(x-e)+H_{e,\max}, & \text{when } x\in(d,e] \end{cases}$$

Lemma 8 *For any value $x\in[s,e]$ on the x-axis, and any integral cdf $\widehat{F}_i\in H$, we have $ub(H,s,e,d)(x)\geqslant\widehat{F}_i(x)$, where $ub(H, s, e, d)(x)$ returns the y-coordinate of the point that is on the upper bound and whose x-coordinate is x.*

Proof We use proof by contradiction. We consider the left range $[s, d]$. Assume $\exists x'\in[s,d]$ and $\exists\widehat{F}_i\in H$ such that $\widehat{F}_i(x') > ub(H,s,e,d)(x')$. If so, the slope k of the chord line that connects point $(s, \widehat{F}_i(s))$ and point $(x', \widehat{F}_i(x'))$ exceeds the slope k' of the upper bound $ub(H, s, e, d)$ on range $[s, d]$, i.e., $k > k'$.

Further, since any cdf is a non-decreasing function, we have $\widehat{F}'_i(x'')\geqslant\widehat{F}'_i(x')$ if $x'' > x'$. Since $d > x'$, this leads to $\widehat{F}'_i(d)\geqslant\widehat{F}'_i(x') = k > k'$. Thus, we have $\widehat{F}_i(d) > ub(H,s,e,d)(d) = H_{d,\max}$. This yields a contradiction. The proof for the right range $[d, e]$ is similar. Therefore, the lemma holds. □

With Lemmas 7 and 8, it is straightforward to obtain the following lemma to decide the dominance relationships between two sets of RVs.

Lemma 9 *Given a master set of RVs with integral cdfs H^{master}, a slave set of RVs with integral cdfs H^{slave}, and a range $[s, e]$, if H^{master} and H^{slave} meet the following three conditions then $\forall \widehat{F_k} \in H^{master}, \forall \widehat{F_j} \in H^{slave}, \forall x \in [s, e]$, we have $\widehat{F_k}(x) \geqslant \widehat{F_j}(x)$.*

(1) $H^{master}_{s,min} \geqslant H^{slave}_{s,max}$;
(2) $H^{master}_{e,min} \geqslant H^{slave}_{e,max}$;
(3) $lb(H^{master}, s, e)(i) \geqslant ub(H^{slave}, s, e, i)(i)$.

Proof Conditions (1) and (3) guarantee that, in range $[s, i]$, the lower bound of H^{master} is above the upper bound of H^{slave}. Similarly, conditions (2) and (3) guarantee that, in range $[i, d]$, the lower bound of H^{master} is above the upper bound of H^{slave}. □

Algorithm 8 proceeds according to Lemma 9. We identify the important points A and B, where A and B are the *lowest* points of the integral cdfs of RVs in T_1 whose x-coordinate are s and e, respectively, as shown in Fig. 8d. We identify two slopes k_A and k_B, where k_A is the *smallest* slope of the integral cdfs of RVs in T_1 when the x-coordinate is s and k_B is the *largest* slope of the integral cdfs of RVs in T_1 when the x-coordinate is e. We construct line l_A through point A with slope k_A and line l_B through point B with slope k_B (lines 9–11). We then compute the intersection E of lines l_A and l_B. If E does not appear below the upper bound of T_2 then all RVs in G from T_1 dominate all RVs in G from T_2 (lines 12–15). Otherwise, we check sub-ranges (s, i) and (i, e) (lines 16–18).

5.4 Checking multiple UTSs

Based on the proposed grouping strategy, we have an efficient method to check the temporal dominance between two UTSs. However, a UTS collection TS may contain multiple UTSs. We propose two alternative methods to check the temporal dominance among all UTSs in a TS. The first simply checks every UTS pair using the efficient algorithm based on the grouping strategy. The second employs a merge-sort-like procedure to compare UTS pairs while employing the intermediate comparison results to construct the final result.

For example, consider $TS = \{T_1, T_2, T_3, T_4\}$. The first method checks the dominance relationships between pairs (T_1, T_2), (T_1, T_3), (T_1, T_4), (T_2, T_3), (T_2, T_4), and (T_3, T_4). The second method first checks (T_1, T_2) and (T_3, T_4) and then uses the results we consider to construct the final result.

5.5 User defined constraints

Users may specify additional travel time constraints. For example, a user may only be interested in paths with an arrival times that are no later than 6 p.m. or that offer a possibility of arriving before 4 p.m.

We use left constraint (LC) and right constraint (RC) to denote the shortest and longest travel times that a user is interested in. For example, assume that the current time is 2 p.m. and that a user specifies the following constraint: "the arrival time must not exceed 6 p.m. or there must be a possibility of arriving before 4 p.m." We can then set LC to 120 min and RC to 240 min.

Based on LC and RC, we can prune random variables, before checking the stochastic dominance relationships. Specifically, using LC, we can prune random variable X if the minimum value of X exceeds LC. In other words, if $F_X(LC) \leq 0$, we prune X. Similarly, based on RC, we can prune random variable X if the maximum value of X exceeds RC. In other words, if $F_X(RC) < 1$, we prune X. We only consider a random variable X if and only if X satisfies both the LC and the RC constraints.

Algorithm 9 shows how we use the two constraints. Given constraints, T_1 and T_2 and time interval j, (1) if the jth random variable in T_1 does not meet one of the constraints and the jth random variable in T_2 meets both constraints, then T_2 is the optimal choice in time interval j (lines 4–6); (2) if the jth random variable in T_2 does not meet one of the constraints and the jth random variable in T_1 meets both constraints, then T_1 is the optimal choice in time interval j (lines 7–9); (3) if the jth random variables in both UTSs

Algorithm 9 UserDefinedConstraintsCheck(T_1, T_2, LC, RC)

```
Input:
    UTSs: $T_1$ and $T_2$, $LC$, $RC$.
Output:
    A sequence of optimal random variables, $q$, and remaining intervals $I$;
1:  Make an empty sequence $q \leftarrow \emptyset$;
2:  Non-grouped interval set $I \leftarrow$ all intervals;
3:  for each interval $j \in I$ do
4:      if ($T_1[j].min > LC$ or $T_1[j].max > RC$) and ($T_2[j].min \leq LC$ and $T_2[j].max \leq RC$) then
5:          Remove $j$ from $I$;
6:          Update($q, j, T_2, T_1$);
7:      else if ($T_1[j].min \leq LC$ and $T_1[j].max \leq RC$) and ($T_2[j].min > LC$ or $T_2[j].max > RC$) then
8:          Remove $j$ from $I$;
9:          Update($q, j, T_1, T_2$);
10:     else if ($T_1[j].min > LC$ or $T_1[j].max > RC$) and ($T_2[j].min > LC$ or $T_2[j].max > RC$) then
11:         Remove $j$ from $I$;
12:     end if
13: end for
14: return $q, I$;
```

Table 6 Number of UTS Collections

$\vert\mathcal{TS}\vert$	2	3	4	5	6
Num of collections	12,461	1266	164	20	4

do not meet the constraints, time interval j has no optimal choice and is removed (lines 10–11).

To incorporate the user-defined constraints, Algorithm 9 must be called to filter random variables that do not satisfy the constraints. Specifically, Algorithm 9 needs to be called after line 3 in Algorithm 7 to update q and $NGIS$, respectively. Afterwards, Algorithm 7 continues to check the remaining random variables that satisfy the constraints.

6 Empirical study

6.1 Experimental setup

We consider two different collections of uncertain time series.

6.1.1 Real UTSs (RU)

We use a large GPS data set of more than 180 million GPS records collected in Denmark from January 2007 to December 2008. We eliminate GPS records with unreasonable speeds, i.e., more than 200 km/h, since the speed limit on highways in Denmark is 130 km/h.

Next, we align GPS records with the road network to obtain trajectories using a well-known map-matching algorithm [12]. Given an origin and a destination (OD) pair, we first count the number of trajectories between the OD pair. Then we select the top 15,000 OD pairs with the largest numbers of trajectories. For each chosen OD pair, we consider all paths that have been used by trajectories that connect the pair.

We derive UTSs for the edges in the road network using the method outlined in Sect. 2. Then, for each considered path, we construct a UTS using the UTSs of the edges in the path using an existing method [7,16]. Note that the procedure of constructing UTSs for paths can be done in parallel, e.g., using MapReduce [22]

This way, we obtain a UTS collection for each OD pair. As OD pairs are connected by between 2 and 6 paths, we obtain UTS collections with cardinalities that vary from 2 to 6, as shown in Table 7 and the corresponding number of UTS collections is shown in Table 6.

6.1.2 Synthetic UTSs (SU)

Following a common practice from studies of UTSs [23–25], we also construct UTSs from deterministic time series (DTSs). In particular, we use a publicly available DTS data set.[3] In a DTS, each interval is associated with a deterministic value c rather than a random variable. To construct a UTS from a DTS, we construct a random variable based on the deterministic value c for each interval. Specifically, generate three distributions—normal, uniform, and exponential—and use c as their mean. The standard deviation is chosen as τ times the standard deviation of all the deterministic values in a DTS, where τ varies from 0.1 to 2. This procedure follows a recommendation from a recent UTS benchmark [23].

Table 7 Parameter Settings

Parameter	Values
b (10^2)	1, 2, 3, 4, 5, 6, 7, 8, **9**, 10
$\vert\mathcal{TS}\vert$	2, 3, 4, **5**, 6
τ	0.1, 0.5, 1, **2**
Distributions	**Normal (n)**, exponential (e), uniform (u)

6.1.3 Data cleaning

In both datasets, we smooth the probability mass function (pmf) of each random variable. In particular, we set $pmf(x) = 0$ if $pmf(x) < \epsilon$, and we set ϵ to 1×10^{-8} in the experiments; and then we normalize each pmf so that $\sum pmf(x) = 1$. By doing this, we remove insignificant biases.

6.1.4 Parameters

For both RU and SU, we use equi-width histograms to represent the random variables. We vary the number b of bins in a histogram from 100 to 1000. We also vary the cardinality $|\mathcal{TS}|$ of a UTS collection, parameter τ, and the distribution of the synthetic UTSs according to Table 7, where the default values are in bold.

Since SSD and SCSD are defined in a similar way and also due to the space limitation, we only report findings for SSD.

Note that temporal dominance queries only require users to provide a stochastic dominance parameter x that indicates fsd, ssd, or scsd—users need not provide specific utility functions.

6.1.5 Implementation details

All algorithms are implemented in Python. We conduct experiments on a server with a 64-core AMD Opteron(tm) 2.24 GHZ CPU, 528 GB main memory under Ubuntu Linux.

[3] http://academic.udayton.edu/kissock/http/Weather/.

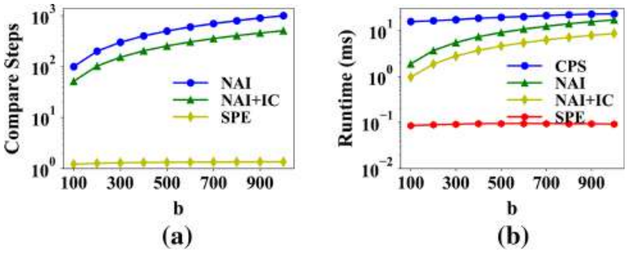

Fig. 9 Impact of b, SSD, RU. **a** Comparison steps and **b** runtime

6.2 Efficiency

6.2.1 Checking stochastic dominance

We first study the performance of checking stochastic dominance between two random variables. We consider four algorithms: (1) *NAI* the naive algorithm according to the definition of the corresponding order. (2) *NAI + IC* the naive algorithm with initial checks. This also corresponds to a state-of-the-art method [26,27], where a detailed discussion is included in Sect. 7. (3) *SPE* the speedup algorithm. (4) *CPS* a method that represents different orders of stochastic dominance as constraints in linear programming and solves the resulting linear problem using CPLEX,[4] a state-of-the-art optimization software package that is widely used in statistics and operations research [28].

We consider two measurements—the number of comparison steps and runtime. For *NAI* and *NAI + IC*, the number of comparison steps corresponds to the number of the comparisons between two random variables' cdfs or cdf integrals. For *SPE*, the number of comparison steps correspond to the number of the recursive calls in Algorithms 3 and 6, because they compare two random variables' cdfs or integrals of cdfs in each recursive call.

For brevity, in this set of experiments, we only report results for SSD and omit results for FSD if they are similar to those for SSD.

Impact of b We first evaluate the scalability of the different algorithms w.r.t. the number of histogram bins, with results shown in Figs. 9 (RU) and 10 (SU). For both data sets, the number of comparison steps and the runtime of *NAI* and *NAI + IC* increase almost linearly with b. However, the speedup algorithm *SPE* has stable performance and is insensitive to b. Further, *SPE* shows substantial improvement over *NAI* and *NAI + IC*.

Next, we consider the method *CPS*. As *CPS* works as a black box, we are unable to count the number of comparison steps and only measure runtime. Figures 9b and 10b show that although *CPS* is also insensitive to b, the runtime is even worse than that of *NAI*. This is because *CPS* has an initialization overhead for constructing its linear programming model.

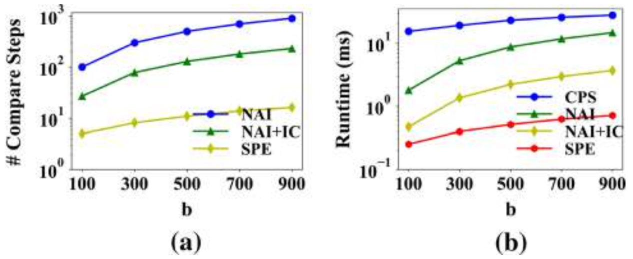

Fig. 10 Impact of b, SSD, SU. **a** Comparison steps and **b** runtime

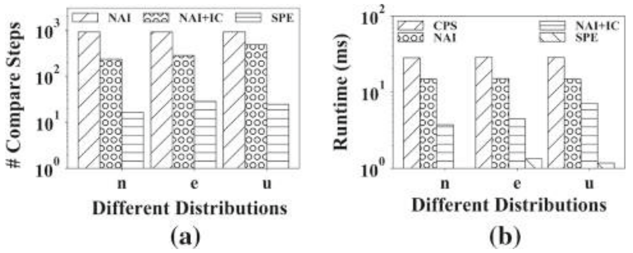

Fig. 11 Impact of distributions, SSD, SU. **a** Comparison steps and **b** runtime

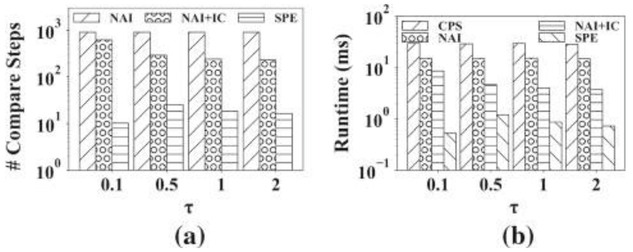

Fig. 12 Impact of τ, SSD, SU. **a** Comparison steps and **b** runtime

Impact of distributions Figure 11 reports the performance of the different methods for different distributions, where n, e, and u denote normal, exponential, and uniform distributions. We see that *SPE* significantly outperforms other methods in all settings. This experiment implies that *SPE* is robust to different distributions.

Impact of τ Next, we consider the impact of varying standard deviations. Figure 12 shows the results for normal distributions. As the standard deviation increases, the performance of *NAI + IC* improves slightly. This is because although the range that needs to be checked increases with a larger standard deviation, *NAI + IC* may find an inconsistent point early and may thus terminate in a few steps, which results in less runtime. The runtime for *SPE* increases when τ increases from 0.1 to 0.5 and then decreases when τ increases from 0.5 to 2. The reason is twofold. First, when τ is very

[4] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

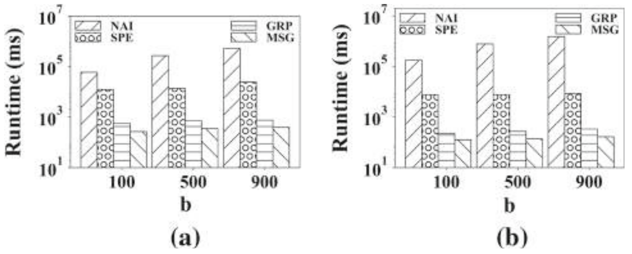

Fig. 13 Impact of b, UTSs, RU. **a** FSD and **b** SSD

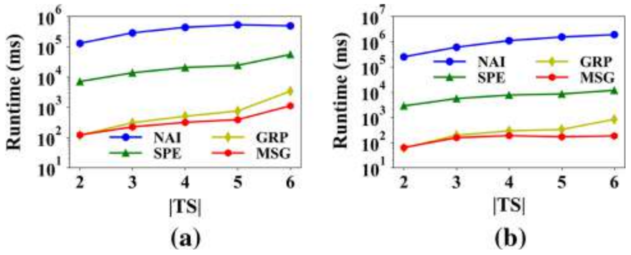

Fig. 14 Impact of $|TS|$, UTSs, RU. **a** FSD and **b** SSD

small, the compared cdf integrals may be separated by substantial gaps that yield a clear dominance relationship, which results in less runtime. Second, when τ becomes larger, e.g., when $\tau > 1$, the gaps become smaller, and the cdf integrals are increasingly likely to intersect, resulting in situations where random variables cannot dominate each other. Therefore, when τ is either very small or very large, it takes less time to check dominance relationships.

6.2.2 Temporal dominance queries in UTSs

We evaluate the performance of processing temporal dominance queries on UTS collections. We consider four methods for temporal dominance queries. Neither *NAI* nor *SPE* use the grouping strategy, but rather use the naive algorithm with initial checking and the speedup algorithm to check the dominance relationships for each interval. Both *GRP* and *MSG* use the grouping strategy, and *GRP* does not use the merge-sort-like procedure that is used by *MSG*.

Impact of b Figure 13 shows the runtime when varying b on RU. The runtime of *NAI* increases almost linearly with the increase in b, while the other three methods, which are all based on the proposed speedup algorithm, only increases slightly. This occurs because the speedup algorithm for checking FSD and SSD between two random variables is insensitive to b. Figure 13 also shows that the proposed grouping strategy is effective—the two methods that use the grouping strategy, i.e., *GRP* and *MSG*, outperform the other two methods that do not use the grouping strategy, i.e., *NAI* and *SPE*.

Impact of $|TS|$ Figure 14 shows the runtime of the different methods when varying the UTS collection cardinality $|TS|$ on RU. The results show that as the cardinality increases, the methods using the grouping strategy achieve more significant runtime improvements. Further, the use of a merge-sort-like procedure enables *MSG* to achieve better runtime than *GRP*, especially when cardinality is large.

Impact of l We show the impact of the length l of UTSs on RU in Fig. 15. As length l increases, the runtimes of all methods increase. However, runtimes of the two methods using the grouping strategy, i.e., *GRP* and *MSG*, only increase slightly, as they are able to group similar random variables together and identify dominance relationships between random variable groups. When the time series are long, there are more random variables that can be grouped. For all values of l, *GRP* and *MSG* are significantly faster than the other two methods that do not use the grouping strategy, on both FSD and SSD.

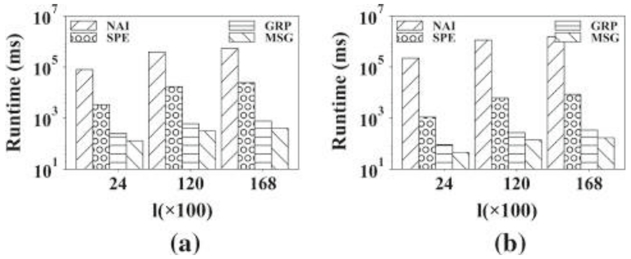

Fig. 15 Impact of length, UTSs, RU. **a** FSD and **b** SSD

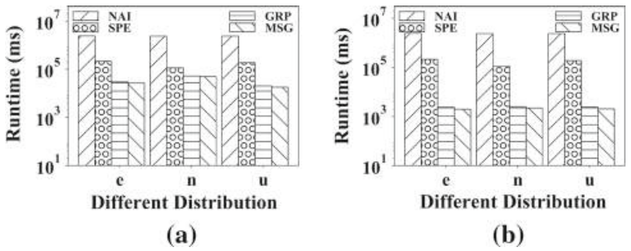

Fig. 16 Impact of distribution, UTSs, SU. **a** FSD and **b** SSD

Impact of distributions Figure 16 shows the effect of different distributions on SU. *GRP* and *MSG*, which use the grouping strategy, outperform the methods without grouping strategy for all distributions. This indicates that the proposed grouping strategy is robust to varying distributions.

Impact of τ Figure 17 shows the impact of varying standard variations when using normal distributions on SU. On FSD, the performance of the grouping strategy degrades slightly with larger standard deviations because such standard deviations have the effect that the initially formed groups based on means will be split into many sub-groups later on, which adversely affects the efficiency. On SSD, the performance of

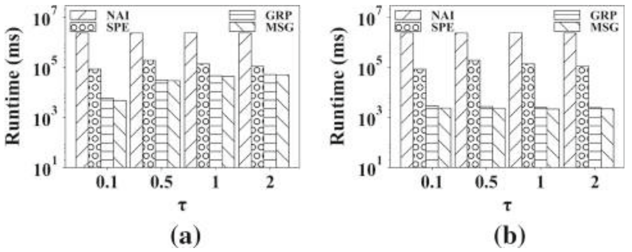

Fig. 17 Impact of τ, UTSs, Normal Dist., SU. **a** FSD and **b** SSD

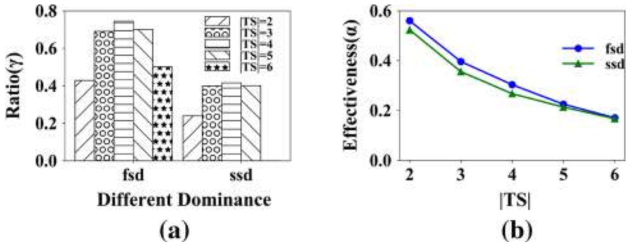

Fig. 18 Effectiveness, RU. **a** Comparison with expectations and **b** ratio of non-dominated variables

the grouping strategy is more stable and is insensitive to standard deviation variations. In all settings, *GRP* and *MSG* are faster than the other two methods that do not use grouping, and *MSG* is the fastest.

6.3 Effectiveness

6.3.1 Comparison with expectations

To gain insight into the functionality improvements of the proposed methods that take into account users' risk preferences, we compare with a method that does not take users' risk preferences into account. Specifically, we employ a method that uses expected values, i.e., expected travel time in our setting. This means that one random variable dominates another random variable if the former has a smaller expected value.

Given a UTS collection, this expectation-based method always chooses, for each interval, the random variable with the smallest expected travel time, meaning that it does not consider any risk preferences. In contrast, the temporal dominance queries choose, for each interval, the random variables that are not dominated by other random variables w.r.t. fsd (i.e., for risk-neutral users), ssd (i.e., for risk-loving users), or scsd (i.e., for risk-averse users).

We compute the ratio $\gamma = \frac{Diff}{Total}$, where $Diff$ is the number of queries whose results returned by the two methods are different and $Total$ is the number of all queries. Thus, the ratio expresses the difference between the results returned by the two methods. Figure 18 shows that the temporal dominance queries w.r.t. fsd and ssd return significantly different results when compared to the method using expected values. This offers evidence that the proposed temporal dominance queries are able to support different scenarios with different risk preferences.

6.3.2 Ratio of non-dominated variables

In this experiment, we evaluate the effectiveness of the proposed temporal dominance queries when a specific utility function is provided, i.e., the third scenario in Sect. 3.3. In this scenario, we consider two options. First, we compute the expectations for all random variables based on the given utility function. We denote the number of expectation computations by total. Second, we first categorize each utility function as convex, concave, or other; then we only need to compute expected utilities for the random variables in $O_{\text{fsd}}(\text{RVS})$, $O_{\text{ssd}}(\text{RVS})$, or $O_{\text{scsd}}(\text{RVS})$, not for other dominated RVs. We denote the number of expectation computation in the second option as $NonDominated$.

Next, we use the ratio $\frac{NonDominated}{Total}$ to measure the effectiveness of calculating expected utilities using the two options. The lower the ratio is, the better pruning effectiveness the temporal dominance queries provide. Fig. 18b shows that the ratios w.r.t fsd and ssd go down when the cardinalities of the UTS increase. On average, more than half of the random variables can be safely pruned when the cardinality exceeds 2. In addition, we also observe that the ratio for fsd is always larger than that for ssd. This is true because the non-dominated set regarding ssd is always a subset of that regarding fsd.

7 Related work

We review related work in relation to stochastic dominance, uncertain time series, and path selection.

7.1 Stochastic dominance

Stochastic dominance is fundamentally different from the dominance notion that is used widely in *skyline* queries [29], where dominance is defined in terms of pareto-optimality for multi-dimensional, deterministic data. In contrast, the stochastic dominance considered here is defined on one-dimensional, uncertain data.

Skyline queries can also be applied to uncertain data, yielding *probabilistic* [30,31] and *stochastic* [7,26,27,32] skyline queries. Probabilistic skylines adopt a possible worlds semantics [33]. A skyline is computed for each possible world, and an uncertain object has a probability of being in the skyline when considering all possible worlds. Stochastic dominance does not use possible world semantics, but instead considers cdfs and integrals of cdfs.

Stochastic skylines [26,27] are defined in terms of stochastic dominance. However, only first-order dominance is considered, and algorithms exist only for discrete distributions in the form of (*value*, *probability*) pairs. The efficiency of an existing method relies on the number of *distinct values* [27]. When the *value*s in the discrete distributions are sparse and not aligned, there exist only a limited number of distinct values, which offers good efficiency. In contrast, when the *value*s in the discrete distributions are not sparse or well aligned, e.g., discrete representations of continuous distributions, all the *value*s become distinct values, and the method deteriorates to the *NAI*+*IC* method covered in Sect. 6.2.1. Our proposal supports stochastic dominance with different orders of dominance and supports both discrete and continuous distributions. Further, the stochastic skyline operator does not take into account random variable sequences, while our proposal considers this setting and provides efficient group dominance checking for random variable sequences.

7.2 Uncertain time series

Most studies involving uncertain time series (UTS) concern similarity search in a top-k or threshold-based manner [23–25,34,35]. In contrast, our temporal dominance query on uncertain time series identifies the optimal random variables in each considered time interval, which is different from similarity search.

In particular, MUNICH [24] and PROUD [34] consider threshold-based similarity search where two thresholds are provided in advance—a distance threshold ϵ and a probability threshold τ. Given a query UTS T_Q, a threshold-based similarity search returns a set of UTSs. Each returned UTS, say T_i, has a sufficiently large (i.e., larger than τ) probability that the distance between T_i and the query UTS T_Q is smaller than the distance threshold ϵ. MUNICH assumes that the random variables in different intervals are independent and offers speedup techniques to answer threshold-based similar searches. PROUD assumes that statistical information, e.g., means and variances, is available for all random variables in different intervals and utilizes the central limit theorem to prune dissimilar UTSs.

DUST [35] also considers similarity search but relies on only a single distance threshold ϵ. In particular, given two UTSs, DUST is able to return a deterministic distance value that measures the similarity between the two UTSs without requiring a probability threshold τ. DUST performs better than MUNICH and PROUD when random variables follow different distributions during different intervals. A benchmark [23] compares the three methods, i.e., MUNICH, PROUD, and DUST, for UTS similarity search and makes recommendations on the scenarios in which each method performs the best.

Holistic-PkNN [25] investigates the problem of efficiently computing top-k nearest neighbors of UTSs. Given a query UTS T_Q, the probability of a UTS being the nearest neighbor to T_Q is defined, and an efficient query processing algorithm is proposed to find k UTSs having the top-k largest probabilities.

One study considers probabilistic skylines on uncertain time series [36]. It adopts possible worlds semantics to model a UTS as a collection of multi-dimensional points and then applies probabilistic skyline queries to them. Our study also differs from this study, as we do not use possible worlds semantics.

7.3 Path selection

A few studies consider path selection under different user preference [14,37–41]. They either do not consider uncertain travel times or consider only uncertain travel times for some specific departure time or interval. Some of these studies also consider specific preference functions or preference vectors, but they do not consider preference categories. Thus, the paper's proposal also differs from this line of research. A recent paper considers identifying non-dominated paths w.r.t. fsd [42] for a given interval, e.g., a peak or off-peak interval, while our paper is able to identify non-dominated paths w.r.t fsd, ssd, and scsd for different intervals.

8 Conclusions and outlook

We propose and study temporal dominance queries on uncertain time series. We provide a comprehensive analysis of stochastic dominance and user risk preferences, and utility functions. Efficient methods for checking stochastic dominance between two random variables and two random variable groups are proposed to efficiently compute temporal dominance queries on UTSs. Empirical studies with two real-world UTS collections suggest that the proposed methods are effective and efficient.

In future work, it is of interest to consider decision making with multivariate random variables, e.g., travel paths with both travel time and fuel consumption [43–45] distributions. It is also of interest to consider grouping strategies involving different UTSs, not only different intervals, to further improve temporal dominance query processing efficiency.

Acknowledgements This research was supported in part by a grant from the Obel Family Foundation and by the DiCyPS center, funded by Innovation Fund Denmark.

References

1. Guo, C., Jensen, C.S., Yang, B.: Towards total traffic awareness. SIGMOD Record **43**(3), 18–23 (2014)

2. Walpen, J., Mancinelli, E.M., Lotito, P.A.: A heuristic for the od matrix adjustment problem in a congested transport network. Eur. J. Oper. Res. **242**(3), 807–819 (2015)
3. Youngblom, E.: Travel time in macroscopic traffic models for origin-destination estimation. Master's Thesis, University of Wisconsin-Milwaukee (2013)
4. Multimodal accessibility measures in the Helsinki metropolitan region: metropaccess-travel time matrix. The Department of Geosciences and Geography, University of Helsinki. http://blogs.helsinki.fi/accessibility/data/ (2015)
5. Wang, F., Yanqing, X.: Estimating o-d travel time matrix by google maps api: implementation, advantages, and implications. Ann. GIS **17**(4), 199–209 (2011)
6. Yang, B., Kaul, M., Jensen, C.S.: Using incomplete information for complete weight annotation of road networks. IEEE Trans. Knowl. Data Eng. **26**(5), 1267–1279 (2014)
7. Yang, B., Guo, C., Jensen, C.S., Kaul, M., Shang, S.: Stochastic skyline route planning under time-varying uncertainty. In: ICDE, pp. 136–147 (2014)
8. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio-temporally correlated time series using markov models. PVLDB **6**(9), 769–780 (2013)
9. Erkut, E., Ingolfsson, A., Erdoğan, G.: Ambulance location for maximum survival. Naval Res. Log. (NRL) **55**(1), 42–58 (2008)
10. Woodard, D., Nogin, G., Koch, P., Racz, D., Goldszmidt, M., Horvitz, E.: Predicting travel time reliability using mobile phone GPS data. Transp. Res. Part C Emerg. Technol. **75**, 30–44 (2017)
11. Jenelius, E.: The value of travel time variability with trip chains, flexible scheduling and correlated travel times. Transp. Res. Part B Methodol. **46**(6), 762–780 (2012)
12. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: SIGSPATIAL, pp. 336–343 (2009)
13. Ding, Z., Yang, B., Güting, R.H., Li, Y.: Network-matched trajectory-based moving-object database: models and applications. IEEE Trans Intell Transp Syst **16**(4), 1918–1928 (2015)
14. Yang, B., Guo, C., Ma, Y., Jensen, C.S.: Toward personalized, context-aware routing. VLDB J. **24**(2), 297–318 (2015)
15. Ding, Z., Yang, B., Chi, Y., Guo, L.: Enabling smart transportation systems: A parallel spatio-temporal database approach. IEEE Trans Comput **65**(5), 1377–1391 (2016)
16. Dai, J., Yang, B., Guo, C., Jensen, C.S., Jilin, H.: Path cost distribution estimation using trajectory data. PVLDB **10**(3), 85–96 (2016)
17. Jilin, H., Yang, B., Jensen, C.S., Ma, Y.: Enabling time-dependent uncertain eco-weights for road networks. GeoInformatica **21**(1), 57–88 (2017)
18. Yang, B., Dai, J., Guo, C., Jensen, CS.: PACE: A PAth-CEntric paradigm for stochastic path finding. VLDB J (2017)
19. Kijima, M., Ohnishi, M.: Stochastic orders and their applications in financial optimization. Math. Methods Oper. Res. **50**(2), 351–372 (1999)
20. Levy, H.: Stochastic dominance and expected utility: survey and analysis. Manag. Sci. **38**(4), 555–593 (1992)
21. Weinstein, A., Marsden, J.: Calculus II, 2nd edn. Springer, New York (1985)
22. Yang, B., Ma, Q., Qian, W., Zhou, A.: TRUSTER: TRajectory data processing on ClUSTERs. DASFAA 768–771 (2009)
23. Dallachiesa, M., Nushi, B., Mirylenka, K., Palpanas, T.: Uncertain time-series similarity: return to the basics. PVLDB **5**(11), 1662–1673 (2012)
24. Aßfalg, J., Kriegel, H.P., Kröger, P., Renz, M.: Probabilistic similarity search for uncertain time series. In: SSDBM, pp. 435–443 (2009)
25. Dallachiesa, M., Palpanas, T., Ilyas, I.F.: Top-k nearest neighbor search in uncertain data series. PVLDB **8**(1), 13–24 (2014)
26. Lin, X., Zhang, Y., Zhang, W., Cheema, M.A.: Stochastic skyline operator. In: ICDE, pp. 721–732 (2011)
27. Zhang, W., Lin, X., Zhang, Y., Cheema, M.A., Zhang, Q.: Stochastic skylines. TODS **37**(2), 14 (2012)
28. Li, Q., Nie, Y.M., Vallamsundar, S., Lin, J., Homem-de Mello, T.: Finding efficient and environmentally friendly paths for risk-averse freight carriers. Netw. Spat. Econ. **16**(1), 255–275 (2016)
29. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
30. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: PVLDB, pp. 15–26 (2007)
31. Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.X.: Probabilistic skyline operator over sliding windows. In: ICDE, pp. 1060–1071 (2009)
32. Guo, C., Yang, B., Andersen, O., Jensen, C.S., Torp, K.: Ecosky: Reducing vehicular environmental impact through eco-routing. In: ICDE, pp. 1412–1415 (2015)
33. Sarma, A.D., Benjelloun, O., Halevy, A., Widom, J.: Working models for uncertain data. In: ICDE, p. 7 (2006)
34. Yeh, M.-Y., Wu, K.-L., Yu, P.S., Chen, M.-S.: PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In: EDBT, pp. 684–695 (2009)
35. Sarangi, S. R., Murthy, K.: DUST: a generalized notion of similarity between uncertain time series. In: SIGKDD, pp. 383–392 (2010)
36. Guoliang He, L., Chen, C.Z., Zheng, Q., Zhou, G.: Probabilistic skyline queries on uncertain time series. Neurocomputing **191**, 224–237 (2016)
37. Dai, J., Yang, B., Guo, C., Ding, Z.: Personalized route recommendation using big trajectory data. In: ICDE, pp. 543–554 (2015)
38. Delling, D., Goldberg, A. V., Goldszmidt, M., Krumm, J., Talwar, K., Werneck, R.F.: Navigation made personal: inferring driving preferences from GPS traces. In: SIGSPATIAL, pp. 31:1–31:9 (2015)
39. Balteanu, A., Jossé, G., Schubert, M.: Mining driving preferences in multi-cost networks. In: SSTD, pp. 74–91 (2013)
40. Liu, H., Jin, C., Yang, B., Zhou, A.: Finding top-k shortest paths with diversity. IEEE Trans Knowl Data Eng (2017)
41. Shang, S., Zheng, K., Jensen, C.S., Yang, B., Kalnis, P., Li, G., Wen, J.-R.: Discovery of path nearby clusters in spatial networks. IEEE Trans Knowl Data Eng **27**(6), 1505–1518 (2015)
42. Aljubayrin, S., Yang, B., Jensen, C.S., Zhang, R: Finding non-dominated paths in uncertain road networks. In: SIGSPATIAL, pp. 15:1–15:10 (2016)
43. Guo, C., Yang, B., Andersen, O., Jensen, C.S., Torp, K.: Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data. GeoInformatica **19**(3), 567–599 (2015)
44. Guo, C., Ma, Y., Yang, B., Jensen, C.S., Kaul, M.: EcoMark: evaluating models of vehicular environmental impact. In: SIGSPATIAL, pp. 269–278 (2012)
45. Andersen, O., Jensen, C.S., Torp, K., Yang, B.: Ecotour: Reducing the environmental footprint of vehicles using eco-routes. In: MDM, pp. 338–340 (2013)