Approximating Multivariate Markov Chains for Bootstrapping through Contiguous Partitions[1]

Roy Cerqueti

Università degli Studi di Macerata - Dipartimento di Economia e Diritto

Via Crescimbeni, 20 - 62100 Macerata MC - Italy

E-mail: roy.cerqueti@unimc.it

Paolo Falbo*, Gianfranco Guastaroba, Cristian Pelizzari

Università degli Studi di Brescia - Dipartimento di Economia e Management

Contrada Santa Chiara, 50 - 25122 Brescia BS - Italy

E-mail: {paolo.falbo, gianfranco.guastaroba, cristian.pelizzari}@unibs.it

*Corresponding author. Tel.: +39 030 2988531. Fax: +39 030 2400925. E-mail: paolo.falbo@unibs.it

Abstract This paper extends Markov chain bootstrapping to the case of multivariate continuous-valued stochastic processes. To this purpose we follow the approach of searching an optimal partition of the state space of an observed (multivariate) time series. The optimization problem is based on a distance indicator calculated on the transition probabilities of the Markov chain. Such criterion searches to group those states showing similar transition probabilities. A second methodological contribution is represented by the addition of a contiguity constraint, which is introduced to force the states to group only if they have "near" values (in the state space). This requirement meets two important aspects: firstly, it allows a more intuitive interpretation of the results; secondly, it contributes to control the complexity of the problem, which explodes with the cardinality of the states. The computational complexity of the optimization problem is also addressed through the introduction of a novel Tabu Search algorithm, which improves both the quality of the solution found and the computational times, with respect to a similar heuristic previously advanced in the literature. The bootstrap method is applied to two empirical cases: the bivariate process of prices and volumes of electricity in the Spanish market; the trivariate process composed by prices and volumes of a US company stock (McDonald's) and prices of the Dow Jones Industrial Average index. Additionally, the method is compared with two other well established bootstrap methods. The results show the good distributional properties of the present proposal, as well as a clear superiority in reproducing the dependence among the data.

Keywords: Multivariate Markov chains; Bootstrapping; Heuristic algorithms; Tabu Search; Contiguous states; Stock and electricity prices and volumes

[1] The work has been supported under grant by "Regione Lombardia: Metodi di integrazione delle fonti energetiche rinnovabili e monitoraggio satellitare dell'impatto ambientale", EN-17, ID 17369.10.

1 Introduction and bibliography review

The correct understanding of economic and financial phenomena is crucial to several purposes, from model calibration to forecasting. Besides theoretical analysis, many statistical approaches have been developed. Among these a powerful method is the bootstrap, originally introduced by Efron (1979). To apply the original method to an observed time series, it is preliminarily required that the series is fitted with an accurate model, so that the residuals are strictly stationary and independent. If these conditions are met, the bootstrap can be successfully applied resampling the residuals (with repetition) to reproduce a large number of bootstrapped series. Needless to say, the more appropriate is the choice for the model of the observed time series, the more accurate and informative the bootstrap. Financial applications of this methodology are the focus of a wide strand of literature, here we mention only the prominent studies on financial markets due to Brock et al. (1992) and Sullivan et al. (1999).
However, the strict stationarity and independence conditions imposed in the original method by Efron happen to be too restrictive in many practical applications, especially where data tend to show nonlinear dependence. To overcome such drawback, bootstrap theory has addressed to more data driven approaches. Block, sieve, and local bootstraps are examples of these approaches (see Bühlmann, 2002, for a comparison), which require weaker stationarity conditions and, even more important, are less demanding with respect to the independence of data. Indeed, these methods aim at capturing such dependence directly from the data, therefore they reduce the risk of misspecifying the model applied to prepare the data. In the case of Markov chain bootstrapping a simple removal from the observed time series of the deterministic components (such as trend and seasonality) suffices to proceed safely with the bootstrap. This class of bootstrap methods do not require specific assumptions on the probability distributions of the random variables, that is why they are also referred to as nonparametric and in a certain sense model-free. Therefore they can be fruitfully applied in situations where analytical (reduced form) models and econometric models show serious drawbacks. In many economic areas where time series analysis has flourished, such as in finance, there are many examples (e.g., stock returns, exchange rates, interest rates) where analytic (reduced form) models successfully fit observed time series. A discussion on the parametric case with financial applications has been carried out in Guastaroba et al. (2009). However, turning to less liquid markets, such as commodities or energy, the shocks on prices arise much more irregularly, therefore several analytic models show serious difficulties. Econometric models are in general much more flexible than analytic ones. They include a large variety of ways to link a dependent variable to its past values, its past residuals, as well as other variables and their residuals. A natural limit of these models is of course represented by the calibration of their parameters, which often becomes unmanageable in the presence of nonlinear dependence among the data. In general, this is the case where data driven approaches, such as the Markov chain bootstrapping proposed here, are successfully adopted.
In this paper we focus on the family of bootstrap methods based on Markov chain theory. More specifically, we apply Markov chain theory to bootstrap observed multivariate time series taking continuous values.
Markov chain bootstrapping is based on the assumption that an observed time series may be viewed as a realization of a Markov chain.
The case of univariate Markov chains with finite states is undoubtedly one of the most explored. Some classical references are Kulperger and Prakasa Rao (1989), Athreya and Fuh (1992), and Datta and McCormick (1992). The quoted papers face explicitly the problem of estimating the transition probability matrix of a stationary Markov chain with the objective of saving most information on data dependence.
Some papers refer to Markovian processes, and provide an estimation of their transition density function by adopting a kernel-based procedure (see, e.g., Rajarshi, 1990; Horowitz, 2003; Paparoditis and Politis, 2001b). These works assume that similar trajectories (with respect to a distance measured on the state space) will tend to show similar transition probabilities. Moreover, no particular attention

is paid to the order of the chain (i.e. the memory of the Markov chain).
In general, modeling an observed time series as a discrete Markov chain moves from the assumption that the observed values are its states. Of course, this assumption does not hold when dealing with continuous-valued time series. To apply Markov chain theory to the continuous case, a discretization (i.e. a partition) of the range is required, where each interval is assumed as a state. An early analysis of the advantages of adopting Markov chain bootstrapping to continuous-valued stochastic processes has been faced by Anatolyev and Vasnev (2002). They construct a bootstrap method on the basis of a partition of the state space, such that each state comprises approximately a constant number of observations, and conclude that such bootstrap method shows interesting estimation properties. Of course, such simple approach does not really tell us much about the data generating process. In particular, the number of states used to partition the range can be either larger or smaller than it is required for description purposes, and the cutting points do not necessarily reflect any effective change of regime.

Indeed, optimally partitioning the range of a continuous-valued time series, for the purpose of applying Markov chain bootstrapping, is a major problem for different reasons.
The natural objective of describing satisfactorily the dependence structure of the observed time series would push in the direction of searching for partitions rich of states, implying the estimation of larger and larger transition probability matrices. Such tendency contrasts however with the number of available observations (i.e. the length of the original sample), which become rapidly "short" as soon as the dimension of the transition probability matrix increases. Moreover, the mathematical tractability of a Markov chain also reduces as the dimension of its transition probability matrix increases.
There is another subtle difficulty linked to increasing the number of states. With continuous ranges, it is often possible to identify a partition such that every state occurred only once in the observed time series. In such a case, the transition probability matrix would degenerate to a "0-1" matrix[2], and the series which would result by bootstrapping it would just be exact replications of the observed time series. This is of course an unwanted result, showing that the number of states required to bootstrap a continuous-valued stochastic process through Markov chain must be balanced between a search for description quality and a need of diversification.
Finally, a relevant difficulty in determining the states of a Markov chain for a continuous-valued stochastic process is that of identifying the correct "cutting values" of its range, which is at least as important as deciding on the optimal number of states.

A general approach introduced to solve this class of problems has been advanced in Cerqueti et al. (2010). They start from a given partition of the range, which includes a number of states larger than those needed for a Markov chain to describe successfully the data process. They then look for an information efficient assessment of the transition probability matrix by implementing a suitable clustering of the states. In this way they reduce their number, approximate the discriminant cutting points of the range and assess the relevance of the time lags within the order of the chain.
Some contributions are relevant, in our opinion, to analyze the order of a Markov chain. Bühlmann and Wyner (1999) propose a sophisticated procedure -the so-called *variable length Markov chain*-. Basically, starting from the initial data of a path, a recursive routine is established to differentiate states only when they contribute to differentiate future evolution. At the last step of the recursion, the identification of a Markov process whose memory depends on the realized paths is obtained. The authors achieve a satisfactory computational efficiency. Some other contributions are worth mentioning. Merhav et al. (1989), Kieffer (1993), Csiszár and Shields (2000), and Morvai and Weiss (2005) deal with the estimation of the order of a Markov chain in a framework of equal relevance of the states at each time lag.

Following Cerqueti et al. (2010), we move from a discretized version of a continuous-valued stochastic process, and so our work can be inserted among the research area of the reduction of

[2] A "0-1" matrix is a stochastic matrix whose rows contain zeros in all positions but one, where there is a 1.

a discrete Markov chain by clustering its state space. In general, the reduction of the state space of a Markov chain by means of some clustering procedures returns more tractable processes, but at the cost of some information loss, as it is well known in clustering (or lumpability) theory. Among the drawbacks, there is the risk that part of the mathematical properties of the original Markov chain get lost.
The reference literature in this area is in general concerned with the problem of information loss and lumpability theory. Some classical references on lumpability are Burke and Rosenblatt (1958) and Kemeny and Snell (1976). An encyclopedic survey is provided by Thomas (2010). Relevant applications of this theory include the compression of data represented through a finite alphabet. Some important examples of data compression criteria are the AIC (Akaike Information Criterion, Akaike, 1970), the BIC (Bayesian Information Criterion, Schwarz, 1978) and the MDL principle (Minimum Description Length principle, Rissanen, 1978). In each criterion, the compression is implemented to obtain the maximum level of simplification (i.e. minimization of an entropy-based functional) also by pursuing the scope of minimizing the information loss (i.e. minimization of a penalty term).
In the present approach we extend the theory of lumpability and model jointly the clustering of the state space and the assessment of the order of a Markov chain. At this aim, we construct a unique optimization problem to identify both relevant states and time lags. Such optimization problem is linked to information theory: specifically, two definitions of distance measure are introduced, in order to pursue simultaneously the maximum level of simplification and the minimum level of information loss.

Another relevant comparison is that between our approach and Markov switching models (on these models see, for example, the recent review in Franke, 2012). Despite their name, these models are analytic: a mathematical expression defines the dynamics of the process. The difference in these models is represented by the particular link that one or more parameters establish with the current value of the process, or with one of its attributes (e.g., its local volatility). In particular, this link is modeled classifying the value of the process (or of its attribute) according to few "regimes". Its parameters take different values conditioning on the current state of the process and on a transition probability matrix. In this way the process changes "regimes" through time, respecting Markov chain probabilities. The difference with our Markov chain bootstrapping is deep. The lack of any analytic expression to define the dynamics of the process and the hidden number of states (as well as of the cutting values on the range used to define them) are key distinguishing features.

Our extension to the multivariate case increases the computational complexity of the problem. To deal with multivariate Markov chains of order $k \geq 1$, it is required to pass from defining a state as the combination of values observed for a single variable over an interval of k times, to the combination of values of a vector (over the same period). Some attempts have been proposed in the literature to manage the complexity of the clustering problem. In Cerqueti et al. (2012), the optimization problem has been conveniently rewritten as a mixed integer linear programming problem and its exact solution has been found. The authors obtain satisfactory results in terms of computing times to find the optimal solution, even though their approach has to be adopted to solve problems of small to medium size (i.e. 12-15 states observed over 2-3 time lags). Cerqueti et al. (2013) overcome the complexity problem and solve real life instances of several observed time series and a large number of relevant time lags by implementing a heuristic procedure. Specifically, the authors adopt a Tabu Search algorithm and find heuristic solutions of the optimization problem in a reasonable computational time. Moreover, extensive numerical experiments are provided to show that the resulting bootstrapped series respect in full the statistical properties of the observed time series.
Given the relevancy of the computational complexity, in this paper we also rely on a Tabu Search approach. Besides, we introduce a contiguity constraint for the clusters of the partitions, with the purposes of reducing the complexity of the problem and of improving the interpretation of the relevant states. It is worth observing that, in our approach, the distance between any two states is calculated considering only the values of their transition probabilities. Such approach is therefore free to group states which can be arbitrarily distant on the range of the process. The contiguity con-

straint introduced here mitigates that extreme freedom. It also meets the evidence that, in several cases, the rationale of a group of states (such as a regime) can be better described combining the closeness of the transition probabilities with the closeness of their values. In this respect, the specific context of the regimes of the electricity prices is paradigmatic. Indeed, electricity prices exhibit a rather stationary behavior with some sudden peaks, and the identification of the regimes is based on grouping together states belonging to the same space-intervals (see, Huisman and Mahieu, 2003; Bunn, 2004; Weron et al., 2004; Weron, 2006).
To the best of our knowledge, this paper is the first to explore multivariate Markov chain bootstrapping. The theoretical advantages of univariate Markov chain bootstrapping are even stronger under the multivariate setting. Transition probability matrices are suitable to model arbitrary types of dependence among the components of a multivariate stochastic process as well as those among their values at different time lags. This aspect is quite relevant in real life applications, mainly for what concerns the analysis of contagion effects and causality relations. Our paper also contributes to the operational research literature. We design a novel Tabu Search heuristic to solve the optimization problem. The flexibility of the heuristic enables to solve problems under univariate or multivariate settings, as well as with or without a contiguity constraint. The only solution strategy available in the literature is the heuristic algorithm introduced in the aforementioned work of Cerqueti et al. (2013), and developed to solve the optimization problem under a univariate setting and without a contiguity constraint. We show that the new Tabu Search algorithm outperforms the latter heuristic, in terms of both solution quality and computing times.

The paper is organized as follows. Section 2 introduces the model of the optimal clustering of multivariate states of a Markov chain of order $k \geq 1$. Section 3 describes the Tabu Search algorithm advanced to solve the computational complexity of the solution search. Section 4 focuses on the application of the bootstrap method to two empirical cases. The first is concerned with prices and volumes of electricity in the Spanish market. The second considers a trivariate stochastic process: prices and volumes of a US company stock (McDonald's) and prices of the Dow Jones Industrial Average (DJIA) index. Section 5 comments on the results and Section 6 concludes. Finally, Online Resource 1 details Tabu Search pseudo-codes, removal of trend and seasonality, initial states, Tabu Search settings and computational results, and information loss and distance measures.

2 Problem Setting

2.1 Preliminaries and notation

Let us consider an N-dimensional discrete time continuous-valued stochastic process $\mathbf{Y} = \{\mathbf{Y}_m\}_{m\in\mathbb{N}^+} \subseteq \mathbb{R}^N$. For each $n = 1, ..., N$, we will refer to the n-th component of $\mathbf{Y}$ as $\{Y_m^{(n)}\}_{m\in\mathbb{N}^+}$. The realization of $\mathbf{Y}_m$ (i.e. the value assumed by $\mathbf{Y}$ at time m and under the occurrence of a given state of the world) will be named as $\mathbf{y}_m$ and that of its n-th component as $y_m^{(n)}$.
Consider the vector $\mathcal{O}$ collecting the values of a trajectory of length M of our continuous-valued stochastic process $\mathbf{Y}$:

$$\mathcal{O} = (\mathbf{y}_1, ..., \mathbf{y}_m, ..., \mathbf{y}_M). \tag{1}$$

We will call a vector of k consecutive realizations of the process as a k-trajectory. The k-trajectory starting at time s is the vector $\mathbf{y}_{s,k} = (\mathbf{y}_s, ..., \mathbf{y}_{s+k-1})$, with $M \geq 2$, $k = 1, ..., M-1$ and $s \in \{1, ..., M-k+1\}$.
Define the set $\mathcal{O}_k$ collecting all the k-trajectories that can be extracted from $\mathcal{O}$ as:

$$\mathcal{O}_k = \{\mathbf{y}_{1,k}, ..., \mathbf{y}_{M-k+1,k}\}. \tag{2}$$

Let's now define the discretized version of $\mathbf{Y}$. Fix $n = 1, ..., N$ and assume that the support of the n-th component of $\mathbf{Y}$ is given by an interval $\mathbb{I}_n \subseteq \mathbb{R}$. Consider a partition of $\mathbb{I}_n$ into L_n

(non-overlapping) intervals, or *states*, $a_1^{(n)}, a_2^{(n)}, ..., a_l^{(n)}, ..., a_{L_n}^{(n)} \subseteq \mathbb{I}_n$, such that:

$$\begin{cases} a_{l'}^{(n)} \cap a_{l''}^{(n)} = \varnothing, & \text{for } l' \neq l'',\ l', l'' = 1, ..., L_n, \\ \bigcup_{l=1}^{L_n} a_l^{(n)} = \mathbb{I}_n, \end{cases} \tag{3}$$

and let $\mathbb{A}_n = \left\{a_1^{(n)}, a_2^{(n)}, ..., a_l^{(n)}, ..., a_{L_n}^{(n)}\right\}$. We will refer to $\mathbb{A}_n$ as to the *initial* partition of $\mathbb{I}_n$.

Remark 1 Without loss of generality, we assume that the intervals of $\mathbb{A}_n$ are indexed in an increasing order, that is:

$$\sup(a_l^{(n)}) \leq \inf(a_{l+1}^{(n)}), \qquad \text{for each } l = 1, ..., L_n. \tag{4}$$

The Cartesian product of the $\mathbb{A}_n$ is

$$\mathbb{A} = \mathbb{A}_1 \times ... \times \mathbb{A}_n \times ... \times \mathbb{A}_N. \tag{5}$$

The z-th element of $\mathbb{A}$ is the vector of N components $\mathbf{a}_z = \left(a^{(1)}, ..., a^{(n)}, ..., a^{(N)}\right)$, where $a^{(n)}$ is a generic interval, or state, of $\mathbb{A}_n$, and $z = 1, ..., \sharp(\mathbb{A})$. The number $\sharp(\mathbb{A}) = \prod_{n=1}^{N} L_n$ is the cardinality of $\mathbb{A}$. We can also define $\mathbb{A}$ by listing its elements:

$$\mathbb{A} = \{\mathbf{a}_1, ..., \mathbf{a}_z, ..., \mathbf{a}_{\sharp(\mathbb{A})}\}. \tag{6}$$

The elements of $\mathbb{A}$ are called *N-intervals*, or *N-states*. The set $\mathbb{A}$ can be viewed as the support of a *discretized version* of $\mathbf{Y}$.
Since the elements of $\mathbb{A}_n$ are non-overlapping intervals for each $n = 1, ..., N$, there exists a unique $\mathbf{a}_z$, with $z = 1, ..., \sharp(\mathbb{A})$, such that $\mathbf{y}_m \in \mathbf{a}_z$, for all $m \in \mathbb{N}^+$. Therefore, letting:

$$\mathbf{x}_m = \mathbf{a}_z \text{ if } \mathbf{y}_m \in \mathbf{a}_z,$$

vector $\mathcal{O}$ in Eq. (1) can be rewritten as a vector of the same length, whose elements are N-states extracted from $\mathbb{A}$ as:

$$\mathcal{A} = (\mathbf{x}_1, ..., \mathbf{x}_m, ..., \mathbf{x}_M).$$

In a similar way, let us consider the vector $\mathbf{a}_{h,k} = (\mathbf{a}_{h_k}, ..., \mathbf{a}_{h_1}) \in \mathbb{A}^k$, where $h_w = 1, ..., \sharp(\mathbb{A})$, $w = 1, ..., k$. The row vector $\mathbf{a}_{h,k}$ is the ordered set of k N-states $\mathbf{a}_{h_k} \in \mathbb{A}$, listed, in a natural way, from the furthest to the closest realization of the chain. The row vector $\mathbf{a}_{h,k}$ will be called *kN-interval*, or *kN-state*. Now, the k-trajectories collected in set $\mathcal{O}_k$ of Eq. (2) can be rewritten by means of the corresponding kN-states of $\mathbb{A}^k$ giving:

$$\mathcal{A}_k = \{\mathbf{x}_{1,k}, ..., \mathbf{x}_{M-k+1,k}\},$$

where

$$\mathbf{x}_{m,k} = \mathbf{a}_{h,k} \text{ if } \mathbf{y}_{m,k} \in \mathbf{a}_{h,k}.$$

2.2 Problem overview

It is now worth recalling the basic idea of our problem. Given a realization of a multivariate discrete time continuous-valued stochastic process, i.e. an observed multivariate time series, we aim at finding a suitable approximation of the process by means of a Markov chain of order $k \geq 1$ with a finite number of multivariate states. We start with $\mathcal{O}$, i.e. a realization of the process. $\mathcal{O}$ consists of N observed time series of length M, linked through an unknown dependence structure. Given $\mathcal{O}$, we want to bootstrap the process which generated it. The approach that we adopt in this work is to resort to the theory of Markov chains of order $k \geq 1$. To this purpose, instead of focusing on the observed values collected in $\mathcal{O}$, we will focus on their *discretized* representation collected in $\mathcal{A}$. Specifically, $\mathcal{A}$ may be viewed as a vector collecting M realizations of a Markov chain $\mathbf{X} = \{\mathbf{X}_m\}_{m \in \mathbb{N}^+}$ of order $k \geq 1$ and with discrete support $\mathbb{A}$. Hence, we will refer to $\mathcal{A}$ as a discretized version of $\mathcal{O}$, and $\mathbf{X}$ may be intuitively thought as a discretized version of $\mathbf{Y}$.
The critical point for the Markov chain $\mathbf{X}$ to represent the *best* approximation of the stochastic process $\mathbf{Y}$ is to obtain the *best* estimation of its transition probability matrix. The probabilities of this matrix describe the dynamics of $\mathbf{X}$. However, as pointed out in Cerqueti et al. (2010), two contrasting objectives emerge here. On the one hand, we would like to estimate the matrix keeping all the N-states of $\mathbb{A}$ distinguished, so to keep all the available information contained in the original sample $\mathcal{A}$. On the other hand, aggregating some N-states can be a desirable result, since merging them could result in several benefits, such as a simpler representation of $\mathbf{X}$, a better understanding of the relevant regimes of the unknown stochastic process $\mathbf{Y}$, a clearer view of its significant thresholds, and a better diversification of the bootstrapped series.
At the same time, for cardinalities of $\mathbb{A}$ larger than some hundreds, finding the optimal aggregation of the N-states of $\mathbb{A}$ to balance these two competing requirements is a computationally complex problem. To handle such complexity, the approach we take is to combine two devices: a Tabu Search heuristic to scan efficiently the space of admissible aggregations of N-states and a contiguity constraint to remove from the analysis aggregations of N-states where the N-states grouped in the same class are *distant* (in a sense which will be made clear in the following).

Subsection 2.6 provides an outline of the method we propose to tackle the problem.

2.3 Estimation of the transition probability matrices

Fix $M \geq 2$, $k = 1, ..., M-1$, $z = 1, ..., \sharp(\mathbb{A})$, and $h = 1, ..., \sharp(\mathbb{A}^k)$. Consider $\mathbf{a}_z \in \mathbb{A}$ and $\mathbf{a}_{h,k} \in \mathbb{A}^k$. The transition probability of the Markov chain $\mathbf{X}$ from the kN-state $\mathbf{a}_{h,k}$ to the N-state $\mathbf{a}_z$ is defined as:

$$P(\mathbf{a}_z|\mathbf{a}_{h,k}) = P(\mathbf{X}_m = \mathbf{a}_z|\mathbf{X}_{m-1} = \mathbf{a}_{h_1}, ..., \mathbf{X}_{m-k} = \mathbf{a}_{h_k}). \tag{7}$$

These probabilities are estimated on the basis of the original sample $\mathcal{O}$. More precisely, the transition probability $P(\mathbf{a}_z|\mathbf{a}_{h,k})$ in Eq. (7) is estimated through the empirical frequency $f(\mathbf{a}_z|\mathbf{a}_{h,k})$ of kN-state $\mathbf{a}_{h,k}$ evolving to N-state $\mathbf{a}_z$. This frequency can be formalized as follows:

$$f(\mathbf{a}_z|\mathbf{a}_{h,k}) = \sharp\left(\left\{\mathbf{x}_{s,k+1} \in \mathcal{A}_{k+1} : \mathbf{x}_{s,k+1} = (\mathbf{a}_{h_k}, \mathbf{a}_{h_{k-1}}, ..., \mathbf{a}_{h_1}, \mathbf{a}_z)\right\}\right). \tag{8}$$

Based on Ching et al. (2008) and on Eq. (8), we can provide an estimate of $P(\mathbf{a}_z|\mathbf{a}_{h,k})$ as:

$$\hat{P}(\mathbf{a}_z|\mathbf{a}_{h,k}) = \begin{cases} \frac{f(\mathbf{a}_z|\mathbf{a}_{h,k})}{\sum_{j=1}^{\sharp(\mathbb{A})} f(\mathbf{a}_j|\mathbf{a}_{h,k})}, \text{ if } \sum_{j=1}^{\sharp(\mathbb{A})} f(\mathbf{a}_j|\mathbf{a}_{h,k}) > 0, \\ \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \tag{9}$$

2.4 Contiguous partitions

We focus our attention only on contiguous partitions of $\mathbb{A}^k$ (see the application of our model in Section 4). Let us define them.

Definition 1 Fix $n = 1, ..., N$ and consider the following partition of $\mathbb{A}_n$:

$$\lambda_v^{(n)} = \{\theta_{v,1}^{(n)}, \theta_{v,2}^{(n)}, ..., \theta_{v,u}^{(n)}, ..., \theta_{v,U_n}^{(n)}\},$$

where $U_n = 1, ..., L_n$ and the subscript v is placed for notation convenience, as we will see below in Eqs. (10) and (11). $\lambda_v^{(n)}$ is said to be a *contiguous partition* of $\mathbb{A}_n$ when:

$$a_{l'}^{(n)} \subseteq \theta_{v,u'}^{(n)}, \;\; a_{l''}^{(n)} \subseteq \theta_{v,u''}^{(n)}, \qquad \text{with } l' < l'',$$

implies that $u' \leq u''$.
For a contiguous partition, $\theta_{v,u-1}^{(n)}$ and $\theta_{v,u+1}^{(n)}$ are said to be the *adjacent classes* of $\theta_{v,u}^{(n)}$, for each $u = 2, ..., U_n - 1$, while $\theta_{v,2}^{(n)}$ ($\theta_{v,U_n-1}^{(n)}$) is the only *adjacent class* of $\theta_{v,1}^{(n)}$ ($\theta_{v,U_n}^{(n)}$).
The contiguous partitions of $\mathbb{A}_n$ are collected in set Λ_n.

In loose words, a partition of $\mathbb{A}_n$ is contiguous if, given $l' < l'' < l'''$, it never puts states $a_{l'}^{(n)}$ and $a_{l'''}^{(n)}$ in the same class without also grouping state $a_{l''}^{(n)}$ in that class. We will refer to the characterization of contiguous partitions in Definition 1 as *contiguity constraint*.

Remark 2 We assume, without loss of generality, that the intervals $a_l^{(n)}$ of $\mathbb{A}_n$ belonging to a class $\theta_{v,u}^{(n)}$ of contiguous partition $\lambda_v^{(n)}$ are indexed in an increasing order, that is they satisfy Eq. (4).

Definition 1 can be easily extended to the multivariate case by introducing the set Λ of contiguous multivariate partitions of $\mathbb{A}$, which we will call N-partitions.

Definition 2 A (contiguous multivariate) N-partition of $\mathbb{A}$ is denoted as λ_v and is defined as:

$$\lambda_v = \{\theta_{v,u_1}^{(1)} \times ... \times \theta_{v,u_n}^{(n)} \times ... \times \theta_{v,u_N}^{(N)} | u_n = 1, ..., U_n, n = 1, ..., N\}, \tag{10}$$

where $\theta_{v,u_n}^{(n)}$ is a generic class of contiguous partition $\lambda_v^{(n)}$ of $\mathbb{A}_n$. To simplify notation, we put:

$$\theta_{v,u} = \theta_{v,u_1}^{(1)} \times ... \times \theta_{v,u_n}^{(n)} \times ... \times \theta_{v,u_N}^{(N)}.$$

Remark 3 An N-partition of $\mathbb{A}$ can also be (more easily) represented by the n-tuple of contiguous partitions $\lambda_v^{(n)}$, $n = 1, ..., N$, which the classes $\theta_{v,u_n}^{(n)}$ belong to. So N-partition λ_v can also be written as:

$$\lambda_v = (\lambda_v^{(1)}, \lambda_v^{(2)}, ..., \lambda_v^{(n)}, ..., \lambda_v^{(N)}).$$

Analogously to what we did for $\mathbb{A}$ in Eq. (6) and to simplify the notation, it is convenient to list the elements of Λ and write:

$$\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_v, ..., \lambda_{\sharp(\Lambda)}\}. \tag{11}$$

Definition 2 can be easily extended to the multidimensional case by introducing the set Λ^k of contiguous multivariate and multidimensional partitions of $\mathbb{A}^k$, which we will call kN-partitions.

Definition 3 A (contiguous multivariate and multidimensional) kN-partition of $\mathbb{A}^k$ is denoted as $\lambda_{\mathbf{v}}$ and is defined as:

$$\lambda_{\mathbf{v}} = \{\theta_{v_k,u_k} \times \theta_{v_{k-1},u_{k-1}} \times ... \times \theta_{v_w,u_w} \times ... \times \theta_{v_1,u_1} | w \in \{1, ..., k\}\},$$

where the double subscript v_w, u_w identifies the generic class θ_{v_w,u_w} of N-partition λ_{v_w} for time lag w. To simplify notation, we put:

$$\Theta_{\mathbf{v},\mathbf{u}} = \theta_{v_k,u_k} \times \theta_{v_{k-1},u_{k-1}} \times ... \times \theta_{v_w,u_w} \times ... \times \theta_{v_1,u_1}.$$

Remark 4 A kN-partition of $\mathbb{A}^k$ can also be (more easily) represented by the k-tuple of N-partitions λ_{v_w}, $w = 1, ..., k$, which the classes θ_{v_w,u_w} belong to. So kN-partition $\lambda_{\mathbf{v}}$ can also be written as:

$$\lambda_{\mathbf{v}} = (\lambda_{v_k}, \lambda_{v_{k-1}}, ..., \lambda_{v_w}, ..., \lambda_{v_1}).$$

The set Λ^k of kN-partitions of $\mathbb{A}^k$ can be represented by listing its elements as follows:

$$\Lambda^k = \{\lambda_{\mathbf{1}}, \lambda_{\mathbf{2}}, ..., \lambda_{\mathbf{v}}, ..., \lambda_{\sharp(\Lambda^k)}\}.$$

Remark 5 Consider a kN-partition of $\mathbb{A}^k$. If we choose $n \in \{1, ..., N\}$ and select $w \in \{1, ..., k\}$, the kN-partition of $\mathbb{A}^k$ reduces to a contiguous partition $\lambda_{v_w}^{(n)}$ of $\mathbb{A}_n$, which will also be referred to as *Univariate and Unidimensional Partition (UUP)*.

It is now convenient to define the transition probability between class $\Theta_{\mathbf{v},\mathbf{u}}$ of kN-partition $\lambda_{\mathbf{v}}$ and N-state $\mathbf{a}_z$ of $\mathbb{A}$. By referring to the probability law P introduced in Eq. (7), we have:

$$P(\mathbf{a}_z|\Theta_{\mathbf{v},\mathbf{u}}) = P(\mathbf{X}_m = \mathbf{a}_z|\mathbf{X}_{m-1} \in \theta_{v_1,u_1}, ..., \mathbf{X}_{m-k} \in \theta_{v_k,u_k}), \tag{12}$$

where the inclusions in the right-hand side have to be understood componentwise. Analogously to Eq. (9), we can estimate probability $P(\mathbf{a}_z|\Theta_{\mathbf{v},\mathbf{u}})$ in Eq. (12) through the empirical frequencies:

$$\hat{P}(\mathbf{a}_z|\Theta_{\mathbf{v},\mathbf{u}}) = \begin{cases} \frac{\sum_{h\,:\,\mathbf{a}_{h,k} \in \Theta_{\mathbf{v},\mathbf{u}}} f(\mathbf{a}_z|\mathbf{a}_{h,k})}{\sum_{h\,:\,\mathbf{a}_{h,k} \in \Theta_{\mathbf{v},\mathbf{u}}} \sum_{j=1}^{\sharp(\mathbb{A})} f(\mathbf{a}_j|\mathbf{a}_{h,k})}, & \text{if } \sum_{h\,:\,\mathbf{a}_{h,k} \in \Theta_{\mathbf{v},\mathbf{u}}} \sum_{j=1}^{\sharp(\mathbb{A})} f(\mathbf{a}_j|\mathbf{a}_{h,k}) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Probabilities $\hat{P}(\mathbf{a}_z|\Theta_{\mathbf{v},\mathbf{u}})$ generate a new transition matrix, where the kN-states are replaced by the classes of the kN-partitions. To simplify the notation, we use hereafter the symbol P to refer both to the transition probabilities P and to their estimates $\hat{P}$.

The introduction of the partitions of $\mathbb{A}^k$ and of an optimal selection procedure among them are the tools used in the present paper to the aim of reducing the cardinality of the state space $\mathbb{A}$ of $\mathbf{X}$ to best approximate $\mathbf{Y}$ with a Markov chain of order $k \geq 1$.

2.5 Optimization problem

The optimization problem takes into account two competing objectives: on the one side, we want bootstrapped series as "similar" as possible to the original sample $\mathcal{O}$; on the other side, we want to avoid that they are "too similar" to (eventually coinciding with) $\mathcal{O}$.
The optimization problem is constructed on the basis of the distance indicator $d_{\lambda_{\mathbf{v}}}$ and the multiplicity measure $m_{\lambda_{\mathbf{v}}}$ (see Eqs. (14) and (15), respectively, below). For a theoretical foundation of distance measures, like the distance indicator and the multiplicity measure, on information theory, refer to Online resource 1 - Appendix F. Here, a link between information loss (see Kolmogorov, 1965) and distance measures is presented and convincingly supports the choice of $d_{\lambda_{\mathbf{v}}}$ and $m_{\lambda_{\mathbf{v}}}$. The distance indicator proposed here penalizes the classes grouping one (or more) pairs of kN-states, say $\mathbf{a}_{h',k}$ and $\mathbf{a}_{h'',k}$, whose transition probabilities to the same N-state are very different. To keep this indicator low, a partition should group in any class kN-states having similar transition probabilities.
First of all, we need to introduce the distance $d_{h',h''}$ between two kN-states $\mathbf{a}_{h',k}$ and $\mathbf{a}_{h'',k}$:

$$d_{h',h''} := \sum_{z=1}^{\sharp(\mathbb{A})} \left| P(\mathbf{a}_z|\mathbf{a}_{h',k}) - P(\mathbf{a}_z|\mathbf{a}_{h'',k}) \right|.$$

Next, the distance within the classes of kN-partition $\lambda_{\mathbf{v}}$ can be defined as:

$$d_{\Theta_{\mathbf{v},\mathbf{u}}} := \max_{h',h'' : \mathbf{a}_{h',k},\mathbf{a}_{h'',k} \in \Theta_{\mathbf{v},\mathbf{u}}} d_{h',h''}.$$

Finally, the distance $d_{\lambda_{\mathbf{v}}}$ of the kN-partition $\lambda_{\mathbf{v}}$ is given by the average of the distances of its classes:

$$d_{\lambda_{\mathbf{v}}} := \frac{1}{C} \times \sum_{\mathbf{u}=1}^{\sharp(\lambda_{\mathbf{v}})} d_{\Theta_{\mathbf{v},\mathbf{u}}} \times \sharp\left(\Theta_{\mathbf{v},\mathbf{u}}\right), \tag{14}$$

where $C := \sum_{\mathbf{u}=1}^{\sharp(\lambda_{\mathbf{v}})} \sharp\left(\Theta_{\mathbf{v},\mathbf{u}}\right)$. The multiplicity measure $m_{\lambda_{\mathbf{v}}}$ of the kN-partition $\lambda_{\mathbf{v}}$ is:

$$m_{\lambda_{\mathbf{v}}} := \frac{\sqrt{B} - \sqrt{C}}{C - \sqrt{C}}, \tag{15}$$

where $B := \sum_{\mathbf{u}=1}^{\sharp(\lambda_{\mathbf{v}})} \left[\sharp\left(\Theta_{\mathbf{v},\mathbf{u}}\right)\right]^2$. Notice that a high multiplicity measure is typically associated to partitions consisting of few classes and where one or few of them have a larger number of kN-states than the others. Given the information loss theory of Kolmogorov, large classes tend to increase the information loss. The information loss of a class is maximized when its row (in the transition probability matrix) contains all equal values. Indeed this amounts to voiding that class of any conditioning power. So a high multiplicity measure allows partitions with large classes to be chosen. This, in turn, implies that the bootstrap method will frequently sample the next value using low informative classes, reducing the risk of the mechanical replication of part of (if not even all) the observed time series.

It can be shown that $d_{\lambda_{\mathbf{v}}} \in [0, 2]$ and $m_{\lambda_{\mathbf{v}}} \in [0, 1]$. Furthermore, the *singleton partition* -the one with cardinality $\sharp(\mathbb{A}^k)$- is associated to $d_{\lambda_{\mathbf{v}}} = m_{\lambda_{\mathbf{v}}} = 0$, while the *all-comprehensive partition* -the one with cardinality 1- exhibits the maximum value of $d_{\lambda_{\mathbf{v}}}$ (not necessarily 2) and $m_{\lambda_{\mathbf{v}}}$.

Given the previous properties of the distance indicator and of the multiplicity measure, the choice of partitions with a low distance indicator results in bootstrapped series "similar" to the original sample $\mathcal{O}$, while the choice of partitions with a high multiplicity measure avoids bootstrapped series "too similar" to $\mathcal{O}$. The trade-off between the distance indicator and the multiplicity measure motivates the choice of partitions through the optimization problem that we introduce in the following.

Definition 4 Let us consider $\gamma \in [0, 1]$, $k^* = 1, ..., M - 1$, and $\lambda^* \in \Lambda^{k^*}$.
We say that the couple (k^*, λ^*) is γ-optimal when it solves the following minimization problem:

$$\min_{(k,\lambda) \in \{1,\dots,M-1\} \times \Lambda^k} d_\lambda \tag{16}$$

$$\text{s.t.} \quad m_\lambda \geq \gamma. \tag{17}$$

Remark 6 When needed and to put in evidence the dependence from γ, the γ-optimal couple (k^*, λ^*) will be indicated as $(k^*(\gamma), \lambda^*(\gamma))$.

Given the multiplicity constraint threshold $\gamma \in [0, 1]$ and the solution $(k^*(\gamma), \lambda^*(\gamma))$ of the minimization problem in Definition 4, we will replicate the original sample $\mathcal{O}$ through a Markov chain of order $k^*(\gamma)$ described by a transition probability matrix of order $k^*(\gamma)$ estimated by means of Eq. (13). The matrix is based on $k^*(\gamma)N$-partition $\lambda^*(\gamma)$.
As it will turn out to be useful for the interpretation of the results of the optimization problem, we introduce the *efficient frontier* associated to the optimal kN-partitions $\lambda^*(\gamma)$ as γ is let vary in $[0, 1]$ and k is fixed to a given value.

Definition 5 Set $k = 1, ..., M-1$ to a chosen value. The efficient frontier $\mathcal{F}_k$ related to the minimization problem (16)-(17) is:

$$\mathcal{F}_k := \bigcup_{\gamma \in [0,1]} \left\{ (m_{\lambda^*(\gamma)}, d_{\lambda^*(\gamma)}) \in [0,1] \times [0,2] \right\},$$

where $\lambda^*(\gamma)$ is the solution of the problem:

$$\min_{\lambda \in \Lambda^k} d_\lambda \tag{18}$$

$$\text{s.t.} \quad m_\lambda \geq \gamma. \tag{19}$$

2.6 Outline of the method

In the following, we provide an intuitive and short description of the method we propose, which is outlined in Algorithm 1.

Starting from a realization of a multivariate discrete time continuous-valued stochastic process, i.e. an observed multivariate time series, a regression analysis is used to remove possible deterministic elements, such as trend and seasonality (Step 1 in Algorithm 1). The removal of these deterministic elements makes the series weakly stationary. Other unknown deterministic components possibly remaining in the residuals will be captured, along with other dependencies among the data, by the transition probability matrix. The support of each component of the stochastic process is initially partitioned into intervals, where each interval corresponds to a state (Step 2). Multivariate states are obtained by a Cartesian product of the latter states (Step 3). Consider multivariate and multidimensional states, i.e. sequences of $k \geq 1$ multivariate states. Based on the frequencies of transitions from multivariate and multidimensional states to multivariate states, the transition probabilities of the multivariate Markov chain of order k are estimated (Step 4). The approximation of the stochastic process is found by aggregating the multivariate and multidimensional states based on the distance between their transition probabilities. Specifically, we formulate the problem of finding the approximation as an optimization problem, where a distance indicator is minimized subject to a constraint on a multiplicity measure. Each feasible solution of the optimization problem is required to fulfill a contiguity constraint. Varying the right-hand side of the multiplicity constraint, the corresponding optimal solutions define an efficient frontier. We build an approximation of the efficient frontier embedding a Tabu Search heuristic into a classical ϵ-constraint method (Step 5). A point on the approximation of the efficient frontier is chosen (Step 6). Based on the aggregation of the multivariate and multidimensional states corresponding to the selected point, the corresponding transition probability matrix is estimated[3] (Step 7). This matrix defines a Markov chain of order k, which approximates the stochastic process. The performance of this approximation is validated via bootstrapping multivariate series (Step 8). To generate a bootstrapped multivariate series, an initial sequence of k multivariate states is selected from any of those defining the rows of the transition probability matrix. The probability distribution in the corresponding row is then used to extract randomly the next multivariate state of the Markov chain. After updating the sequence of k multivariate states (discarding the oldest and including the new one), the generation can be iterated. Finally, some statistics are calculated on the bootstrapped multivariate series and are compared with the corresponding statistics computed on the observed time series.

[3] To the sake of brevity, we do not report here an example of such transition probability matrix. The interested reader is referred to the illustrative example in Subsection 2.3 of Cerqueti et al. (2013) for a univariate case. Despite the extension to the present multivariate setting would be straightforward, its representation would require involved notation without adding any new insights.

Algorithm 1 General Scheme of the Method.

Input: an observed (multivariate) time series.
Output: a given number of bootstrapped (multivariate) series.

1: A regression analysis is used to remove possible deterministic components, such as trend and seasonality.
2: Segment the support of each component of the stochastic process into initial intervals, or states.
3: Build the multidimensional discrete support, $\mathbb{A}$, of the observed time series using the previous states.
4: Use Eq. (9) to estimate the transition probability matrix of the multivariate Markov chain of order $k \geq 1$ with support $\mathbb{A}$.
5: Build an approximation of the efficient frontier by means of Algorithm 2.
6: Choose a point on the approximation of the efficient frontier.
7: Estimate the transition probability matrix corresponding to the chosen point according to Eq. (13).
8: Use the bootstrap method in Subsection 3.2 of Cerqueti et al. (2013).

3 Solution Strategy

Due to the computational difficulties of determining the optimal solution of problem (18)-(19), we develop a heuristic algorithm to tackle the problem. The heuristic algorithm is based on the Tabu Search framework. We assume that the reader is familiar with the Tabu Search framework and we refer to Glover and Laguna (1997) for a detailed description of the metaheuristic.

The Tabu Search algorithm is composed of two main phases, and then it is referred to as the *Two-Phase Tabu Search*. *Phase 1* (*TSP1*, hereafter) is designed to explore the set Λ^k of kN-partitions, i.e. the set collecting all the contiguous multivariate and multidimensional partitions, to the goal of (heuristically) solving problem (18)-(19).

Nevertheless, as the problem proposed in this paper is new in the literature and due to the difficulties of solving it to optimality, we design a second phase where the partitions of $\mathbb{A}^k$ that violate the contiguity constraint are evaluated, as well. Particularly, *Phase 2* (*TSP2*, hereafter) is designed to be performed after TSP1 and considers both kN-partitions in Λ^k and partitions of $\mathbb{A}^k$ that violate the contiguity constraint (hereafter the procedure is referred to as *TSP1+TSP2*). This design of the algorithm allows the researcher interested in exploring only the kN-partitions in Λ^k, that is the scope of this paper, to perform only TSP1. Nevertheless, it allows us to assess the effectiveness of the Two-Phase Tabu Search by comparing its performance with the results reported in Cerqueti et al. (2013), where the contiguity constraint is not considered. Besides, this design allows us to evaluate the impact of the contiguity constraint on the distance indicator d_λ and the multiplicity measure m_λ by comparing the output of procedure TSP1 (partitions with contiguity constraint) with the output of procedure TSP1+TSP2 (partitions where the contiguity constraint is relaxed).

3.1 Building an approximation of the efficient frontier

In Definition 5, the efficient frontier $\mathcal{F}_k$ is defined as a set of solutions of optimization problem (18)-(19) as γ is let vary in $[0, 1]$ and k is fixed to a given value. As we solve problem (18)-(19) by means of a heuristic algorithm, the outcome is an approximation of the efficient frontier $\mathcal{F}_k$. Therefore, we introduce the following definition that holds for $\gamma \in [0, 1]$ and for a given value of k.

Definition 6 Set $k = 1, ..., M - 1$. An approximation of the efficient frontier $\mathcal{F}_k$ is the set:

$$\mathcal{AF}_k := \bigcup_{\gamma \in [0,1]} \left\{ (m_{\lambda^H(\gamma)}, d_{\lambda^H(\gamma)}) \in [0, 1] \times [0, 2] \right\},$$

where $\lambda^H(\gamma)$ is a heuristic solution of problem (18)-(19), for a generic heuristic algorithm. Furthermore, set $\mathcal{AF}_k$ is composed of mutually non-dominated points, i.e. there does not exist any pair of heuristic solutions $\lambda_1^H(\gamma)$ and $\lambda_2^H(\gamma)$ such that $m_{\lambda_1^H(\gamma)} \geq m_{\lambda_2^H(\gamma)}$, $d_{\lambda_1^H(\gamma)} \leq d_{\lambda_2^H(\gamma)}$ and at least one inequality is strict.

To build the approximated efficient frontier $\mathcal{AF}_k$, we embed the Two-Phase Tabu Search into an ϵ-constraint method. The ϵ-constraint method is one of the most known approaches for the solution of multi-objective optimization problems (see, e.g., Chankong and Haimes, 1983; Miettinen, 1999). The method generates a sequence of single-objective optimization problems, referred to as *ϵ-constraint problems*, by transforming all the objective functions but one into constraints.

Let us fix $i \in \mathbb{N}$ and consider the following ϵ-constraint formulation of problem (18)-(19):

$$\min_{\lambda \in \Lambda^k} d_\lambda \tag{20}$$

$$\text{s.t.} \quad m_\lambda \geq \gamma_i, \tag{21}$$

where $\gamma_i \in [0, 1]$. The idea is to construct a sequence of ϵ-constraint problems (20)-(21) where the right-hand side of constraint (21) is iteratively updated as follows: once problem (20)-(21) is solved for a given value of γ_i, we set γ_i equal to $m_{\lambda_i^H}$, and then we set $\gamma_{i+1} := \gamma_i + \Delta$, where Δ is a given parameter. Each ϵ-constraint problem (20)-(21) is then solved by means of the Two-Phase Tabu Search described below. We denote as λ_i^H the best-known solution found by our heuristic algorithm for the i-th ϵ-constraint problem (20)-(21).

The general scheme of the algorithm proposed to construct the approximated efficient frontier $\mathcal{AF}_k$ is sketched in Algorithm 2.

Algorithm 2 Procedure: General Scheme to Build $\mathcal{AF}_k$.

Input: an instance of problem (18)-(19) and parameter Δ.
Output: an approximated efficient frontier $\mathcal{AF}_k = \{(m_{\lambda_1^H}, d_{\lambda_1^H}), ..., (m_{\lambda^H_{\sharp(\mathcal{AF}_k)}}, d_{\lambda^H_{\sharp(\mathcal{AF}_k)}})\}$.

1: Set $(m_{\lambda_1^H}, d_{\lambda_1^H}) := (0, 0)$ and insert it in $\mathcal{AF}_k$.
2: Set $i := 2$, $\gamma_i := \Delta$.
3: **while** $\gamma_i \leq 1$ **do**
4: solve problem (20)-(21) by means of Algorithm 2;
5: let λ_i^H be the returned solution;
6: insert $(m_{\lambda_i^H}, d_{\lambda_i^H})$ in $\mathcal{AF}_k$ after $(m_{\lambda^H_{i-1}}, d_{\lambda^H_{i-1}})$;
7: set $\gamma_i := m_{\lambda_i^H}$;
8: set $\gamma_{i+1} := \gamma_i + \Delta$ and $i := (i + 1)$.
9: **end while**
10: Remove dominated points from $\mathcal{AF}_k$ (if any).

Handling the dominated points. It is worth highlighting that some dominated points might be generated solving the sequence of ϵ-constraint problems (20)-(21) with any heuristic algorithm. Indeed, let us assume that the best-known solution found by a given heuristic for the i-th ϵ-constraint problem is λ_i^H. It might happen that the best-known solution λ_{i+1}^H to the $(i+1)$-th ϵ-constraint problem is such that $d_{\lambda^H_{i+1}} \leq d_{\lambda_i^H}$ (with $m_{\lambda^H_{i+1}} \geq m_{\lambda_i^H} + \Delta$ by construction). Hence, point $(m_{\lambda^H_{i+1}}, d_{\lambda^H_{i+1}})$ dominates point $(m_{\lambda_i^H}, d_{\lambda_i^H})$. Therefore, we include in our algorithm a post-processing procedure that scans all the points included in set $\mathcal{AF}_k$ and removes the dominated ones (Step 10 in Algorithm 2).

3.2 TSP1: Exploring contiguous partitions

The description of TSP1 is organized as follows. Firstly, we define the neighborhood structure as it represents the core of the algorithm. Subsequently, each component of TSP1 is detailed.

Neighborhood structure. The introduction of the multivariate assumption for the stochastic process and of the contiguity constraint for the states impose the following redefinition of the neighborhood structure introduced in Cerqueti et al. (2013).

Consider an incumbent solution $\lambda \in \Lambda^k$ of problem (20)-(21). At each iteration of procedure TSP1, one time lag $\bar{w} \in \{1, ..., k\}$ is first selected, as detailed below, and then the analysis moves from the incumbent solution λ to a neighboring solution λ'. Given the incumbent solution $\lambda \in \Lambda^k$, consider its N-partition $\lambda_{\bar{w}} = (\lambda_{\bar{w}}^{(1)}, \lambda_{\bar{w}}^{(2)}, ..., \lambda_{\bar{w}}^{(n)}, ..., \lambda_{\bar{w}}^{(N)}) \in \Lambda$ at time lag $\bar{w}$. We define two types of neighborhood. The first type, denoted as $\mathcal{N}_{P1}\binom{n}{\bar{w}}$, is a set containing all the UUPs (see Remark (5)) $\lambda_{\bar{w}}^{(n)\prime}$ that can be obtained performing local changes only on UUP $\lambda_{\bar{w}}^{(n)}$. The second type, denoted as $\mathcal{N}_{P1}\binom{n,\bar{n}}{\bar{w}}$, is a set containing all pairs of UUPs $(\lambda_{\bar{w}}^{(n)\prime}, \lambda_{\bar{w}}^{(\bar{n})\prime})$ that can be obtained performing combined local changes on UUPs $\lambda_{\bar{w}}^{(n)}$ and $\lambda_{\bar{w}}^{(\bar{n})}$.
Specifically, given UUP $\lambda_{\bar{w}}^{(n)}$, the neighborhood $\mathcal{N}_{P1}\binom{n}{\bar{w}}$ is defined by the following moves:

- (q, n)-*relocation:* q contiguous states $\{a_l^{(n)}, ..., a_{l+q-1}^{(n)}\}$ are removed from their origin class and assigned to an existing and not empty adjacent class;
- (q, n)-*creation:* q contiguous states $\{a_l^{(n)}, ..., a_{l+q-1}^{(n)}\}$ are removed from their origin class and assigned to a new adjacent class containing them as the only elements.

In order to guarantee that the contiguity constraint is fulfilled after performing the aforementioned moves, the q contiguous states $\{a_l^{(n)}, ..., a_{l+q-1}^{(n)}\}$ are jointly taken from the origin class and then jointly assigned to an adjacent class. Additionally, if the number of states belonging to a class is smaller than q, a (q, n)-relocation is performed by moving all the available states in the class. Finally, if the number of states belonging to a class is equal to q, no (q, n)-creation is performed as it would not change the current UUP.

Example 1 As an illustrative example, consider UUP $\lambda_{\bar{w}}^{(n)} = \{\{1, 2\}, \{3\}\}$. For $q = 1$, the available $(1, n)$-relocations lead to UUPs $\lambda_{\bar{w}}^{(n)\prime} = \{\{1\}, \{2, 3\}\}$ (state 2 is moved from its class to the adjacent one) and $\lambda_{\bar{w}}^{(n)\prime\prime} = \{\{1, 2, 3\}\}$ (state 3 is assigned to the adjacent class). For $q = 2$, one $(2, n)$-relocation is available leading again to UUP $\lambda_{\bar{w}}^{(n)\prime\prime} = \{\{1, 2, 3\}\}$ (states 2 and 3 are assigned to the adjacent class or, similarly, state 3 is assigned to the adjacent class). Furthermore, one $(1, n)$-creation is available leading to UUP $\lambda_{\bar{w}}^{(n)\prime\prime\prime} = \{\{1\}, \{2\}, \{3\}\}$ (state 1, or, alternatively, state 2 is assigned to a new adjacent class). Note that the $(2, n)$-creation consisting in removing states 1 and 2 from their origin class and assigning them to a new adjacent class is not considered, as it does not alter UUP $\lambda_{\bar{w}}^{(n)}$.

Besides the previous moves which modify only the single component $\lambda_{\bar{w}}^{(n)}$ of N-partition $\lambda_{\bar{w}}$ at time lag $\bar{w}$ in the incumbent solution λ, algorithm TSP1 also considers the following moves that change simultaneously two components of $\lambda_{\bar{w}}$, say $\lambda_{\bar{w}}^{(n)}$ and $\lambda_{\bar{w}}^{(\bar{n})}$, with $n, \bar{n} = 1, ..., N$ and $n \neq \bar{n}$. In particular, the following moves defining neighborhood $\mathcal{N}_{P1}\binom{n,\bar{n}}{\bar{w}}$ are considered:

- *move 1:* performs a (q, n)-relocation and a $(q, \bar{n})$-relocation;
- *move 2:* performs a (q, n)-relocation and a $(q, \bar{n})$-creation;
- *move 3:* performs a (q, n)-creation and a $(q, \bar{n})$-relocation;
- *move 4:* performs a (q, n)-creation and a $(q, \bar{n})$-creation.

The rationale of introducing moves 1-4 is related to the fact that restricting to the (q, n)-relocations and (q, n)-creations has proved to be inefficient in some cases. On the one hand, a small neighborhood is easier to explore thoroughly but, on the other hand, it might worsen the quality of the solution found when N-partitions consist of many components. In this case, the interaction between changes in the N-partitions of two or more components is neglected. Note that further moves involving more than two components could be introduced. Nevertheless, this would expand excessively the neighborhood making its exploration particularly slow.

Initial feasible solution. Note that the best-known solution found for ϵ-constraint problem i is not feasible for ϵ-constraint problem $i + 1$. Indeed, let $m_{\lambda_i^H}$ be the value of the multiplicity measure for ϵ-constraint problem i. The multiplicity constraint for problem $i + 1$ is $m_\lambda \geq m_{\lambda_i^H} + \Delta$ (see Steps 7 and 8 in Algorithm 2), making the best-known solution λ_i^H unfeasible for ϵ-constraint problem $i + 1$.

The initial solution procedure takes as input the best-known solution of the previous ϵ-constraint problem found by TSP1[4] and performs a sequence of $(1,n)$-relocations until the feasibility of the current solution is restored. Indeed, performing a $(1,n)$-relocation tends to increase the value of m_λ (and of d_λ as well). At each iteration, the $(1,n)$-relocation determining the largest increase of m_λ is selected. Once the first feasible solution is found, the procedure to compute the initial solution stops.

Random selection of time lags. As mentioned above, at each iteration of TSP1 one time lag is randomly selected. To this aim, we use the heuristic procedure proposed in Cerqueti et al. (2013). In few words, the time lag is selected according to a discrete probability distribution function giving higher priority to those time lags that bring a key information to drive the evolution of the process. The interested reader is referred to the quoted paper for further details.

Tabu lists. We use $k \times N$ tabu lists $TL_w^{(n)}$, $w = 1, ..., k$ and $n = 1, ..., N$, i.e. one tabu list for each time lag and component. When UUP $\lambda_{\bar{w}}^{(n)}$ is selected to form the new incumbent solution, the selection of UUP $\lambda_{\bar{w}}^{(n)}$ at time lag $\bar{w}$ becomes forbidden (tabu) for τ iterations (see Online Resource 1 - Appendix D for further details). Note that when one of moves 1-4 is performed, two UUPs become temporarily tabu.

Aspiration criterion. In order to avoid that feasible solutions of excellent quality are not considered because of the tabu status for some of their UUPs, we introduce the following standard aspiration criterion. A UUP belonging to a tabu list can be selected (i.e. its tabu status is revoked) if its selection leads to a feasible solution better than the best-known solution λ^H encountered so far.

Diversification strategy. Using tabu lists prevents short-term cycling, i.e. visiting the same feasible solution in two successive or very close iterations, but it is not adequate to avoid long-term cycling, i.e. being trapped in local optima. Hence, a wider exploration of the solution space has to be encouraged. To this aim, we design a diversification strategy in order to move the search to a different portion of the solution space.
Firstly, it is worth noticing that, given UUP $\lambda_{\bar{w}}^{(n)}$, the UUPs obtained by performing (q,n)-relocations and (q,n)-creations tend to be "more different" with respect to $\lambda_{\bar{w}}^{(n)}$ when q increases. Consider the following example.

Example 2 Let $\lambda_{\bar{w}}^{(n)} = \{\{1,2,3\},\{4,5\}\}$ be a UUP of N-partition $\lambda_{\bar{w}}$ at time lag $\bar{w}$ in the incumbent solution λ. For $q = 1$, the $(1,n)$-relocations available lead to: $\lambda_{\bar{w}}^{(n)\prime} = \{\{1,2\},\{3,4,5\}\}$ and $\lambda_{\bar{w}}^{(n)\prime\prime} = \{\{1,2,3,4\},\{5\}\}$. Conversely, the $(2,n)$-relocations available yield: $\lambda_{\bar{w}}^{(n)\prime\prime\prime} = \{\{1\},\{2,3,4,5\}\}$ and $\lambda_{\bar{w}}^{(n)\prime\prime\prime\prime} = \{\{1,2,3,4,5\}\}$. Trivial examples can be found for the (q,n)-creation and for moves 1-4.

Therefore, in a first attempt, the value of q is increased by 1 every time that a maximum number of iterations without improvement (parameter $iterNoImprP1_{max}$) have been consecutively performed. Secondly, in order to move significantly the search to different regions of the solution space, a *jump* is performed once that a maximum value of q is exceeded (parameter q_{max}). The procedure designed to perform jumps in TSP1 is referred to as *JumpingP1* and is sketched in Algorithm 4. Procedure JumpingP1 is an adaptation of the corresponding procedure introduced in Cerqueti et al. (2013) to the multivariate assumption for the stochastic process and to the contiguity constraint for the states.

Stopping criterion. Algorithm TSP1 stops when $iterP1_{max}$ iterations have been performed.

Tabu Search algorithm. In Online Resource 1 - Appendix A, a pseudo-code for procedure TSP1 is provided in Algorithm 2 along with a description. We highlight that the best-known solution λ^H found at the end of one execution of TSP1 is provided as input of procedure TSP1 called to solve the next ϵ-constraint problem in the sequence.

[4] The singleton partition is provided as input of TSP1 for the first ϵ-constraint problem.

3.3 TSP2: Extending the search also to non-contiguous partitions

This subsection is devoted to the description of TSP2. To the sake of brevity, we here highlight only the differences with respect to algorithm TSP1 described above.

Algorithm TSP2 takes as input the best-known solution $\lambda^H \in \Lambda^k$ found performing TSP1. The neighborhood structure is different from that described above for TSP1 as, in TSP2, non-contiguous partitions of $\mathbb{A}^k$ are explored too. At each iteration of TSP2, given an incumbent solution λ of $\mathbb{A}^k$, which does not necessarily belong to the set of contiguous multivariate and multidimensional partitions Λ^k, select partition $\lambda_{\bar{w}}$, i.e. the time lag $\bar{w}$ component of λ, with $\bar{w} \in \{1, ..., k\}$, and then partition $\lambda_{\bar{w}}^{(n)}$, i.e. the n-th component of $\lambda_{\bar{w}}$, with $n = 1, ..., N$. The neighborhood $\mathcal{N}_{P2}(\lambda_{\bar{w}}^{(n)})$ of $\lambda_{\bar{w}}^{(n)}$ is defined by the following moves:

- $(q, n)_{P2}$-*relocation:* q states (not necessarily contiguous) are removed from their origin class and assigned to a not empty (and not necessarily adjacent) class;
- $(q, n)_{P2}$-*creation:* q states (not necessarily contiguous) are removed from their origin class and assigned to a new (not necessarily adjacent) class containing them as the only elements;
- (q, n)-*swap:* q states (not necessarily contiguous) are swapped with q states (not necessarily contiguous) belonging to another class.

Algorithm TSP2 stops when $iterP2_{max}$ iterations have been performed. In Online Resource 1 - Appendix A a pseudo-code of TSP2 is sketched in Algorithm 3.

4 Application to Empirical Data

4.1 Data description

Our bootstrap method is applied to two observed multidimensional time series. In particular, we consider here a trivariate series composed by the prices and volumes of McDonald's and the prices of Dow Jones Industrial Average and a bivariate series composed by the "Mibel Spanish Electric System Arithmetic Average Price" and the corresponding total volume. The trivariate series covers the trading days on the New York Stock Exchange from January 2^{nd}, 2004 to December 31^{st}, 2012, and the prices of McDonald's are expressed in US dollars. The bivariate series spans the trading days from January 1^{st}, 2001 to May 6^{th}, 2010, and the prices are expressed in euros per MWh. The stock-index case consists of $T = 2265$ observations, while for the electricity series we have $T = 2439$.

Figures 1 and 2 show the two observed multidimensional time series. We can make some comments.

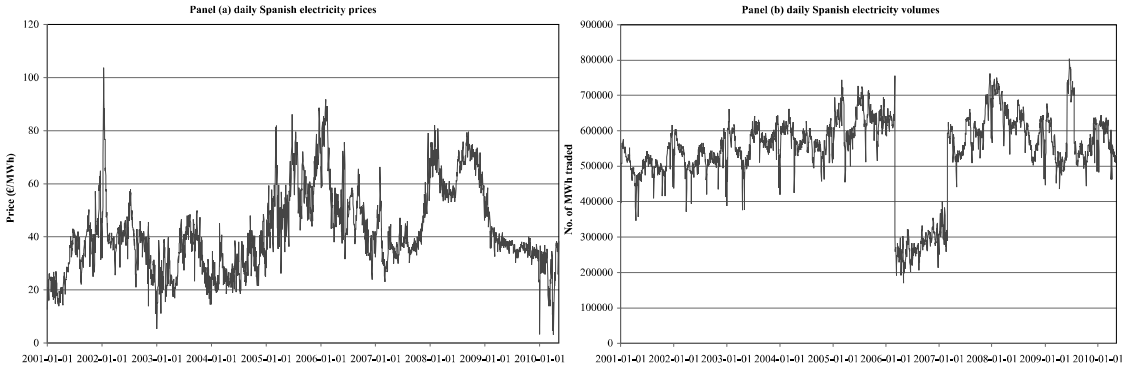

Fig. 1: Electricity case - observed time series of prices and volumes.

The original sample of electricity prices and volumes is characterized by the following features:
- a weekly and annual seasonality (the annual seasonality is more evident for volumes);
- a slightly positive trend of prices, a significant positive trend of volumes;
- stochastic volatility;
- nonlinear dependence of data;
- two clear regimes of prices: normal trading and occasional spiking periods.

Spikes are occasional, since they usually correspond to unexpected shortages on the supply side of the electricity system, or unexpected and temporary increases of the demand (e.g., sudden meteorological events driving to high consumption). Because of the joint presence of such features and nonlinear dependence between prices and volumes, the literature on time series of electricity prices and volumes usually considers them as "hard to model" cases. For a review of the difficulties in modeling electricity prices and volumes and the methods developed to solve them, see, for example, Bunn (2004), Huisman and Mahieu (2003), Weron et al. (2004), and Weron (2006).

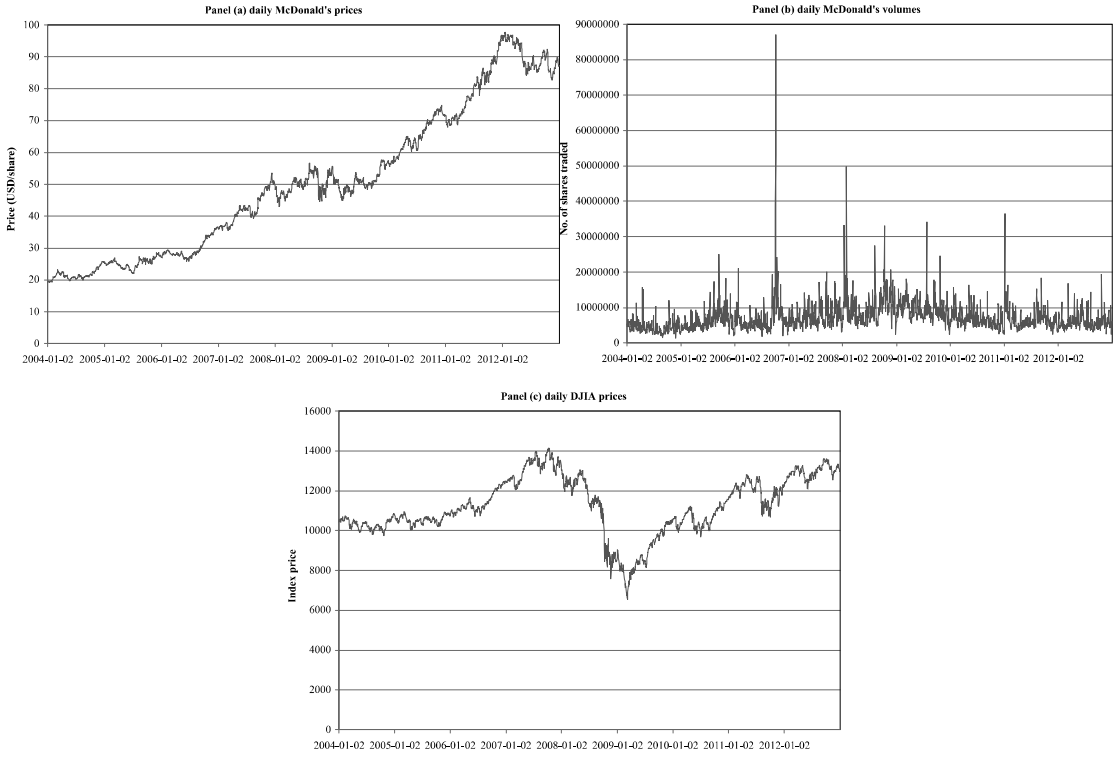

Fig. 2: Stock-index case - observed time series of prices and volumes.

About stock prices and volumes and index prices, the following features are commonly observed:
- stock volumes show mean reversion, volatility clustering, and spikes;
- stock and index prices appear more in line with the theory of market efficiency and seem to represent realizations of a geometric Brownian motion, even though the normality of stock and index return distributions has been regularly rejected in the empirical literature;
- the joint series of volumes and prices shows unclear causal relationships, so its bootstrapping can be considered as a "hard test" for bootstrapping purposes.

In the electricity case, an (exponential) weekly seasonal component as well as an (exponential) trend have been removed from the raw prices and volumes. Removing the weekly seasonality lets us reduce the order of the Markov chain below 7, which corresponds to a great reduction in the complexity of our problem. The removal of the trend component makes the series (quite) stationary. Of course both components have been added back to the bootstrapped series.
In the stock-index case, no seasonality has been removed, because such feature is less evident than in the electricity case, therefore only a trend has been estimated to the goal of making the series stationary. To gain further stationarity, an additional logarithmic transformation of the detrended series has been performed in the case of stock and index prices. Also in this case the trend component has been added back (for the stock and index prices after exponentiation of the bootstrapped series).

The estimation of exponential (rather than linear) components prevents the removal/reintroduction process from generating occasional negative prices (see Online Resource 1 - Appendix B for further details).

4.2 Preliminary segmentation of the support

As explained in the previous subsection, the data treatment applied to the bivariate and trivariate series to gain stationarity can be synthesized as follows:

- series of electricity prices: detrended and weekly deseasonalized;
- series of electricity volumes: detrended and weekly deseasonalized;
- series of stock volumes: detrended;
- series of stock prices: detrended and then transformed into series of logarithmic returns;
- series of index prices: detrended and then transformed into series of logarithmic returns.

After the data treatment, a preliminary segmentation of the supports of the continuous-valued stochastic processes generating the bivariate and trivariate series is performed (see Subsection 2.1). We make the following choice: 8 initial intervals for the *stock volumes* (*sv*), 6 for the *stock returns* (*sr*), and 6 for the *index returns* (*ir*) in the trivariate case, and 13 initial intervals for both the *electricity prices* (*ep*) and *electricity volumes* (*ev*) in the bivariate case. Thus, according to Eq. (5), the cardinality of $\mathbb{A}$ is $8 \times 6 \times 6 = 288$ 3-states in the trivariate case and $13 \times 13 = 169$ 2-states in the electricity case.
Such numbers of initial intervals correspond to an order of magnitude larger than the number of states commonly considered in the literature for the regimes of the stock-index and electricity cases (usually 2 or 3 for both the cases. See, e.g., Bühlmann, 1998 for the stock-index case and Huisman and Mahieu, 2003 for the electricity case).

The preliminary segmentation is performed through the minimum variance clustering procedure provided in Ward Jr. (1963). For details on this preliminary segmentation, see Online Resource 1 - Appendix C.

4.3 Transition probability matrices

The solution of optimization problem (16)-(17) for a given $\gamma \in [0, 1]$ consists of the γ-optimal pair $(k^*(\gamma), \lambda^*(\gamma))$, i.e. the optimal order $k^*(\gamma) \in \{1, ..., M-1\}$ and the optimal $k^*(\gamma)N$-partition $\lambda^*(\gamma) \in \Lambda^{k^*}$. However, to the sake of reducing the complexity of the problem, we restrict the search of the solution to the space Λ^k and fix k. Notice that the solutions found under such restriction still tell us much about the "relevance" of the different time lags up to k.
Observe that we can fix different values of k for each component of our multivariate Markov chain, although the (overall) order of the Markov chain will be the highest value of k. In particular, the order of the Markov chain is 2 for the components represented by electricity prices, stock returns, and index returns, while it is set equal to 1 for the electricity and stock volumes. As a consequence,

the order of the multivariate Markov chain will be 2 for both the bivariate and the trivariate cases. In the present application, next to a reduction of the complexity of the problem, a reason to fix k is that the data treatment applied to the observed multivariate time series (trend and weekly seasonality removal and logarithmic return calculation) should reduce the need to look back far in time.

Eq. (9) has been used to estimate $\mathcal{M}_T$ and $\mathcal{M}_B$, i.e. the transition probability matrices of order 2 for the trivariate (T) case and for the bivariate (B) case, respectively. The matrices are available from the authors upon request.
The sets of 2-trajectories $\mathcal{O}_2$ (see Eq. (2)) comprise 2437 elements for the electricity instance and 2262 elements for the stock-index one[5]. Each set $\mathcal{O}_2$ has been rewritten by means of the corresponding $2N$-states of $\mathbb{A}^2$ giving $\mathcal{A}_2$ ($N = 2$ in the electricity case, while $N = 3$ in the stock-index case). About 7.5% of the $2N$-states of $\mathbb{A}^2$ in the electricity case has been observed to evolve to the same 2-state. We call these $2N$-states deterministic. In the stock-index case, the percentage of deterministic $2N$-states is about 20%. The remaining observed $2N$-states are called probabilistic (2255 for the bivariate case and 1834 for the trivariate case).
$\mathcal{M}_T$ consists of 725 rows, 304 referred to probabilistic $2N$-states and 421 to deterministic ones, while $\mathcal{M}_B$ consists of 472 rows, 305 referred to probabilistic $2N$-states and 167 deterministic ones.
The $2N$-states of $\mathbb{A}^2$ not observed in $\mathcal{A}_2$ are neglected, as their estimated transition probabilities are null. For bootstrapping purposes, deterministic $2N$-states need not be aggregated to other $2N$-states, therefore the optimization problem (16)-(17) is applied only to the set of probabilistic $2N$-states. As a result of these settings, all the admissible kN-partitions λ are characterized by three kinds of classes:

- the unique class collecting the unobserved $2N$-states,
- as many classes as the different observed deterministic $2N$-states,
- the classes aggregating the observed probabilistic $2N$-states.

5 Results and Discussion

5.1 Analysis of the distributional properties

We evaluate the performance of our bootstrap method by choosing two heuristic solutions, λ_B^H and λ_T^H, respectively for the bivariate and the trivariate case. Each solution has been chosen as follows: select the pair of adjacent points of the approximations of the efficient frontier with the largest slope. Select the point of that pair with the smallest d_λ and m_λ. Such point shows that an additional improvement with respect to the multiplicity measure corresponds to a large loss in terms of distance indicator. For a visual understanding of how a solution is selected, consider, as an example, the red circles on the efficient frontiers in Figure 3. The partitions so selected have multiplicity measure and distance indicator equal to 0.366 and 1.507, for the electricity case, and to 0.298 and 1.763, for the stock-index case (see Table 14 in Online Resource 1 - Appendix E). These partitions are expected to generate diversified bootstrapped series, which could represent a varied enough set of scenarios to test the effectiveness of the procedure.
In particular, we generate 2 sets of 5000 bootstrapped series, one set for each case. Each bootstrapped series includes $\ell = 2439$ trading days for the bivariate case and $\ell = 2265$ trading days for the trivariate one. The bootstrap method used to generate the 5000 bivariate and trivariate series is described in Subsection 3.2 of Cerqueti et al. (2013), which the interested reader is referred to. The lengths of the bootstrapped series are equal to the lengths of the corresponding observed time series (net of the values required to initialize the procedure) to allow for a fairer comparison (see Subsection 3.2 of Cerqueti et al., 2013, for the initialization steps of the bootstrap method).

[5] These two numbers exclude the last 2-trajectories, because they do not evolve to any realization of the process.

To assess the quality of the method, we analyze the statistical properties of the bootstrapped series and compare them with the ones of the observed time series. To this goal, we calculate the following statistics on each bootstrapped series:

1. average, standard deviation, skewness, kurtosis, minimum, maximum,

2. autocorrelation at lag k, $k = 1, 2, 3$,

3. linear regression slope, b, with $\hat{x}_t = a + bt + g_t$, $t = 1, ..., \ell$, where $\hat{x}_t$ stands for the t-th value of a bootstrapped series of prices or volumes.

Moreover, to analyse the power of the method to capture the cross-dependencies among the components of the multivariate bootstrapped series, we consider the following two simple vector autoregression models of order 1, for the stock-index case:

$$\begin{cases} \hat{sv}_t = a_1 + a_2\hat{sp}_t + a_3\hat{ip}_t + a_4\hat{sv}_{t-1} + g_{sv,t} \\ \hat{sp}_t = b_1 + b_2\hat{ip}_t + b_3\hat{sv}_t + b_4\hat{sp}_{t-1} + g_{sp,t} \\ \hat{ip}_t = c_1 + c_2\hat{sp}_t + c_3\hat{sv}_t + c_4\hat{ip}_{t-1} + g_{ip,t} \end{cases}, t = 1, ..., \ell \tag{22}$$

where $\hat{sv}_t$, $\hat{sp}_t$, and $\hat{ip}_t$ represent the t-th value of a bootstrapped series of stock volumes, stock prices, and index prices, respectively, and the following model for the electricity case:

$$\begin{cases} \hat{ep}_t = d_1 + d_2\hat{ev}_t + d_3\hat{ep}_{t-1} + g_{ep,t} \\ \hat{ev}_t = f_1 + f_2\hat{ep}_t + f_3\hat{ev}_{t-1} + g_{ev,t} \end{cases}, t = 1, ..., \ell \tag{23}$$

where $\hat{ep}_t$ and $\hat{ev}_t$ represent the t-th value of a bootstrapped series of electricity prices and electricity volumes, respectively. We therefore also calculate:

4. autoregression coefficients for the two previous models.

The statistics in 1. are concerned with the distribution of prices and volumes, while the statistics from 2. to 4. are more concerned with the dynamic structure of the series. The autocorrelation at lag 3 is observed to check if the similarity between the observed time series and the bootstrapped series is kept beyond the order $k = 2$.

Tables 1-3 and 4-5 report (for the trivariate and the bivariate case, respectively) the 5^{th} and 95^{th} percentiles of the distributions of the mentioned statistics, together with the actual value of the observed time series. We also report the percentile rank, i.e. the percentage of cases for which the statistic value is smaller than or equal to the value of the observed time series.

5.1.1 Stock-index case

Tables 1, 2, and 3 report the statistics for the stock-index case. In particular, we show the 5^{th} and the 95^{th} percentiles of the bootstrap distributions of some statistics calculated on the 5000 series generated from heuristic solution λ_T^H. We also show the percentile ranks of the observed time series.

Table 1: Percentiles and percentile ranks of the observed daily McDonald's volumes with respect to the bootstrap distributions of some statistics calculated on 5000 scenarios.

Statistics	5^{th} percentile	95^{th} percentile	Value of original sample	Percentile rank of original sample value
Average	6,730,960.772	7,564,220.589	7,193,208.794	66
Standard dev.	3,157,406.854	5,948,913.693	4,105,352.715	66
Skewness	1.872	8.639	5.734	65
Kurtosis	6.456	130.564	79.457	68
Minimum	1,344,005.926	1,973,540.942	1,337,000.000	4
Maximum	33,284,396.275	101,221,682.830	86,981,300.000	71
Autocorr. at lag 1	0.542	0.660	0.611	56
Autocorr. at lag 2	0.378	0.544	0.466	61
Autocorr. at lag 3	0.279	0.482	0.380	57
Lin. regr. slope	138.332	997.487	369.670	23
Autoregr. coeff. a_1	1,725,362.023	4,753,715.870	4,684,954.817	94
Autoregr. coeff. a_2	−6,226.651	27,858.678	5,555.865	41
Autoregr. coeff. a_3	−205.723	60.702	−187.065	6
Autoregr. coeff. a_4	0.531	0.653	0.602	57

Table 2: Percentiles and percentile ranks of the observed daily McDonald's prices with respect to the bootstrap distributions of some statistics calculated on 5000 scenarios.

Statistics	5^{th} percentile	95^{th} percentile	Value of original sample	Percentile rank of original sample value
Average	25.854	85.976	50.531	57
Standard dev.	5.932	57.117	22.951	54
Skewness	−0.243	1.351	0.435	37
Kurtosis	−1.450	1.239	−0.970	34
Minimum	14.494	19.660	19.130	64
Maximum	42.667	226.801	97.790	49
Autocorr. at lag 1	0.996	0.999	0.999	83
Autocorr. at lag 2	0.992	0.998	0.997	83
Autocorr. at lag 3	0.989	0.997	0.996	82
Lin. regr. slope	0.004	0.057	0.024	57
Autoregr. coeff. b_1	−0.627	0.151	−0.070	49
Autoregr. coeff. b_4	0.995	0.997	1.001	41

Table 3: Percentiles and percentile ranks of the observed daily DJIA prices with respect to the bootstrap distributions of some statistics calculated on 5000 scenarios.

Statistics	5^{th} percentile	95^{th} percentile	Value of original sample	Percentile rank of original sample value
Average	5,797.459	16,172.161	11,239.488	67
Standard dev.	974.561	4,035.718	1,455.575	28
Skewness	−0.609	1.331	−0.356	10
Kurtosis	−1.435	1.591	−0.160	73
Minimum	2,687.604	10,131.350	6,547.050	50
Maximum	10,774.554	24,367.306	14,164.530	56
Autocorr. at lag 1	0.992	0.999	0.996	31
Autocorr. at lag 2	0.984	0.998	0.992	33
Autocorr. at lag 3	0.978	0.997	0.989	32
Lin. regr. slope	−2.623	3.301	0.413	70
Autoregr. coeff. c_1	−0.781	128.955	46.353	60
Autoregr. coeff. c_2	−0.281	1.255	0.190	66
Autoregr. coeff. c_4	0.986	0.999	0.995	40

From Tables 1, 2, and 3, we can make the following remarks:

- all the statistics computed for the observed time series take values between the 5^{th} and the 95^{th} percentiles, except for the minimum of McDonald's volumes ranked in the 4^{th} percentile of the corresponding distribution,

- the previous observation applies in particular to the autocorrelation coefficients at lag 3, which capture an auto-dependence beyond the order 2 of the Markov chain,

- one could expect the statistics computed for the observed time series to fall close to the 50^{th} percentile of the corresponding bootstrap distribution. However, such intuitive stance has to be restrained if the observed time series is characterized by rare events, such as jumps or spikes. To be more precise, consider the maximum of the observed volumes of McDonald's: its percentile rank is 71. This score could be acceptable, as the observed volumes show a strong spike at the beginning of 2007 (see Panel (b) of Figure 2). Indeed, the 5000 bootstrapped series need not reproduce regularly such a high maximum (such regularity would contrast with the definition of rare event). So the linear regression slope of McDonald's volumes is low (belonging to the 23^{rd} percentile), since the majority of bootstrapped series, without such a spike, take a definitely more positive slope,

- the autocorrelation coefficients at lag 1, 2, and 3 of stock and index prices opposite behaviors, being the ones of the observed McDonald's prices ranked very high (82-83) with respect to their bootstrap distributions and the ones of the observed DJIA prices ranked very low (31-33). This fact could be explained observing that the original sample of stock prices seems to show less jumps that the corresponding observed time series of index prices, therefore we could expect scenarios where the bootstrapped stock prices would include more jumps than the observed ones and the index prices less jumps than their observed counterpart,

- the coefficients of the vector autoregression model of order 1 in Eq. (22) indicate that each component of the trivariate bootstrapped series is significantly explained by its value at time lag 1 (coefficients a_4, b_4, and c_4). Other significant explanatory power is given by coefficients a_2 and c_2, indicating some explanatory power of stock prices on stock volumes and index prices, respectively. Note that autoregression coefficients b_2, b_3, and c_3 are not reported since they are not significantly different from 0. Although not meant as the true model for the cross-dependencies among the three components of the trivariate series, the simple VAR(1) model shows a distribution of the coefficients such that the original sample can not be distinguished from the others.

5.1.2 Electricity case

Tables 4 and 5 report the statistics for the electricity case. In particular, we show the 5^{th} and the 95^{th} percentiles of the bootstrap distributions of some statistics calculated on the 5000 series generated from heuristic solution λ_B^H. We also show the percentile ranks of the original sample.

Table 4: Percentiles and percentile ranks of the observed daily Spanish electricity prices with respect to the bootstrap distributions of some statistics calculated on 5000 scenarios.

Statistics	5^{th} percentile	95^{th} percentile	Value of original sample	Percentile rank of original sample value
Average	36.451	45.882	42.162	66
Standard dev.	12.926	18.859	15.656	48
Skewness	0.285	1.206	0.645	42
Kurtosis	−0.417	2.375	0.060	29
Minimum	2.144	4.848	3.127	84
Maximum	86.104	143.489	103.757	44
Autocorr. at lag 1	0.916	0.957	0.951	83
Autocorr. at lag 2	0.885	0.939	0.925	67
Autocorr. at lag 3	0.855	0.922	0.909	76
Lin. regr. slope	0.001	0.010	0.005	41
Autoregr. coeff. d_1	−5.829	2.808	0.882	76
Autoregr. coeff. d_3	0.903	0.948	0.948	94

Table 5: Percentiles and percentile ranks of the observed daily Spanish electricity volumes with respect to the bootstrap distributions of some statistics calculated on 5000 scenarios.

Statistics	5^{th} percentile	95^{th} percentile	Value of original sample	Percentile rank of original sample value
Average	499,724.466	583,968.549	544,522.651	22
Standard dev.	55,015.327	141,208.856	110,929.655	80
Skewness	−1.460	0.468	−1.171	21
Kurtosis	−0.768	3.626	1.413	72
Minimum	170,699.541	376,542.745	170,199.000	4
Maximum	748,242.305	802,544.245	804,089.000	96
Autocorr. at lag 1	0.887	0.979	0.971	85
Autocorr. at lag 2	0.799	0.963	0.946	84
Autocorr. at lag 3	0.735	0.951	0.932	85
Lin. regr. slope	−37.172	39.750	14.233	66
Autoregr. coeff. f_1	8,529.110	69,177.578	12,599.858	14
Autoregr. coeff. f_2	34.233	354.585	112.301	24
Autoregr. coeff. f_3	0.857	0.979	0.968	83

From Tables 4 and 5, we can make the following remarks:

- all the statistics computed for the observed time series take values between the 5^{th} and the 95^{th} percentiles, except for the minimum and maximum of the electricity volumes ranked in the 4^{th} and 96^{th} percentiles of the corresponding distributions,

- as in the stock-index case, the previous observation applies in particular to the autocorrelation coefficients at lag 3, which capture an auto-dependence beyond the order 2 of the Markov chain,

- as in the stock-index case, the coefficients of the vector autoregression model of order 1 in Eq. (23) indicate that each component of the bivariate bootstrapped series is significantly explained by its value at time lag 1 (coefficients d_3 and f_3). Other significant explanatory power is given by coefficient f_2, indicating some explanatory power of electricity prices on electricity volumes. Note that autoregression coefficient d_2 is not reported since it is not significantly different from 0. Although not meant as the true model for the cross-dependencies among the two components of the bivariate series, the simple VAR(1) model shows a distribution of the coefficients such that the observed time series can not be distinguished from the others.

5.1.3 Rare events

To show that the presence of rare events is correctly handled by our Markov chain bootstrapping, we want to show how the percentile rank of the original sample changes and become more centered when calculated on a specific subset of the 5000 bootstrapped series. In particular, given the fact that the observed electricity prices show a significant positive spike at the beginning of 2002 (see Panel (a) of Figure 1), we would like to separate the scenarios between those which reproduce such spike and those not showing it. The filter is represented by the maximum electricity price of the bootstrapped series. To this end, we choose to set a threshold of 70 euros per MWh: the scenarios where the electricity price has crossed such value are considered "spiked"[6]. This set contains 4751 elements, while the remaining 249 "non-spiked" scenarios fall into the other subset.

From Table 6, we can see that the overall effect of this separation is that the observed time series now "falls" much better in the middle of the 5-95 percentile range. The absence of a spike in a subsample of the entire bootstrap set is however welcome. Indeed, since spikes are a rare event, we should not expect that they appear "regularly" in all the bootstrapped series.

[6] The maximum value of the bootstrapped series to be matched with the threshold of 70 euros per MWh is calculated on the prices before adding trend and seasonality.

Table 6: Percentile ranks of the observed daily Spanish electricity prices with respect to the bootstrap distributions of some statistics calculated on 4751 spiked scenarios and 249 non-spiked scenarios.

Statistics	Value of original sample	Percentile rank of original sample value with respect to **spiked** scenarios	Percentile rank of original sample value with respect to **non-spiked** scenarios
Average	42.162	65	95
Standard dev.	15.656	46	94
Skewness	0.645	41	63
Kurtosis	0.060	28	45
Minimum	3.127	84	87
Maximum	103.757	41	99
Autocorr. at lag 1	0.951	82	96
Autocorr. at lag 2	0.925	67	87
Autocorr. at lag 3	0.909	75	92
Lin. regr. slope	0.005	40	45
Autoregr. coeff. d_1	0.882	77	56
Autoregr. coeff. d_3	0.948	94	98

The statistical soundness of the previous results, in terms of the percentile ranks of the statistics, is an important validation of the proposed bootstrap method. In particular, it is significant in the case of the autocorrelations and the autoregression coefficients, which capture the auto- and the cross-dependence among the data. Given these results, it can be said that the observed time series (both the trivariate series and the bivariate ones) can be considered as trajectories sampled from the same processes generating the bootstrapped series.

5.2 A comparison with other bootstrap methods

Given that the proposed method is in the class of nonparametric approaches, to highlight its performance we develop here a comparison with other two well established bootstrap methods of the same kind, namely the Variable Length Markov Chain (VLMC) bootstrap (see Bühlmann and Wyner, 1999 and Mächler and Bühlmann, 2004) and the Multivariate Block (MB) bootstrap (see Künsch, 1989). The observed time series of prices and volumes of the Spanish electricity market have been used again in this comparison, thanks to their challenging features. Each method has been applied to generate 5000 bootstrapped series.

The VLMC bootstrap belongs to the class of Markov chain methods. However, to the best of our knowledge, this method addresses only the case of univariate processes. For this reason we bounded our method to the univariate case of the electricity prices. We also bounded our method up to order $k = 7$, whereas the VLMC bootstrap self-calibrates the order of the Markov chain.

The MB bootstrap is not a Markov chain method, but it is multivariate, so we have been able to fully compare it with our method. We therefore observed the bivariate series of electricity prices and volumes. To facilitate the comparison between the two methods, we set the length of blocks equal to 2, i.e. the order of our Markov chain in the bivariate case.

We analysed the same statistics considered in the previous subsections (without distinguishing for rare events). Tables 7, 8, and 9 compare the 5^{th} and the 95^{th} percentiles as well as the percentile ranks that the original sample takes with respect to the bootstrap distributions generated by the methods compared. The packages written in R named "VLMC" and "boot" (available at the web page http://cran.r-project.org/) were used to generate the bootstrapped series for the VLMC and the MB methods, respectively. In particular, the MB bootstrap was carried out using the function "tsboot" available in the package "boot" (other details in Davison and Hinkley, 1997).

Overall, our method outperforms the MB bootstrap (see Tables 7 and 8). The MB bootstrap actually shows better performances with respect to the Average and Standard deviation of both prices and volumes. However, it clearly performs poorly with respect to almost all the other statistics,

in particular with respect to those focusing on the auto- and the cross-dependence among the data. The autocorrelations at all lags are severely underestimated compared to the corresponding value of the original sample, with percentile ranks of 99%. The autoregression coefficients are also extremely underestimated, with percentile ranks either of 0% or of 99%.

Table 7: Percentiles and percentile ranks of the bootstrap distributions of some statistics calculated on 5000 scenarios generated with the Multivariate Block bootstrap (MB boot.) and the bootstrap method proposed in this paper (Spanish electricity prices).

		MB boot.			Our bootstrap		
Statistics	Value[a]	5^{th} pctl[b]	95^{th} pctl[b]	Pctl rank[c]	5^{th} pctl[b]	95^{th} pctl[b]	Pctl rank[c]
Average	42.162	41.453	42.862	51	36.451	45.882	66
Standard dev.	15.656	15.142	16.372	41	12.926	18.859	48
Skewness	0.645	0.615	0.950	9	0.285	1.206	42
Kurtosis	0.060	0.244	1.833	0	-0.417	2.375	29
Minimum	3.127	2.183	4.770	81	2.144	4.848	84
Maximum	103.757	103.169	146.083	6	86.104	143.489	44
Aut. at lag 1	0.951	0.493	0.546	99	0.916	0.957	83
Aut. at lag 2	0.925	0.042	0.143	99	0.885	0.939	67
Aut. at lag 3	0.909	0.052	0.132	99	0.855	0.922	76
Lin. regr. slope	0.005	0.004	0.006	27	0.001	0.010	41
Autoregr. coeff. d_1	0.882	7.367	13.067	0	-5.829	2.808	76
Autoregr. coeff. d_3	0.948	0.479	0.533	99	0.903	0.948	94

[a]: value is the actual value of the statistic observed in the original sample
[b]: pctl stands for percentile
[c]: pctl rank stands for the percentile rank of the original sample value

Table 8: Percentiles and percentile ranks of the bootstrap distributions of some statistics calculated on 5000 scenarios generated with the Multivariate Block bootstrap (MB boot.) and the bootstrap method proposed in this paper (Spanish electricity volumes).

		MB boot.			Our bootstrap		
Statistics	Value[a]	5^{th} pctl[b]	95^{th} pctl[b]	Pctl rank[c]	5^{th} pctl[b]	95^{th} pctl[b]	Pctl rank[c]
Average	544,522.651	539,226.587	549,531.631	51	499,724.466	583,968.549	22
Standard dev.	110,929.655	104,926.277	114,694.541	64	55,015.327	141,208.856	80
Skewness	-1.171	-1.350	-1.177	96	-1.460	0.468	21
Kurtosis	1.413	1.172	2.018	25	-0.768	3.626	72
Minimum	170,199.000	166,888.158	195,722.908	20	170,699.541	376,542.745	4
Maximum	804,089.000	767,548.231	806,977.318	90	748,242.305	802,544.245	96
Aut. at lag 1	0.971	0.464	0.513	99	0.887	0.979	85
Aut. at lag 2	0.946	-0.041	0.053	99	0.799	0.963	84
Aut. at lag 3	0.932	-0.029	0.040	99	0.735	0.951	85
Lin. regr. slope	14.233	4.197	14.760	92	-37.172	39.750	66
Autoregr. coeff. f_1	12,599.858	231,501.028	259,480.648	0	8,529.110	69,177.578	14
Autoregr. coeff. f_2	112.301	741.012	1,183.616	0	34.233	354.585	24
Autoregr. coeff. f_3	0.968	0.450	0.499	99	0.857	0.979	83

[a]: value is the actual value of the statistic observed in the original sample
[b]: pctl stands for percentile
[c]: pctl rank stands for the percentile rank of the original sample value

Our method also outperforms overall the VLMC bootstrap (see Table 9). What the VLMC does nicely is that it generates narrower ranges between the 5^{th} and the 95^{th} percentiles than our method. Unfortunately, they are only seldom consistent. In particular, the autocorrelations at all lags are severely underestimated in the VLMC bootstrapped series. This outcome is similar to the case of the MB bootstrap, but, given the Markov chain nature of the VLMC bootstrap, it is more surprising.

Table 9: Percentiles and percentile ranks of the bootstrap distributions of some statistics calculated on 5000 scenarios generated with the VLMC bootstrap and the bootstrap method proposed in this paper (Spanish electricity prices).

		VLMC bootstrap			Our bootstrap		
Statistics	Value[a]	5^{Th} pctl[b]	95^{Th} pctl[b]	Pctl rank[c]	5^{Th} pctl[b]	95^{Th} pctl[b]	Pctl rank[c]
Average	29.692	28.707	30.704	53	26.574	32.074	57
Standard dev.	9.570	8.157	10.581	72	7.113	10.979	68
Skewness	1.381	0.414	1.953	68	0.114	2.033	66
Kurtosis	5.081	0.293	9.352	59	-0.571	9.327	63
Minimum	5.469	5.726	9.311	0	4.546	11.754	58
Maximum	103.757	66.604	110.968	73	50.971	111.382	71
Aut. at lag 1	0.818	0.737	0.817	95	0.737	0.859	62
Aut. at lag 2	0.706	0.579	0.702	95	0.579	0.772	61
Aut. at lag 3	0.706	0.463	0.615	99	0.547	0.745	63
Aut. at lag 4	0.667	0.371	0.540	99	0.529	0.733	63
Aut. at lag 5	0.661	0.297	0.476	99	0.52	0.73	62
Aut. at lag 6	0.721	0.236	0.423	99	0.614	0.764	65
Aut. at lag 7	0.802	0.187	0.378	99	0.728	0.829	68
Aut. at lag 8	0.683	0.148	0.338	99	0.581	0.727	64
Lin. regr. slope	0.004	0.002	0.005	78	-0.001	0.007	63

[a]: value is the actual value of the statistic observed in the original sample
[b]: pctl stands for percentile
[c]: pctl rank stands for the percentile rank of the original sample value

The general conclusion of the present comparison is that the superior performance of the method proposed here depends crucially, from a mathematical point of view, on a transition probability matrix keeping much of the information contained in the observed time series. This indeed explains why each value of the bootstrapped series keeps, much more precisely than the methods compared here, the original dependence from its previous values, as well as from those of the other components (bivariate case).

5.3 Performance evaluation of the Two-Phase Tabu Search

The goal of this subsection is twofold. Firstly, we provide some evidence on the effectiveness of the Two-Phase Tabu Search described in Section 3 comparing its performance with that of the Tabu Search algorithm introduced in Cerqueti et al. (2013) for univariate instances without the contiguity constraint. Secondly, we give some insights on the computational results for the multivariate instances.

The Two-Phase Tabu Search has been coded in Java. The computational experiments have been conducted on a PC Intel Xeon with a 3.33 GHz processor, and 12.00 GB of RAM. After extensive preliminary experiments, the parameters of the heuristic have been set to the values detailed in Online Resource 1 - Appendix D.

It is worth to highlight that in Cerqueti et al. (2013) the approximation of the efficient frontier is constructed with a slightly different approach with respect to the one described in Subsection 3.1 (see Subsection 3.1 of Cerqueti et al., 2013, for further details). Hence, in order to make a fair comparison, we adapt the Two-Phase Tabu Search and compare the approximated efficient frontiers. The comparison is based on the Spanish and German electricity prices analyzed in Cerqueti et al. (2013). The experiments are performed on the hardware used in Cerqueti et al. (2013) to allow a direct comparison of computing times. As far as the quality of the approximations of the efficient frontier is considered, we show in Figure 3 three approximations of the efficient frontiers for the Spanish and German instances, respectively. The three approximations correspond to the heuristic designed in Cerqueti et al. (2013) (denoted as Tabu Search), algorithm TSP1, and procedure TSP1+TSP2 with the parameter settings described in Online Resource 1 - Appendix D. The coordinates $(m_{\lambda^H}, d_{\lambda^H})$ of the points composing the approximations of the efficient frontiers for the Spanish and German instances are reported in Tables 12 and 13 in Online Resource 1 - Appendix E, respectively. In those

tables, for algorithms TSP1 and TSP1+TSP2 we also report the computing time (in seconds) needed to solve each problem, while for the Tabu Search we report only the total and average computing times retrieved from Cerqueti et al. (2013).

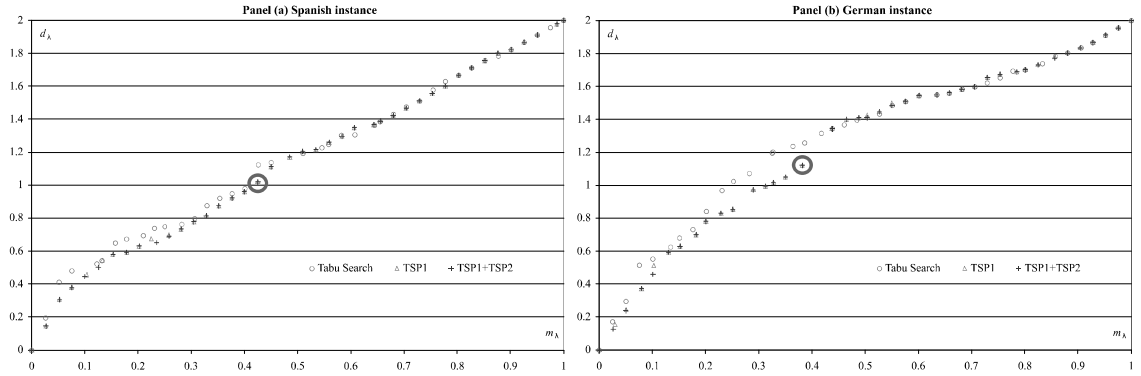

Fig. 3: Comparison of three approximations of the efficient frontiers for the Spanish and the German instances. The points in the red circles show possible preferred choices, according to the rule described at the beginning of Subsection 5.1.

As it can be noticed from Figure 3, the approximations provided by both TSP1 and TSP1+TSP2 outperform the approximation found with the Tabu Search. This is evident for most of the points lying in the left half of the chart. Indeed, for most of these points, given a value of multiplicity measure m_λ, both TSP1 and TSP1+TSP2 provide a solution with a smaller value of distance indicator d_λ. Regarding the right half of the chart, it seems that the three heuristics found roughly the same set of points. As far as computing times are taken into consideration, both TSP1 and TSP1+TSP2 are faster than Tabu Search as it can be seen in Table 10.

Table 10: The Spanish and German instances introduced in Cerqueti et al. (2013): A summary of computing times (in seconds) for different approximations of the efficient frontiers.

	Tabu Search		TSP1		TSP1+TSP2	
Instance	Total	Average	Total	Average	Total	Average
Spanish	211.550	5.424	84.365	2.163	156.359	4.009
German	619.669	15.889	110.915	2.844	202.114	5.182

Regarding the impact of the introduction of the contiguity constraint, we can draw some conclusions. On the one hand, it seems that imposing the contiguity constraint has negligible effects on the solution quality. Indeed, in only few cases algorithm TSP1+TSP2 has improved upon the solution found by algorithm TSP1 (note that, as we are dealing with heuristic algorithms, this outcome might also be related to a not effective performance of procedure TSP2). On the other hand, the introduction of the contiguity constraint restricts significantly the solution space making its exploration much quicker and more effective.

As far as computing the approximations of the efficient frontiers for the electricity and the stock-index cases is considered, algorithm TSP1 took approximately 5066 seconds to compute the approximation for the former case, whereas it required less than 4160 seconds for the latter case. We remark that no dominated point was found for the two instances. The details of the heuristic solutions λ_B^H and λ_T^H, i.e. the kN-partitions used as inputs for the bootstrap method in the electricity and the

stock-index cases, are reported in Tables 11 and 12, respectively. Finally, Table 14 in Online Resource 1 - Appendix E details the coordinates of all the points found by algorithm TSP1 (the two points corresponding to λ_B^H and λ_T^H are highlighted in bold).

Table 11: Heuristic solution λ_B^H of the electricity case. To simplify readability, we refer to initial state $a_i^{(n)}$ as i.

$\lambda_B^H = (\lambda_{B,2}^H, \lambda_{B,1}^H)$	
Time lag 2 partition:	$\lambda_{B,2}^H = (\lambda_{B,2}^{H,(1)}, \lambda_{B,2}^{H,(2)})$
prices (ep):	$\lambda_{B,2}^{H,(1)} = \{\{1,2,3,4,5,6,7\},\{8\},\{9\},\{10\},\{11\},\{12,13\}\}$
volumes (ev):	$\lambda_{B,2}^{H,(2)} = \{\{1,2,3,4,5,6,7,8,9,10,11,12,13\}\}$
Time lag 1 partition:	$\lambda_{B,1}^H = (\lambda_{B,1}^{H,(1)}, \lambda_{B,1}^{H,(2)})$
prices (ep):	$\lambda_{B,1}^{H,(1)} = \{\{1,2,3,4,5,6\},\{7\},\{8\},\{9\},\{10\},\{11\},\{12,13\}\}$
volumes (ev):	$\lambda_{B,1}^{H,(2)} = \{\{1\},\{2\},\{3\},\{4\},\{5,6,7,8,9,10\},\{11\},\{12\},\{13\}\}$

Table 12: Heuristic solution λ_T^H of the stock-index case. To simplify readability, we refer to initial state $a_i^{(n)}$ as i.

$\lambda_T^H = (\lambda_{T,2}^H, \lambda_{T,1}^H)$	
Time lag 2 partition:	$\lambda_{T,2}^H = (\lambda_{T,2}^{H,(1)}, \lambda_{T,2}^{H,(2)}, \lambda_{T,2}^{H,(3)})$
stock volumes (sv):	$\lambda_{T,2}^{H,(1)} = \{\{1,2,3,4,5,6,7,8\}\}$
stock returns (sr):	$\lambda_{T,2}^{H,(2)} = \{\{1\},\{2,3,4\},\{5\},\{6\}\}$
index returns (ir):	$\lambda_{T,2}^{H,(3)} = \{\{1\},\{2,3\},\{4\},\{5\},\{6\}\}$
Time lag 1 partition:	$\lambda_{T,1}^H = (\lambda_{T,1}^{H,(1)}, \lambda_{T,1}^{H,(2)}, \lambda_{T,1}^{H,(3)})$
stock volumes (sv):	$\lambda_{T,1}^{H,(1)} = \{\{1,2,3\},\{4\},\{5\},\{6\},\{7\},\{8\}\}$
stock returns (sr):	$\lambda_{T,1}^{H,(2)} = \{\{1\},\{2\},\{3,4,5\},\{6\}\}$
index returns (ir):	$\lambda_{T,1}^{H,(3)} = \{\{1\},\{2\},\{3,4,5\},\{6\}\}$

6 Conclusions

This paper contains a new methodological procedure for the application of Markov chain bootstrapping to treat the case of N-dimensional discrete time continuous-valued stochastic processes, which are of remarkable importance in many economic and financial applications. Indeed, the world of economics and finance is populated by strongly interconnected phenomena. Besides, in many cases the dependence among variables is often difficult to know or to model satisfactorily. In all such cases bootstrapping can show severe drawbacks if based on analytic or econometric models. On the contrary, the multivariate Markov chain bootstrapping proposed here can capture conditional dependencies of arbitrary complexity among variables.

We proceed by formalizing a constrained optimization problem grounded on the basic requirements of the bootstrap theory: the synthetic series should maintain the statistical features of the original sample and, at the same time, have very low probability to replicate it.

The optimization problem has also been characterized by the introduction of a contiguity constraint. This constraint has proved useful in two ways: it reduces the cardinality of the solution space of the optimization problem and it offers a more intuitive interpretation of the resulting partitions of the state space of the Markov chain. The method has been applied to two empirical cases: the bivariate process of prices and volumes of electricity in the Spanish market; the trivariate process composed by prices and volumes of the US company stock McDonald's and prices of the Dow Jones Industrial Average index. Both cases present major difficulties for bootstrap methods based on ana-

lytic or econometric models. In the case of the electricity markets, even the univariate bootstrapping of prices is still today an open challenge.

The method has proved to guarantee good statistical properties of the bootstrapped series. Moreover, the method has also been compared with two other well established bootstrap approaches, namely the Multivariate Block bootstrap and the (univariate) Variable Length Markov Chain bootstrap. Our method outperforms both the approaches, especially in terms of auto- and cross-dependence among the data. Finally, the computational results show that the proposed solution strategy improves the performance of an alternative heuristic available in the literature (designed for the univariate case), in terms of both solution quality and computing times.

References

Akaike, H.: 1970, On a decision procedure for system identification, *in* Kyoto Symposium on System Engineering Approach to Computer Control (ed.), *Proceedings of the IFAC Kyoto Symposium on System Engineering Approach to Computer Control*, Kyoto Symposium on System Engineering Approach to Computer Control, Kyoto, Japan, pp. 485–490.

Anatolyev, S. and Vasnev, A.: 2002, Markov chain approximation in bootstrapping autoregressions, *Economics Bulletin* **3**(19), 1–8.

Athreya, K. B. and Fuh, C. D.: 1992, Bootstrapping Markov chains: Countable case, *Journal of Statistical Planning and Inference* **33**(3), 311–331.

Brock, W., Lakonishok, J. and LeBaron, B.: 1992, Simple technical trading rules and the stochastic properties of stock returns, *The Journal of Finance* **47**(5), 1731–1764.

Bühlmann, P.: 1998, Extreme events from the return-volume process: A discretization approach for complexity reduction, *Applied Financial Economics* **8**(3), 267–278.

Bühlmann, P.: 2002, Bootstraps for time series, *Statistical Science* **17**(1), 52–72.

Bühlmann, P. and Wyner, A. J.: 1999, Variable length Markov chains, *The Annals of Statistics* **27**(2), 480–513.

Bunn, D. W. (ed.): 2004, *Modelling Prices in Competitive Electricity Markets*, John Wiley & Sons, Chichester, UK.

Burke, C. J. and Rosenblatt, M.: 1958, A Markovian function of a Markov chain, *The Annals of Mathematical Statistics* **29**(4), 1112–1122.

Cerqueti, R., Falbo, P., Guastaroba, G. and Pelizzari, C.: 2013, A tabu search heuristic procedure in Markov chain bootstrapping, *European Journal of Operational Research* **227**(2), 367–384.

Cerqueti, R., Falbo, P. and Pelizzari, C.: 2010, Relevant states and memory in Markov chain bootstrapping and simulation, *Munich Personal RePEc Archive Paper 46254*, University Library of Munich, Germany. http://mpra.ub.uni-muenchen.de/46254/1/MPRA_paper_46250.pdf.

Cerqueti, R., Falbo, P., Pelizzari, C., Ricca, F. and Scozzari, A.: 2012, A mixed integer linear programming approach to Markov chain bootstrapping, *Quaderni del Dipartimento di Economia e Diritto dell'Università degli Studi di Macerata 67*, Università degli Studi di Macerata, Macerata, Italy. http://www2.unimc.it/ricerca/dipartimenti/dipartimento-di-istituzioni-economiche-e/wpaper/wpaper00067/filePaper.

Chankong, V. and Haimes, Y. Y.: 1983, *Multiobjective Decision Making: Theory and Methodology*, North-Holland, New York, NY, USA.

Ching, W.-K., Ng, M. K. and Fung, E. S.: 2008, Higher-order multivariate Markov chains and their applications, *Linear Algebra and Its Applications* **428**(2-3), 492–507.

Csiszár, I. and Shields, P. C.: 2000, The consistency of the BIC Markov order estimator, *The Annals of Statistics* **28**(6), 1601–1619.

Datta, S. and McCormick, W. P.: 1992, Bootstrap for a finite state Markov chain based on i.i.d. resampling, *in* R. LePage and L. Billard (eds), *Exploring the Limits of Bootstrap*, John Wiley & Sons, New York, NY, USA, pp. 77–97.

Davison, A. C. and Hinkley, D. V.: 1997, *Bootstrap Methods and Their Application*, Cambridge University Press, Cambridge, UK.

Efron, B.: 1979, Bootstrap methods: Another look at the jackknife, *The Annals of Statistics* **7**(1), 1–26.

Franke, J.: 2012, Markov switching time series models, *in* T. S. Rao, S. S. Rao and C. R. Rao (eds), *Time Series Analysis: Methods and Applications*, Elsevier, Oxford, UK, pp. 99–122.

Glover, F. and Laguna, M.: 1997, *Tabu Search*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Guastaroba, G., Mansini, R. and Speranza, M. G.: 2009, On the effectiveness of scenario generation techniques in single-period portfolio optimization, *European Journal of Operational Research* **192**(2), 500–511.

Horowitz, J. L.: 2003, Bootstrap methods for Markov processes, *Econometrica* **71**(4), 1049–1082.

Huisman, R. and Mahieu, R.: 2003, Regime jumps in electricity prices, *Energy Economics* **25**(5), 425–434.

Kemeny, J. G. and Snell, J. L.: 1976, *Finite Markov Chains*, Springer, Berlin, Germany.

Kieffer, J. C.: 1993, Strongly consistent code-based identification and order estimation for constrained finite-state model classes, *IEEE Transactions on Information Theory* **39**(3), 893–902.

Kolmogorov, A. N.: 1965, Three approaches to the quantitative definition of information, *Problemy Peredachi Informatsii* **1**(1), 3–11.

Kulperger, R. J. and Prakasa Rao, B. L. S.: 1989, Bootstrapping a finite state Markov chain, *Sankhya, The Indian Journal of Statistics* **51**(2), 178–191.

Künsch, H. R.: 1989, The jackknife and the bootstrap for general stationary observations, *The Annals of Statistics* **17**(3), 1217–1241.

Mächler, M. and Bühlmann, P.: 2004, Variable length Markov chains: Methodology, computing, and software, *Journal of Computational and Graphical Statistics* **13**(2), 435–455.

Merhav, N., Gutman, M. and Ziv, J.: 1989, On the estimation of the order of a Markov chain and universal data compression, *IEEE Transactions on Information Theory* **35**(5), 1014–1019.

Miettinen, K. M.: 1999, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, MA, USA.

Morvai, G. and Weiss, B.: 2005, Order estimation of Markov chains, *IEEE Transactions on Information Theory* **51**(4), 1496–1497.

Paparoditis, E. and Politis, D. N.: 2001b, A Markovian local resampling scheme for nonparametric estimators in time series analysis, *Econometric Theory* **17**(3), 540–566.

Rajarshi, M. B.: 1990, Bootstrap in Markov-sequences based on estimates of transition density, *Annals of the Institute of Statistical Mathematics* **42**(2), 253–268.

Rissanen, J.: 1978, Modeling by shortest data description, *Automatica* **14**(5), 465–471.

Schwarz, G.: 1978, Estimating the dimension of a model, *The Annals of Statistics* **6**(2), 461–464.

Sullivan, R., Timmermann, A. and White, H.: 1999, Data-snooping, technical trading rule performance, and the bootstrap, *The Journal of Finance* **54**(5), 1647–1691.

Thomas, M. U.: 2010, Aggregation and lumping of DTMCs, *in* J. J. Cochran, L. A. Cox Jr., P. Keskinocak, J. P. Kharoufeh and J. C. Smith (eds), *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley & Sons, Hoboken, NJ, USA.

Ward Jr., J. H.: 1963, Hierarchical grouping to optimize an objective function, *Journal of the American Statistical Association* **58**(301), 236–244.

Weron, R.: 2006, *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*, John Wiley & Sons, Chichester, UK.

Weron, R., Bierbrauer, M. and Trück, S.: 2004, Modeling electricity prices: Jump diffusion and regime switching, *Physica A: Statistical Mechanics and Its Applications* **336**(12), 39–48.

Approximating Multivariate Markov Chains for Bootstrapping through Contiguous Partitions[1]

Online Resource 1 - Appendices A, B, C, D, E, and F

Roy Cerqueti
Università degli Studi di Macerata - Dipartimento di Economia e Diritto
Via Crescimbeni, 20 - 62100 Macerata MC - Italy
E-mail: roy.cerqueti@unimc.it

Paolo Falbo*, Gianfranco Guastaroba, Cristian Pelizzari
Università degli Studi di Brescia - Dipartimento di Economia e Management
Contrada Santa Chiara, 50 - 25122 Brescia BS - Italy
E-mail: {paolo.falbo, gianfranco.guastaroba, cristian.pelizzari}@unibs.it

*Corresponding author. Tel.: +39 030 2988531. Fax: +39 030 2400925. E-mail: paolo.falbo@unibs.it

[1] The work has been supported under grant by "Regione Lombardia: Metodi di integrazione delle fonti energetiche rinnovabili e monitoraggio satellitare dell'impatto ambientale", EN-17, ID 17369.10.

Appendix A - Two-Phase Tabu Search: Pseudo-Codes

Algorithm TSP1 begins computing in Steps 1 to 9 an initial feasible solution given the input represented by the infeasible solution $\lambda \in \Lambda^k$. A sequence of $(1,n)$-relocations is performed until the first feasible solution is found. This solution becomes the incumbent one.
The core of algorithm TSP1 is represented by the instructions from Step 10 to Step 33. After performing some initialization tasks, the successive instructions are performed iterP1$_{max}$ times. Firstly, in Step 12 a time lag $\bar{w}$ is randomly selected from the discrete distribution function computed as mentioned in Subsection 3.2. Then, in the subsequent for loop (Steps 13-18) each component n is firstly considered individually to construct the neighborhood $\mathcal{N}_{P1}\binom{n}{\bar{w}}$ consisting in all the UUPs $\lambda_{\bar{w}}^{(n)\prime}$ at time lag $\bar{w}$ that can be obtained performing (q,n)-relocations and (q,n)-creations (Step 14). Any other UUP $\lambda_w^{(n')}$, with $w \neq \bar{w}$ and $n' \neq n$, remains unchanged. Furthermore, among all the possible UUPs only those providing a feasible solution $\lambda' \in \Lambda^k$ are considered. Similarly, each pair of components is considered in Step 16 to construct the neighborhood $\mathcal{N}_{P1}\binom{n,\bar{n}}{\bar{w}}$ consisting in all pairs of UUPs $(\lambda_{\bar{w}}^{(n)\prime}, \lambda_{\bar{w}}^{(\bar{n})\prime})$, with $n \neq \bar{n}$, at time lag $\bar{w}$ that can be obtained performing moves 1-4. After evaluating all the possible moves, the one leading to a feasible solution $\lambda' \in \Lambda^k$ with minimum $d_{\lambda'}$ is implemented in Step 19, provided that UUP $\lambda_{\bar{w}}^{(n)\prime}$ (and also UUP $\lambda_{\bar{w}}^{(\bar{n})\prime}$ if one move among 1-4 is performed) is not tabu or that the cost of the feasible solution after implementing the move is better than the cost of the best-known feasible solution $\lambda^H \in \Lambda^k$ (the aspiration criterion). Feasible solution λ' becomes the new incumbent solution for the next iteration (Step 20), whereas the UUP $\lambda_{\bar{w}}^{(n)}$ is made tabu in Step 21 (as well as UUP $\lambda_{\bar{w}}^{(\bar{n})\prime}$ if one move among 1-4 is performed). Other update operations are performed from Step 22 to Step 25. If iterNoImprP1$_{max}$ iterations have been consecutively performed without any improvement, the value of q is firstly increased by 1 (Step 27). Then, if q takes a value larger than q_{max}, procedure JumpingP1 is called in Step 29.

Algorithm 3 Procedure: TSP1.

Input: an infeasible solution $\lambda \in \Lambda^k$ and parameter γ_i.
Output: the best-known feasible solution $\lambda^H \in \Lambda^k$ given γ_i.

/ Initial feasible solution computation */*
1: **repeat**
2: Select randomly time lag $\bar{w} \in \{1,...,k\}$.
3: Let $\lambda = (\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1) \in \Lambda^k$, and $\lambda_{\bar{w}} = (\lambda_{\bar{w}}^{(1)}, ..., \lambda_{\bar{w}}^{(n)}, ..., \lambda_{\bar{w}}^{(N)}) \in \Lambda$.
4: **for** $n = 1 \rightarrow N$ **do**
5: Evaluate all the $(1,n)$-relocations defining $\mathcal{N}_{P1}\binom{n}{\bar{w}}$.
6: **end for**
7: Choose UUP $\lambda_{\bar{w}}^{(n)\prime}$ that maximizes $m_{\lambda'}$, where λ' is obtained from λ replacing UUP $\lambda_{\bar{w}}^{(n)}$ with $\lambda_{\bar{w}}^{(n)\prime}$ and letting all the other UUPs unchanged.
8: Set $\lambda := \lambda'$.
9: **until** λ is feasible (i.e. $m_\lambda \geq \gamma_i$)
/ Explore contiguous partitions only */*
10: Set $\lambda^H := \lambda$; $TL_w^{(n)} := \varnothing$, where $w = 1, ..., k$ and $n = 1, ..., N$; $q := 1$; iter $:= 0$; and iterNoImpr $:= 0$.
11: **while** iter $<$ iterP1$_{max}$ **do**
12: Select randomly time lag $\bar{w} \in \{1,...,k\}$.
13: **for** $n = 1 \rightarrow N$ **do**
14: Evaluate all the (q,n)-relocations and (q,n)-creations defining $\mathcal{N}_{P1}\binom{n}{\bar{w}}$.
15: **for** $\bar{n} = (n+1) \rightarrow N$ **do**
16: Evaluate all moves 1-4 defining $\mathcal{N}_{P1}\binom{n,\bar{n}}{\bar{w}}$.
17: **end for**
18: **end for**
19: Choose UUP $\lambda_{\bar{w}}^{(n)\prime}$ (and, possibly, UUP $\lambda_{\bar{w}}^{(\bar{n})\prime}$) such that $\lambda_{\bar{w}}^{(n)\prime} \notin TL_{\bar{w}}^{(n)}$ ($\lambda_{\bar{w}}^{(\bar{n})\prime} \notin TL_{\bar{w}}^{(\bar{n})}$) or $d_{\lambda'} < d_{\lambda^H}$ and $d_{\lambda'}$ is minimized, where λ' is a feasible solution obtained from λ replacing UUP $\lambda_{\bar{w}}^{(n)}$ with $\lambda_{\bar{w}}^{(n)\prime}$ ($\lambda_{\bar{w}}^{(\bar{n})}$ with $\lambda_{\bar{w}}^{(\bar{n})\prime}$) and letting all the other UUPs unchanged.

Algorithm 3 Procedure: TSP1 (continued).

Set $\lambda := \lambda'$.
Update $TL_{\bar{w}}^{(n)}$ ($TL_{\bar{w}}^{(\bar{n})}$).
if $d_\lambda < d_{\lambda^H}$ **then**
  Set $\lambda^H := \lambda$, $q := 1$, and iterNoImpr $:= 0$.
else
  Set iterNoImpr := iterNoImpr + 1.
  if iterNoImpr > iterNoImprP1$_{max}$ **then**
    Set $q := q + 1$ and iterNoImpr $:= 0$.
    if $q > q_{max}$ **then**
      Call procedure JumpingP1.
    end if
  end if
end if
end while
/ The best-known feasible solution λ^H at the end of TSP1 is provided as input of procedure TSP1 performed to solve the next ϵ-constraint problem. */*

As far as algorithm TSP2 is considered, we here highlight only the differences with respect to algorithm TSP1 described above. The neighborhood $\mathcal{N}_{P2}\binom{n}{\bar{w}}$ is explored in Step 5 and procedure JumpingP2 is called in Step 17. Procedure JumpingP2 performs the steps sketched in Algorithm 5 with the only exception of Step 4, where the neighborhood $\mathcal{N}_{P2}\binom{n}{\bar{w}}$ is explored instead of the neighborhood $\mathcal{N}_{P1}\binom{n}{\bar{w}}$. Note that one can trivially set iterP2$_{max} := 0$ if the interest is in exploring the solution space whose elements fulfill the contiguity constraint.

Algorithm 4 Procedure: TSP2.

Input: the best-known feasible solution $\lambda^H \in \Lambda^k$, provided by procedure TSP1, and parameter γ_i.
Output: the best-known feasible solution λ^H, representing a (possibly not contiguous) partition of $\mathbb{A}^k$, given γ_i.

/ Explore all partitions of $\mathbb{A}$, contiguous and not contiguous */*
1: Set $\lambda := \lambda^H$; $TL_w := \varnothing$, $w = 1, ..., k$; iter $:= 0$ and iterNoImpr $:= 0$.
2: **while** iter < iterP2$_{max}$ **do**
3: Select randomly time lag $\bar{w} \in \{1, ..., k\}$.
4: **for** $n = 1 \rightarrow N$ **do**
5: Evaluate all the $(q, n)_{P2}$-relocations, $(q, n)_{P2}$-creations, and (q, n)-swaps defining $\mathcal{N}_{P2}\binom{n}{\bar{w}}$.
6: **end for**
7: Choose a partition $\lambda_{\bar{w}}^{(n)\prime}$ of $\mathbb{A}_n$ such that $\lambda_{\bar{w}}^{(n)\prime} \notin TL_{\bar{w}}^{(n)}$ or $d_{\lambda'} < d_{\lambda^H}$ and $d_{\lambda'}$ is minimized, where λ' is a feasible solution representing a partition of $\mathbb{A}^k$ obtained from λ replacing partition $\lambda_{\bar{w}}^{(n)}$ of $\mathbb{A}_n$ with $\lambda_{\bar{w}}^{(n)\prime}$ and letting all the other partitions unchanged.
8: Set $\lambda := \lambda'$.
9: Update $TL_{\bar{w}}^{(n)}$.
10: **if** $d_\lambda < d_{\lambda^H}$ **then**
11: Set $\lambda^H := \lambda$, $q := 1$, and iterNoImpr $:= 0$.
12: **else**
13: Set iterNoImpr := iterNoImpr + 1.
14: **if** iterNoImpr > iterNoImprP2$_{max}$ **then**
15: Set $q := q + 1$ and iterNoImpr $:= 0$.
16: **if** $q > q_{max}$ **then**
17: Call procedure JumpingP2.
18: **end if**
19: **end if**
20: **end if**
21: **end while**

Algorithm 5 Procedure: JumpingP1 (JumpingP2).

Input: the incumbent feasible solution $\lambda \in \Lambda^k$ and parameter δ.
Output: the new incumbent feasible solution $\lambda' \in \Lambda^k$.

1: Set $TL_w^{(n)} := \varnothing$, $w = 1, ..., k$, $n = 1, ..., N$.
2: **for** $w = 1 \rightarrow k$ **do**
3: **for** $n = 1 \rightarrow N$ **do**
4: Evaluate all the $(1, n)$-relocations and $(1, n)$-creations defining $\mathcal{N}_{P1}\binom{n}{w}$.
5: Choose UUP $\lambda_w^{(n)\prime} \in \mathcal{N}_{P1}\binom{n}{w}$ leading to the best feasible solution λ' obtained from λ replacing UUP $\lambda_w^{(n)}$ with $\lambda_w^{(n)\prime}$ and letting all the other UUPs unchanged.
6: **end for**
7: **end for**
8: Sort the selected feasible solutions in nondecreasing order of the corresponding objective function values and create list L with the respective moves.
9: Perform sequentially the first δ moves in list L. Each move is performed provided that it leads to a feasible solution. Let λ' be the resulting feasible solution.
10: **if** $d_{\lambda'} < d_{\lambda^{II}}$ **then**
11: Set $\lambda^{II} := \lambda'$.
12: **end if**
/ Procedure JumpingP2 performs the same steps with the exception of Step 4 that is replaced by: Evaluate all the $(1, n)_{P2}$-relocations, $(1, n)_{P2}$-creations, and $(1, n)$-swaps defining $\mathcal{N}_{P2}\binom{n}{w}$. Besides, the contiguity constraint need not be fulfilled in any step of the procedure. */*

Appendix B - Trend and Weekly Seasonality Removal

The estimation of the exponential trend and weekly seasonality is based on the following model:

$$e_t^{(c)} = C \exp(rt + \eta_1 \mathbb{I}_1(t) + \eta_2 \mathbb{I}_2(t) + \eta_3 \mathbb{I}_3(t) + \eta_4 \mathbb{I}_4(t) + \eta_5 \mathbb{I}_5(t) + \varepsilon_t), \tag{24}$$

where $e_t^{(c)}$ are the raw original prices (of stock, index, and electricity) and volumes (of stock and electricity), $\mathbb{I}_j(t)$ is the dummy variable associated to the j-th trading day of the week, with $j = 1, ..., 5$, r is the growth rate, η_j is the coefficient of dummy variable $\mathbb{I}_j(t)$, with $j = 1, ..., 5$, $C > 0$, and ε_t are the errors. Taking the natural logarithm of both sides of Eq. (24), we obtain:

$$z_t = u + rt + \eta_1 \mathbb{I}_1(t) + \eta_2 \mathbb{I}_2(t) + \eta_3 \mathbb{I}_3(t) + \eta_4 \mathbb{I}_4(t) + \eta_5 \mathbb{I}_5(t) + \varepsilon_t,$$

where $z_t = \ln e_t^{(c)}$ and $u = \ln C$.

For estimation purposes, we assume that the usual hypotheses of linear regression on the errors ε_t hold. We obtain the OLS estimates of r and η_j, $j = 1, ..., 5$, and they are significant at a level of 5% (see Table 10).

Table 10: Coefficients estimates of an exponential regression model of trend and weekly seasonality applied to the series of prices and volumes of the electricity case (Spain) and of the stock-index case (McDonald's and DJIA).

Coefficient estimate	Spain		McDonald's		DJIA
	Prices	Volumes	Prices	Volumes	Prices
$\hat{u}$	3.460815931	13.14958271	2.996443393	15.556310966	9.2614679294
$\hat{r}$	0.000161813	0.0000244772	0.0007206918	0.000111199	0.000050167
$\hat{\eta}_1$	0.014422005	−0.019727962	0	0	0
$\hat{\eta}_2$	0.008914955	0.005566374	0	0	0
$\hat{\eta}_3$	0.01222327	0.008742296	0	0	0
$\hat{\eta}_4$	0.019907922	0.010064997	0	0	0
$\hat{\eta}_5$	0	0	0	0	0

The coefficient estimate $\hat{\eta}_5$ of the Spanish prices and volumes is set to 0 by the estimation procedure and its value is taken by the coefficient estimate $\hat{u}$ of the intercept. This result is due to the fact that the indicator functions $\mathbb{I}_1(t)$, ..., $\mathbb{I}_5(t)$, which are worth, collectively, 1 for every t, can be viewed as a constant, therefore their contribution to explain the dependent variable could be separated into a base component, the intercept, and specific values for each of the 5 explanatory variables. The linear regression model for the stock and the index is implemented without seasonal components.

To the purpose of removing the exponential trend and the exponential weekly seasonality from our original series, we define the series of prices and volumes $e(T) = (e_1, ..., e_t, ..., e_T)$, where:

$$e_t = \exp[z_t - (\hat{r}t + \hat{\eta}_1\mathbb{I}_1(t) + \hat{\eta}_2\mathbb{I}_2(t) + \hat{\eta}_3\mathbb{I}_3(t) + \hat{\eta}_4\mathbb{I}_4(t) + \hat{\eta}_5\mathbb{I}_5(t))],\ t = 1, ..., T.$$

The original time series $e(T)$ is an input of the bootstrap method, while the output is the bootstrapped series, which we indicate by $x(\ell) = (x_1, ..., x_\kappa, ..., x_\ell)$. To re-introduce the exponential trend and the exponential weekly seasonality in $x(\ell)$, we multiply each point x_κ by

$$\exp(\hat{r}\kappa + \hat{\eta}_1\mathbb{I}_1(\kappa) + \hat{\eta}_2\mathbb{I}_2(\kappa) + \hat{\eta}_3\mathbb{I}_3(\kappa) + \hat{\eta}_4\mathbb{I}_4(\kappa) + \hat{\eta}_5\mathbb{I}_5(\kappa)),\ \kappa = 1, ..., \ell.$$

Appendix C - Initial States, or Intervals

Table 11 reports the intervals composing the initial partition of the support $\mathbb{I}_n \subseteq \mathbb{R}$ of the series of returns/prices/volumes for the bivariate and trivariate cases after removal of trend and weekly seasonality. Refer to Eqs. (3) and (4) for a definition of these intervals.

Table 11: Elements of the initial partition of the support of the series of electricity prices and volumes and of the series of stock returns, stock volumes, and index returns. In our experiment we take $\inf(a_1^{(n)}) = 0$ for prices and volumes and $\inf(a_1^{(n)}) = -1$ for returns, while $\sup(a_{L_n}^{(n)})$ is set to $\bar{\beta} = 1,000,000,000$ in both the cases.

Electricity case			
Interval label	Prices	Interval label	Volumes
$a_1^{(1)}$	$[0, 17.30)$	$a_1^{(2)}$	$[0, 237649)$
$a_2^{(1)}$	$[17.30, 21.30)$	$a_2^{(2)}$	$[237649, 286556)$
$a_3^{(1)}$	$[21.30, 24.55)$	$a_3^{(2)}$	$[286556, 336793)$
$a_4^{(1)}$	$[24.55, 27.97)$	$a_4^{(2)}$	$[336793, 396337)$
$a_5^{(1)}$	$[27.97, 30.45)$	$a_5^{(2)}$	$[396337, 446248)$
$a_6^{(1)}$	$[30.45, 34.51)$	$a_6^{(2)}$	$[446248, 487608)$
$a_7^{(1)}$	$[34.51, 39.29)$	$a_7^{(2)}$	$[487608, 520439)$
$a_8^{(1)}$	$[39.29, 43.19)$	$a_8^{(2)}$	$[520439, 541664)$
$a_9^{(1)}$	$[43.19, 47.97)$	$a_9^{(2)}$	$[541664, 572738)$
$a_{10}^{(1)}$	$[47.97, 52.25)$	$a_{10}^{(2)}$	$[572738, 599878)$
$a_{11}^{(1)}$	$[52.25, 60.60)$	$a_{11}^{(2)}$	$[599878, 636553)$
$a_{12}^{(1)}$	$[60.60, 79.46)$	$a_{12}^{(2)}$	$[636553, 674317)$
$a_{13}^{(1)}$	$[79.46, \bar{\beta}]$	$a_{13}^{(2)}$	$[674317, \bar{\beta}]$

Stock-index case					
Interval label	Stock volumes	Interval label	Stock returns	Interval label	Index returns
$a_1^{(1)}$	$[0, 4669188)$	$a_1^{(2)}$	$[-1, -0.033)$	$a_1^{(3)}$	$[-1, -0.035)$
$a_2^{(1)}$	$[4669188, 6721984)$	$a_2^{(2)}$	$[-0.033, -0.017)$	$a_2^{(3)}$	$[-0.035, -0.007)$
$a_3^{(1)}$	$[6721984, 7936812)$	$a_3^{(2)}$	$[-0.017, -0.002)$	$a_3^{(3)}$	$[-0.007, 0.003)$
$a_4^{(1)}$	$[7936812, 10333398)$	$a_4^{(2)}$	$[-0.002, 0.012)$	$a_4^{(3)}$	$[0.003, 0.009)$
$a_5^{(1)}$	$[10333398, 12972818)$	$a_5^{(2)}$	$[0.012, 0.029)$	$a_5^{(3)}$	$[0.009, 0.023)$
$a_6^{(1)}$	$[12972818, 26578726)$	$a_6^{(2)}$	$[0.029, \bar{\beta}]$	$a_6^{(3)}$	$[0.023, \bar{\beta}]$
$a_7^{(1)}$	$[26578726, 64794206)$				
$a_8^{(1)}$	$[64794206, \bar{\beta}]$				

Appendix D - Two-Phase Tabu Search settings

Algorithm TSP1 stops once 2000 iterations have been performed (parameter iterP1_{max}). Parameter q is increased by 1 after 250 consecutive iterations have been performed without any improvement

(parameter iterNoImprP1$_{max}$). The maximum value that parameter q can take is 2 (parameter q_{max}). This implies that procedure JumpingP1 is called after performing 500 consecutive iterations without any improvement. We set τ, i.e. the number of iterations a partition of $\mathbb{A}_n$ remains tabu, equal to a random integer number $\lceil 4 \times U[0,1] \times k \times N \rceil$, where $U[0,1]$ is a uniformly distributed random number between 0 and 1. Parameter δ, i.e. the number of moves to perform in procedure JumpingP1, is set equal to a random integer number $\lceil U[2, \min(6, k \times N)] \rceil$. Finally, parameter Δ, i.e. the constant value used to construct the sequence of ϵ-constraint problems (20)-(21), is set equal to 0.025. Given that $m_{\lambda_1^{II}} = 0$ and $m_{\lambda^{II}_{\sharp(\mathcal{AF}_k)}} \leq 1$ by construction, setting $\Delta = 0.025$ implies that the sequence built is composed of at most 39 ϵ-constraint problems (20)-(21).

As the main scope of this paper is to solve problem (18)-(19), where the contiguity constraint holds, we set iterP2$_{max}$:= 0 for Phase 2 in all the computational experiments, except for the computational results presented in Subsection 5.3, where a comparison of the performance of the Two-Phase Tabu Search with that of the heuristic introduced in Cerqueti et al. (2013) is provided. For those experiments we set iterP2$_{max}$:= 1000 and iterNoImprP2$_{max}$:= 100.

Appendix E - Two-Phase Tabu Search: Detailed Computational Results

Table 12: Detailed comparison of different approximations of the efficient frontier for the Spanish instance introduced in Cerqueti et al. (2013). For each value of γ, we report the coordinates ($m_{\lambda^{II}}$, $d_{\lambda^{II}}$) of the heuristic solutions found by the Tabu Search algorithm presented in Cerqueti et al. (2013), in columns 2 and 3, the coordinates of the heuristic solutions found by procedure TSP1 along with the corresponding computing times (in seconds), in columns 4-6, and the same statistics for procedure TSP1+TSP2, in columns 7-9.

	Tabu Search		TSP1			TSP1+TSP2		
γ	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.025	0.026	0.196	0.027	0.144	2.761	0.027	0.144	5.351
0.05	0.051	0.411	0.052	0.306	2.496	0.052	0.306	5.101
0.075	0.075	0.481	0.075	0.378	2.293	0.075	0.378	4.914
0.1	0.122	0.521	0.104	0.456	2.434	0.100	0.444	4.977
0.125	0.132	0.544	0.133	0.544	2.512	0.126	0.500	5.211
0.15	0.157	0.650	0.153	0.581	2.527	0.153	0.581	5.210
0.175	0.178	0.673	0.179	0.593	2.512	0.179	0.593	5.273
0.2	0.209	0.695	0.202	0.630	2.496	0.202	0.630	5.070
0.225	0.231	0.739	0.225	0.674	2.527	0.235	0.652	5.070
0.25	0.250	0.748	0.258	0.696	2.496	0.258	0.689	5.101
0.275	0.282	0.763	0.281	0.733	2.434	0.281	0.733	4.992
0.3	0.306	0.796	0.305	0.778	2.543	0.305	0.778	4.977
0.325	0.329	0.878	0.328	0.815	2.465	0.328	0.815	4.867
0.35	0.353	0.922	0.352	0.874	2.433	0.352	0.874	4.898
0.375	0.376	0.948	0.376	0.922	2.308	0.377	0.922	4.633
0.4	0.401	0.981	0.400	0.959	2.231	0.400	0.959	4.446
0.425	0.426	1.126	0.425	1.022	2.012	0.426	1.019	3.962
0.45	0.450	1.137	0.450	1.111	1.966	0.450	1.111	3.900
0.475	0.510	1.193	0.485	1.170	1.965	0.485	1.170	3.806
0.5	0.510	1.193	0.510	1.204	2.075	0.510	1.204	3.962
0.525	0.546	1.226	0.534	1.215	2.106	0.534	1.215	3.978
0.55	0.558	1.248	0.559	1.259	2.044	0.559	1.259	3.729
0.575	0.582	1.304	0.583	1.296	1.919	0.583	1.296	3.557
0.6	0.607	1.307	0.607	1.348	2.340	0.607	1.348	4.118
0.625	0.644	1.363	0.644	1.367	2.168	0.644	1.367	3.915
0.65	0.655	1.385	0.656	1.389	2.153	0.656	1.389	3.822
0.675	0.680	1.430	0.680	1.422	2.028	0.680	1.422	3.651
0.7	0.704	1.474	0.704	1.467	1.966	0.704	1.467	3.432
0.725	0.729	1.511	0.729	1.511	1.950	0.729	1.511	3.338
0.75	0.754	1.578	0.753	1.556	1.903	0.753	1.556	3.245
0.775	0.778	1.630	0.778	1.600	1.888	0.778	1.600	3.120
0.8	0.803	1.667	0.803	1.667	1.841	0.803	1.667	2.917
0.825	0.827	1.711	0.827	1.711	1.841	0.827	1.711	2.808
0.85	0.852	1.756	0.852	1.756	1.825	0.852	1.756	2.746

continued on next page

Table 12 – continued from previous page

	Tabu Search		TSP1			TSP1+TSP2		
γ	$m_{\lambda II}$	$d_{\lambda II}$	$m_{\lambda II}$	$d_{\lambda II}$	CPU (secs.)	$m_{\lambda II}$	$d_{\lambda II}$	CPU (secs.)
0.875	0.877	1.785	0.877	1.800	1.856	0.877	1.800	2.668
0.9	0.901	1.822	0.901	1.822	1.685	0.901	1.822	2.387
0.925	0.926	1.867	0.926	1.867	1.809	0.926	1.867	2.605
0.95	0.951	1.911	0.951	1.911	1.825	0.951	1.911	2.433
0.975	0.975	1.956	0.988	1.978	1.732	0.988	1.978	2.169
1	1.000	2.000	1.000	2.000	0.000	1.000	2.000	0.000
Total CPU (secs.)		211.550			84.365			156.359
Average CPU (secs.)		5.424			2.163			4.009

Table 13: Detailed comparison of different approximations of the efficient frontier for the German instance introduced in Cerqueti et al. (2013). For each value of γ, we report the coordinates ($m_{\lambda II}$, $d_{\lambda II}$) of the heuristic solutions found by the Tabu Search algorithm presented in Cerqueti et al. (2013), in columns 2 and 3, the coordinates of the heuristic solutions found by procedure TSP1 along with the corresponding computing times (in seconds), in columns 4-6, and the same statistics for procedure TSP1+TSP2, in columns 7-9.

	Tabu Search		TSP1			TSP1+TSP2		
γ	$m_{\lambda II}$	$d_{\lambda II}$	$m_{\lambda II}$	$d_{\lambda II}$	CPU (secs.)	$m_{\lambda II}$	$d_{\lambda II}$	CPU (secs.)
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.025	0.025	0.172	0.030	0.156	2.808	0.027	0.125	6.068
0.05	0.050	0.294	0.051	0.243	3.198	0.050	0.238	6.224
0.075	0.075	0.517	0.080	0.373	3.105	0.080	0.373	5.991
0.1	0.100	0.554	0.103	0.512	3.058	0.101	0.459	6.147
0.125	0.134	0.627	0.130	0.593	2.886	0.130	0.593	5.725
0.15	0.151	0.681	0.152	0.629	2.886	0.152	0.629	5.741
0.175	0.176	0.733	0.182	0.700	2.839	0.182	0.700	5.694
0.2	0.201	0.841	0.201	0.782	2.714	0.201	0.782	5.507
0.225	0.231	0.969	0.229	0.830	2.465	0.229	0.830	5.382
0.25	0.252	1.023	0.251	0.852	2.590	0.251	0.852	5.413
0.275	0.282	1.072	0.290	0.973	2.590	0.290	0.973	5.273
0.3	0.325	1.197	0.312	0.995	2.605	0.312	0.995	5.195
0.325	0.326	1.202	0.328	1.017	2.480	0.328	1.017	5.101
0.35	0.364	1.237	0.351	1.050	2.543	0.351	1.050	5.179
0.375	0.386	1.259	0.382	1.121	2.496	0.382	1.121	5.133
0.4	0.417	1.318	0.438	1.345	2.543	0.438	1.345	5.117
0.425	0.438	1.345	0.438	1.345	2.558	0.438	1.345	5.038
0.45	0.460	1.367	0.465	1.400	2.527	0.465	1.400	4.945
0.475	0.484	1.396	0.488	1.411	2.496	0.488	1.411	4.961
0.5	0.504	1.411	0.504	1.425	2.636	0.504	1.411	4.820
0.525	0.527	1.433	0.527	1.447	2.574	0.527	1.447	4.836
0.55	0.551	1.484	0.551	1.499	2.715	0.551	1.484	4.930
0.575	0.576	1.510	0.576	1.510	2.715	0.576	1.510	4.961
0.6	0.601	1.545	0.601	1.545	2.714	0.601	1.545	4.976
0.625	0.635	1.550	0.635	1.550	2.745	0.635	1.550	5.195
0.65	0.658	1.562	0.658	1.562	2.730	0.658	1.562	5.241
0.675	0.682	1.584	0.682	1.584	2.870	0.682	1.584	5.242
0.7	0.706	1.599	0.706	1.599	2.871	0.706	1.599	5.211
0.725	0.729	1.621	0.730	1.656	2.870	0.730	1.656	5.070
0.75	0.753	1.654	0.754	1.674	2.917	0.754	1.674	4.570
0.775	0.777	1.696	0.785	1.689	2.902	0.785	1.689	5.195
0.8	0.801	1.702	0.801	1.702	3.011	0.801	1.702	4.899
0.825	0.833	1.739	0.825	1.732	3.229	0.825	1.732	5.023
0.85	0.857	1.783	0.857	1.783	3.385	0.857	1.772	4.961

continued on next page

Table 13 – continued from previous page

	Tabu Search		TSP1			TSP1+TSP2		
γ	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)
0.875	0.880	1.805	0.880	1.805	3.447	0.881	1.801	4.976
0.9	0.905	1.835	0.904	1.835	3.432	0.904	1.835	4.852
0.925	0.928	1.868	0.928	1.868	3.307	0.928	1.868	4.649
0.95	0.952	1.912	0.952	1.912	3.401	0.952	1.912	4.602
0.975	0.976	1.956	0.976	1.956	3.057	0.976	1.956	4.071
1	1.000	2.000	1.000	2.000	0.000	1.000	2.000	0.000
Total CPU (secs.)		619.669			110.915			202.114
Average CPU (secs.)		15.889			2.844			5.182

Table 14: Points composing the approximations of the efficient frontiers for the electricity and the stock-index cases. We report the coordinates ($m_{\lambda^{II}}$, $d_{\lambda^{II}}$) of the heuristic solutions found by procedure TSP1 along with the corresponding computing times (in seconds). The points highlighted in bold represent the heuristic solutions λ_B^{II}, for the electricity case, and λ_T^{II}, for the stock-index case, chosen as inputs of the bootstrap method.

Electricity case				Stock-index case			
Point	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)	Point	$m_{\lambda^{II}}$	$d_{\lambda^{II}}$	CPU (secs.)
1	0.000	0.000	0.000	1	0.000	0.000	0.000
2	0.025	0.312	185.940	2	0.025	0.802	100.593
3	0.051	0.508	125.502	3	0.054	1.164	76.752
4	0.080	0.633	158.482	4	0.080	1.362	86.830
5	0.117	0.834	231.910	5	0.107	1.485	75.052
6	0.142	0.963	216.044	6	0.134	1.605	97.485
7	0.178	1.104	184.978	7	0.165	1.645	116.298
8	0.227	1.295	226.900	8	0.192	1.680	124.036
9	0.275	1.447	256.201	9	0.228	1.732	145.657
10	**0.366**	**1.507**	**311.611**	10	0.256	1.752	165.110
11	0.407	1.552	315.963	**11**	**0.298**	**1.763**	**153.349**
12	0.435	1.598	292.017	12	0.330	1.783	163.348
13	0.461	1.621	287.852	13	0.382	1.822	180.258
14	0.494	1.636	267.774	14	0.409	1.822	231.898
15	0.523	1.688	272.033	15	0.448	1.842	240.615
16	0.573	1.726	263.157	16	0.475	1.849	213.923
17	0.613	1.757	244.156	17	0.508	1.868	223.673
18	0.661	1.788	217.792	18	0.568	1.868	224.656
19	0.698	1.845	194.548	19	0.644	1.882	252.533
20	0.728	1.869	188.261	20	0.686	1.888	225.702
21	0.770	1.883	178.963	21	0.774	1.908	247.713
22	0.807	1.910	154.238	22	0.809	1.914	210.616
23	0.860	1.948	142.272	23	0.876	1.921	193.347
24	0.914	1.974	105.051	24	0.917	1.928	176.796
25	0.997	1.993	43.914	25	0.948	1.934	143.364
26	1.000	2.000	0.000	26	0.976	1.967	90.215
				27	1.000	2.000	0.000
Total CPU (secs.)			5065.847				4159.819
Average CPU (secs.)			211.077				166.401

Appendix F - Information Loss and Distance Measures

The purpose of this section is to highlight how the optimization model presented in this paper fits with the scopes of a bootstrap procedure: to maintain the statistical properties of the original sample (similarity), while allowing for a sufficient level of diversification between the original and the bootstrapped series (multiplicity). At this aim, we propose a discussion on the distance indicator

and multiplicity measure adopted here on the basis of information theory. The same notation used throughout the paper is adopted.

Define a functional space $\mathcal{G}$ whose elements $g \in \mathcal{G}$ act on the multivariate Markov chain $\mathbf{X}$ by defining a new Markov chain $\tilde{\mathbf{X}}$ obtained by considering as single states the elements of a partition of $\mathbb{A}^k$. There is a clear bijection between the g of $\mathcal{G}$ and the partitions λ of $\mathbb{A}^k$. Hereafter, we will denote the partition generated by g as λ_g. Moreover, we denote as $\mathcal{I}_g$ the σ-algebra associated to g, which collects the information generated by λ_g. Hence, we can write $\tilde{\mathbf{X}} = \mathbf{X}|\mathcal{I}_g$ to formalize that the new Markov chain is the original Markov chain $\mathbf{X}$ conditioned to the information provided by $\mathcal{I}_g$.

It is worth noting that a clustering procedure of the kN-states of $\mathbb{A}^k$ implies information loss. In fact, the exact identification of the transition probabilities from a given kN-state of $\mathbb{A}^k$ to the N-states of $\mathbb{A}$ gets lost when the kN-state is joined with others in a partition class. On this ground and following the seminal work of Kolmogorov (1965), we are able to explore the meaning of $\mathcal{I}_g$ -and so of g and of the corresponding λ_g- in terms of information loss. At this aim, we define a nonnegative functional $\eta_{\mathbf{X}} \in [0, \bar{\eta}]$, which measures the information loss related to $\mathcal{I}_g$ when passing from the original Markov chain $\mathbf{X}$ to the new one $\tilde{\mathbf{X}}$. Specifically, $\eta_{\mathbf{X}}(\tilde{\mathbf{X}})$ may be viewed as a *distance measure* between $\mathbf{X}$ and $\tilde{\mathbf{X}}$, which increases as the loss of information does. When $\eta_{\mathbf{X}}(\tilde{\mathbf{X}}) = 0$, no information is lost, while $\eta_{\mathbf{X}}(\tilde{\mathbf{X}}) = \bar{\eta}$ means that $\tilde{\mathbf{X}}$ provides the maximum level of information loss (no information left in passing from $\mathbf{X}$ to $\tilde{\mathbf{X}}$).
The consistency requirements in the boundary situations of 0 and $\bar{\eta}$ lead to specific situations in our setting. We list them below, along with a brief explanation.

(A.1) $\eta_{\mathbf{X}}(\mathbf{X}) = 0$. This is the corner situation of full information and occurs when $\tilde{\mathbf{X}} = \mathbf{X}$. In this case the partition λ_g of the state space -generated by $\mathcal{I}_g$- is the finest one: λ_g is the partition keeping separate all the states of the Markov chain (*singleton partition*).

(A.2) If $\mathcal{I}_g = \{\emptyset, \Omega\}$ -where Ω represents the sample space of the probability space where the Markov chain is defined-, then $\eta_{\mathbf{X}}(\tilde{\mathbf{X}})$ attains its maximum. Also in this case we have a corner situation, where the maximum level of information is lost. In fact, the corresponding partition λ_g is the smallest one, and collects all the elements of the state space in a unique set (*all-comprehensive partition*). The transition probability matrix of the new Markov chain $\tilde{\mathbf{X}}$ is trivially of dimension $1 \times \#(\mathbb{A})$, and no information can be derived from the knowledge of the specific kN-state associated to the past realizations of the chain.

Therefore, in terms of information loss, the two bootstrap requirements can be translated as trying to minimize the loss of information (similarity) without achieving the complete absence of information loss (multiplicity). Hence, a constrained minimization problem should be developed. This is precisely the argument of Subsection 2.5. In this respect, it is important to point out that the distance indicator $d_{\lambda_\mathbf{v}}$ and the multiplicity measure $m_{\lambda_\mathbf{v}}$ -introduced and used in the paper- are two specific information loss distance measures η, which indeed satisfy conditions (A.1) and (A.2) (for a discussion on this, refer to Subsection 2.5).