A Tabu Search Heuristic Procedure in Markov Chain Bootstrapping[1]

Roy Cerqueti

Università degli Studi di Macerata - Dipartimento di Istituzioni Economiche e Finanziarie
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 Metodi Quantitativi
Contrada Santa Chiara, 50 - 25122 Brescia BS - Italy
E-mail: {falbo, guastaro, pelizcri}@eco.unibs.it

*Corresponding author. Tel.: +39 030 2988516. Fax: +39 030 2400925. E-mail: pelizcri@eco.unibs.it

Abstract

Markov chain theory is proving to be a powerful approach to bootstrap finite states processes, especially where time dependence is non linear. In this work we extend such approach to bootstrap discrete time continuous-valued processes. To this purpose we solve a minimization problem to partition the state space of a continuous-valued process into a finite number of intervals or unions of intervals (i.e. its states) and identify the time lags which provide "memory" to the process. A distance is used as objective function to stimulate the clustering of the states having similar transition probabilities. The problem of the exploding number of alternative partitions in the solution space (which grows with the number of states and the order of the Markov chain) is addressed through a Tabu Search algorithm. The method is applied to bootstrap the series of the German and Spanish electricity prices. The analysis of the results confirms the good consistency properties of the method we propose.

Keywords: Markov Chains, Bootstrapping, Heuristic Algorithms, Tabu Search, Electricity Prices.

[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

Bootstrapping is a method largely applied in different research areas ranging from natural to social sciences as the special issue of Statistical Science in 2003 (vol. 18, no. 2) has shown. After the initial proposal by Efron (1979) several developments have appeared in the literature. A group of bootstrap methods which have followed the seminal idea of Efron and have been largely applied in the economic and finance literature is based on re-sampling of a model errors (e.g., see Freedman 1984, Freedman and Peters 1984, Efron and Tibshirani 1993 for a methodological discussion and Brock et al. 1992 for an application to financial markets). This approach suffers in general of model misspecification risk.

Bühlmann (2002) compares the block, the sieve, and the local methods of bootstrapping. These methods are nonparametric, model-free and assume that observations can be (time) dependent, as opposed to the original solution advanced by Efron which requires that observations are independent. In another work, Guastaroba et al. (2009) compare several bootstrap and parametric simulation methods to generate scenarios embedding the relevant features of stock market returns. They find that the risk-return trade-off for the portfolios optimized under the different scenarios is the best out-of-sample when block bootstrap is used.

Among the nonparametric bootstrap methods, a relatively recent method has appeared in the literature, which is based on the Markov chains of order k. In their work Anatolyev and Vasnev (2002) propose a method for a finite state discrete time Markov chain, where the "next step" of the bootstrap procedure is driven by transition probabilities whose conditional event is a trajectory observed on the last k time lags. The strength points of this method are linked to its natural ability to capture arbitrary dependency structures of a time series. Such an advantage is of primary importance, since it relieves a researcher of the responsibility of assuming a model, of estimating complex parameters, and of losing important parts of those structures.

In this paper we apply Markov chain theory to bootstrap series of processes taking continuous values. To the best of our knowledge, this is the first attempt to extend Markov chain bootstrapping in this sense. Some major difficulties have to be solved.

A first and obvious one is that of discretizing the state space of the process into a finite number of "relevant states". In other words, the continuous state space must be partitioned into a number of states which are representative of the levels where the process modifies significantly its dynamics (i.e. its expected value, its variance, etc.). Previous literature adopting Markov chain bootstrap methods, such as papers that appeared in the area of information theory (e.g., see Bühlmann and Wyner 1999, Bühlmann 2002, Zhu et al. 2002), has focused on discrete-valued processes with a finite and fixed number of states, which are all assumed to be relevant.

A second relevant issue is that of assessing the order of the process, i.e. the length of its memory. Several contributions have addressed the problem of identifying the "true" order of a Markov chain, proposing occasionally quite sophisticated methods. In this sense Bühlmann and Wyner (1999) and Bühlmann (2002) are remarkable works as they allow a process to vary its memory depending on the states that the process crosses through the time. The authors consider a tree-based representation of the state space of the Markov chain and propose an algorithm to find the minimal state space. Once a distance function between two probability measures is defined, the algorithm prunes iteratively each leaf node if its distance from the parent node is smaller than a given threshold. As an outcome of the method, the states are kept distinct following a hierarchical structure based on time lags (e.g., time lags $k+1$, $k+2$, ... cannot be relevant to keep states distinct if time lag k is not).

Cerqueti et al. (2009) advance a method to identify *both* the relevant states of a process and the relevance of time lags (without assuming any hierarchical dependence). In particular their approach consists of letting some initially specified states to group together (and so forming a new larger state) if such states have similar transition probabilities. The occurrence of few large groups at some time lags (as opposed to the formation of several small clusters in other time lags) is taken as evidence against the relevance of those time lags. The authors formulate an optimization problem whose solution space is represented by all

the feasible partitions which can be obtained from grouping the rows of the original transition probability matrix. A distance indicator is introduced as the objective function to be minimized. The optimal solution identifies the partition which groups the states having the most similar transition probabilities. A multiplicity constraint is also introduced to avoid that the optimum is found selecting the so called "singleton partition", which is the partition letting each row to form a one element group.
In this work we tackle a major problem which has remained unsolved in Cerqueti et al. (2009). We refer in particular to the fast growing number of the possible partitions which can be formed considering a process with N states and k time lags. Theoretically such number gets large very soon (it is based on the k-th power of the Bell number of N) and the computing time required to evaluate the entire solution space goes beyond any acceptable boundary if we just consider the case of 7 states and 5 time lags (i.e. $5.18798E + 14$ partitions, corresponding to more than $164,500$ years of computing time, letting 0.01 seconds to compute one solution). A heuristic is therefore required to solve real life instances where it is common to face bigger cases than the one previously considered. In particular we implement a heuristic based on a Tabu Search framework.
Partitioning problems are among the most studied in the OR community (e.g., see Anily and Federgruen 1991). In fact, in many important combinatorial optimization problems, such as clustering and bin packing, an optimal partition into groups has to be determined for a finite collection of objects. Several authors have proposed iterative procedures for partitioning objects. For instance the k-means algorithm introduced by Forgy (1965) and MacQueen (1967), and its variations, based on the idea of assigning each element to the cluster whose centroid (usually the average value of all the objects in the cluster) is the nearest considering a given distance measure. Other authors have proposed graph-theoretic approaches to data clustering (e.g., see Wu and Leahy 1993, and references therein), where the data to be clustered are represented by vertices of an undirected graph and the cost of an edge connecting any two vertices is chosen to reflect the similarity (as defined by a given function) between the pair of linked vertices. More recently, the use of metaheuristics to solve the partitioning problem has been investigated by a number of authors. For instance, Sung and Jin (2000) design a Tabu Search-based heuristic, whereas Trejos et al. (2004) implement an Ant Colony algorithm. Ma et al. (2006) use an evolutionary algorithm to identify clusters of records in a database. Most of the former methods, designed for general partitioning problems, can be adapted to solve the problem of partitioning a transition probability matrix in the domain of Markov chains of order one, once a distance measure among transition probabilities has been defined. Other authors have proposed decomposition approaches to solve the problem. Courtois (1977) advances a decomposition approach that performs firstly a one-step approximation to obtain the steady state probability vector by a block diagonal matrix decomposition and then an aggregation method based on the positive eigenvectors of each block. Takahashi (1984) proposes an algorithm based on the observation that, given any partition of the state space, the exact Kolmogorov equations of the whole system can be split into two subsets, one accounting for the links among the partitions (the aggregation step), the other one accounting only for the links inside the partitions (the disaggregation step). An iterative numerical technique can then be devised for recursively solving the two sets of equations until convergence is reached. Spears (1998) proposes an algorithm to compress a transition probability matrix into a smaller matrix with less states. The choice of which states are worth compressing is based on a similarity metric that measures the distance between states based on their transition probabilities, i.e. states with similar transition behaviors are aggregated together to form a new compressed state. Tabu Search algorithms have shown to be quite appropriate also for solving complex financial optimization problems. As an example, we mention the contributions by Ehrgott et al. (2004) and Woodside-Oriakhi et al. (2011), where extensions of the classical Markowitz mean-variance portfolio optimization model are introduced. The proposed models are solved by means of several heuristics, including Tabu Search-based algorithms.
As previously pointed out, in this paper we consider the problem of finding the best partitions of the states for each of the time lags of a Markov chain of order up to k. This represents a constraint which cannot be (directly) addressed through the aforementioned methods.

To show the properties of the bootstrapping method proposed here, an application is developed to the case of electricity prices. Such series are usually considered as "hard cases" in the literature, because of several features: hourly, weekly, and annual seasonality, strong mean reversion, time varying volatility, and occasional presence of "spikes" where prices show sudden jumps which are soon followed by symmetrical reversals. For a review of the difficulties in modeling electricity prices and the literature developed to solve them, see, for example, Bunn (2004), Huisman and Mahieu (2003), Weron et al. (2004), and Weron (2006). In this application we generate several bootstrapped series from the Spanish and German electricity markets and analyze the most relevant parameters of their distributions.
Section 2 of the paper gives the mathematical basis of the problem. Section 3 discusses methodological aspects of the Tabu Search algorithm and gives details on the bootstrap procedure. Section 4 describes the application of the method to the price series of the German and the Spanish electricity market. The computational results are commented in Section 5. Finally, Section 6 provides some concluding remarks.

2 Optimization Model

For the sake of clarity, we introduce the optimization problem in a stepwise form. Appendix A provides a quick reference summary of some notation introduced in the following.

2.1 Problem description

Consider an evolutive phenomenon taking values in an interval $[\alpha, \beta] \subseteq \mathbb{R}$. Suppose that M consecutive realizations are observed and let $O = (x_1, ..., x_M)$ be the set of these time-ordered observations. O is the original sample, and the bootstrapped series will be derived from it. Define a partition of $[\alpha, \beta]$ into N initial states, or intervals, $a_1, \ldots, a_N$, such that $a_i \cap a_j = \varnothing$, $i \neq j$, and $\bigcup_{h=1}^{N} a_h = [\alpha, \beta]$, and collect them in $A = \{a_1, \ldots, a_N\}$.
Consider the observed k-order state (or, simply, k-state) in O starting at time l, and denote it as $\mathbf{x}_{l,k} = (x_l, ..., x_{l+k-1})$. Let us collect all the k-states that can be extracted from O in a set O_k. Therefore O_k contains $M - k + 1$ k-states. Since the x's belong to some a's, we can describe a given observed k-state $\mathbf{x}_{l,k}$ through a unique sequence of elements of A, call it $\tilde{\mathbf{a}}_{l,k} = (a_l, ..., a_{l+k-1})$, where a_t is an element of A assigned to time t and $t = l, ..., l + k - 1$. Let A_k be the set of all the $\tilde{\mathbf{a}}_{l,k}$. Notice that there is a one-to-one correspondence between $\tilde{\mathbf{a}}_{l,k}$ and $\mathbf{x}_{l,k}$, because the sequence of values in $\mathbf{x}_{l,k}$ and the sequence of intervals in $\tilde{\mathbf{a}}_{l,k}$ are indexed by the same time. Of course for some different starting times s and l it can happen that $\tilde{\mathbf{a}}_{s,k} \equiv \tilde{\mathbf{a}}_{l,k}$, i.e. two k-states replicate. We introduce the set of all the possible k-states of a's (without any reference to the starting time) and denote it as A^k: this set is the Cartesian product of A with itself k times. We point out that the cardinality of A^k is N^k. Hence, we have N^k k-states in A^k, and we will refer to any k-state of A^k as $\mathbf{a}_{h,k}$, with $h = 1, ..., N^k$.
To proceed, it is useful to recall the basic idea of our problem. The observed sample O can be replicated by using a k-th order Markov chain $\{X(t); t \geq 0\}$ with state space A. If the cardinality of A is large, the replications may become mathematically hard to treat. We want to gain mathematical tractability by reducing the cardinality of the state space and by selecting the order k, without losing too much information. Since the partition of $[\alpha, \beta]$ is arbitrary, it can be uselessly rich of states. Following the idea of Cerqueti et al. (2009), we seek to reduce the initial number of states joining them based on their transition probabilities. As we will see, this objective will be pursued through an optimal search among the possible partitions of A^k.

2.2 Transition probability matrices

Consider $a_z \in A$, $z = 1, \ldots, N$, and $\mathbf{a}_{h,k} = (a_{h_k}, a_{h_{k-1}}, \ldots, a_{h_1}) \in A^k$, where $h_w = 1, ..., N$ points to an element of A for time lag w of k-state h.

The transition probability of k-state $\mathbf{a}_{h,k}$ to state a_z is defined as:

$$P(a_z|\mathbf{a}_{h,k}) = P(X(t) = a_z|X(t-1) = a_{h_1}, \ldots, X(t-k) = a_{h_k}). \tag{1}$$

The k-th order transition probability matrix of the phenomenon contains the $P(a_z|\mathbf{a}_{h,k})$'s, for all $a_z \in A$ and $\mathbf{a}_{h,k} \in A^k$. The empirical frequency of k-state $\mathbf{a}_{h,k}$ evolving to state a_z is denoted as $f(a_z|\mathbf{a}_{h,k})$ and corresponds to the cardinality of $F_{a_z|\mathbf{a}_{h,k}}$, where:

$$F_{a_z|\mathbf{a}_{h,k}} = \left\{ \tilde{\mathbf{a}}_{l,k} : \tilde{\mathbf{a}}_{l,k} \equiv \mathbf{a}_{h,k}, a_{l+k} \equiv a_z \right\}. \tag{2}$$

In other words, $F_{a_z|\mathbf{a}_{h,k}}$ is the set of all the observed k-states $\tilde{\mathbf{a}}_{l,k}$ replicating k-state $\mathbf{a}_{h,k}$ and evolving to state a_z at time $l+k$.
Based on the definition of Ching et al. (2008), we estimate the k-state transition probability of $X(t)$ by using the empirical frequencies $f(a_z|\mathbf{a}_{h,k})$ related to the observed phenomenon:

$$\hat{P}(a_z|\mathbf{a}_{h,k}) = \begin{cases} \frac{f(a_z|\mathbf{a}_{h,k})}{\sum_{j=1}^{N} f(a_j|\mathbf{a}_{h,k})} & \text{if } \sum_{j=1}^{N} f(a_j|\mathbf{a}_{h,k}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The estimated k-th order transition probability matrix of the phenomenon contains the $\hat{P}(a_z|\mathbf{a}_{h,k})$'s and is denoted as $\mathcal{M}$.
Now, denote as Λ the set of the partitions of A. Then, a partition $\lambda \in \Lambda$ can be represented as $\lambda = \{A_{\lambda,1}, \ldots, A_{\lambda,\#(\lambda)}\}$, where $\{A_{\lambda,q}\}_{q=1,\ldots,\#(\lambda)}$ is a partition of nonempty subsets of A and $\#(\lambda)$ denotes the cardinality of set λ. Notice that $1 \leq \#(\lambda) \leq N$.
Extending to a multidimensional context, we introduce the set of *multidimensional partitions* $\boldsymbol{\lambda}$ of A^k and denote it as Λ^k. This set consists of the k replications of the set Λ and is used to describe the time dependent partitions of A. More precisely Λ^k is defined as:

$$\Lambda^k = \{\boldsymbol{\lambda} = (\lambda_k, \lambda_{k-1}, \ldots, \lambda_1) : \lambda_w \in \Lambda,\ w = 1, \ldots, k\}, \tag{4}$$

where λ_w, $w = 1, \ldots, k$, is the *unidimensional partition* of A referred to time lag w.
The definition of the transition probability can be extended to the case where the conditional event is a class of partition $\boldsymbol{\lambda}$ instead of a k-state. By referring to the probability law P introduced in Eq. (1), we define:

$$P(a_z|\mathbf{A}_{\boldsymbol{\lambda},h,k}) = P(X(t) = a_z|X(t-1) \in A_{\lambda_1,h_1}, \ldots, X(t-k) \in A_{\lambda_k,h_k}), \tag{5}$$

where $a_z \in A$, $h_w \in \{1, \ldots, \#(\lambda_w)\}$, for each $w = 1, \ldots, k$, and:

$$\mathbf{A}_{\boldsymbol{\lambda},h,k} = (A_{\lambda_k,h_k}, A_{\lambda_{k-1},h_{k-1}}, ..., A_{\lambda_1,h_1}) \subseteq A^k \tag{6}$$

can be viewed as the h-th class of partition $\boldsymbol{\lambda}$ expressed as an ordered sequence of classes extracted from partitions $\lambda_k, \lambda_{k-1}, \ldots, \lambda_1$.
Probability $P(a_z|\mathbf{A}_{\boldsymbol{\lambda},h,k})$ in Eq. (5) is estimated through the empirical frequencies analogously to Eq. (3):

$$\hat{P}(a_z|\mathbf{A}_{\boldsymbol{\lambda},h,k}) = \begin{cases} \frac{\sum_{i:\mathbf{a}_{i,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}} f(a_z|\mathbf{a}_{i,k})}{\sum_{i:\mathbf{a}_{i,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}} \sum_{j=1}^{N} f(a_j|\mathbf{a}_{i,k})} & \text{if } \sum_{i:\mathbf{a}_{i,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}} \sum_{j=1}^{N} f(a_j|\mathbf{a}_{i,k}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Probabilities $\hat{P}(a_z|\mathbf{A}_{\boldsymbol{\lambda},h,k})$'s generate a new transition matrix.
To simplify the notation, we use hereafter the symbol P to refer both to the transition probabilities P and to their estimates $\hat{P}$.

2.3 Formalization of the optimal partition problem

We are now able to formalize the optimization problem. Two competing objectives should be pursued: statistical similarity and multiplicity of the bootstrapped series with respect to the original sample O. We achieve this aim by defining a distance indicator (to take into account similarity) and a multiplicity measure, both to be calculated on the transition probabilities of the classes of the k-dimensional partition $\boldsymbol{\lambda} \in \Lambda^k$. A deep discussion on this point is developed in Section 3.1 of Cerqueti et al. (2009).
The *distance indicator* of partition $\boldsymbol{\lambda}$ is based on the absolute difference between the elements of the k-th order transition probability matrix $\mathcal{M}$. We can define a distance $d_{i,j}$ between two k-states $\mathbf{a}_{i,k}$ and $\mathbf{a}_{j,k}$ as follows:

$$d_{i,j} := \sum_{z=1}^{N} \left| P(a_z|\mathbf{a}_{i,k}) - P(a_z|\mathbf{a}_{j,k}) \right|.$$

Notice that similarity of bootstrapped series with respect to the original sample is achieved if $\mathbf{a}_{i,k}$ and $\mathbf{a}_{j,k}$ are grouped together when their distance $d_{i,j}$ is close to zero. As we shall see, this evidence will drive the formalization of the optimization problem. The definition of distance can be extended to a class $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ of partition $\boldsymbol{\lambda}$ as follows:

$$d_{\mathbf{A}_{\boldsymbol{\lambda},h,k}} := \max_{i,j:\mathbf{a}_{i,k},\mathbf{a}_{j,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}} d_{i,j}.$$

Analogously, the distance $d_{\boldsymbol{\lambda}}$ of partition $\boldsymbol{\lambda}$ is given by the average value of its classes distances:

$$d_{\boldsymbol{\lambda}} := \frac{1}{D} \cdot \sum_{h=1}^{\#(\boldsymbol{\lambda})} d_{\mathbf{A}_{\boldsymbol{\lambda},h,k}} \cdot \#\left(\mathbf{A}_{\boldsymbol{\lambda},h,k}\right),$$

where $\#\left(\mathbf{A}_{\boldsymbol{\lambda},h,k}\right)$ is the cardinality of partition class $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ and $D = \sum_{h=1}^{\#(\boldsymbol{\lambda})} \#\left(\mathbf{A}_{\boldsymbol{\lambda},h,k}\right)$. The absolute multiplicity measure of partition $\boldsymbol{\lambda}$, denoted as $l_{\boldsymbol{\lambda}}$, is based on the size of the partition classes:

$$l_{\boldsymbol{\lambda}} := \sum_{h=1}^{\#(\boldsymbol{\lambda})} \#\left(\mathbf{A}_{\boldsymbol{\lambda},h,k}\right)^2.$$

In our optimization problem we rather apply the relative *multiplicity measure* $m_{\boldsymbol{\lambda}}$, i.e. a normalization of $l_{\boldsymbol{\lambda}}$:

$$m_{\boldsymbol{\lambda}} := \frac{\sqrt{l_{\boldsymbol{\lambda}}} - \sqrt{D}}{D - \sqrt{D}},$$

with $D = \sum_{h=1}^{\#(\boldsymbol{\lambda})} \#\left(\mathbf{A}_{\boldsymbol{\lambda},h,k}\right)$. The main properties of $d_{\boldsymbol{\lambda}}$ and $m_{\boldsymbol{\lambda}}$ are introduced and discussed in Sections 3.2 and 3.4 in Cerqueti et al. (2009). It is worth recalling that $m_{\boldsymbol{\lambda}} \in [0,1]$, while $d_{\boldsymbol{\lambda}} \in [0,2]$ (see Propositions 4 and 9 in Cerqueti et al. 2009).
We now present the optimization problem based on $d_{\boldsymbol{\lambda}}$ and $m_{\boldsymbol{\lambda}}$.

Definition 1 *Let us consider $\gamma \in [0,1]$, $k^* \in \{1,...,N\}$, and $\boldsymbol{\lambda}^* = (\lambda^*_{k^*}, \lambda^*_{k^*-1}, \ldots, \lambda^*_1) \in \Lambda^{k^*}$.*
We say that the couple $(k^*, \boldsymbol{\lambda}^*)$ is γ-optimal if and only if it is the solution of the following minimization problem:

$$\min_{(k,\boldsymbol{\lambda}) \in \mathbb{N} \times \Lambda^k} d_{\boldsymbol{\lambda}} \tag{8}$$

$$\text{s.t.} \quad m_{\boldsymbol{\lambda}} \geq \gamma. \tag{9}$$

The optimal k^* represents the optimal order of a Markov chain $\{X^*(t); t \geq 0\}$ describing the evolutive phenomenon, while $\boldsymbol{\lambda}^*$ is the optimal partition of A^{k^*}. Given such optimally chosen Markov chain, the corresponding k^*-th order transition probability matrix $\mathcal{M}^*$ is estimated by means of Eq. (7).
It will turn out to be useful the interpretation of the results of the optimization problem through the introduction of the *optimal efficient frontier*, defined as follows:

Definition 2 *Set $k \in \{1, ..., N\}$ to a chosen value. The optimal efficient frontier $\mathcal{F}_k$ related to optimization problem (8)-(9) is:*

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

and $\boldsymbol{\lambda}^*$ is the solution of the problem:

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \in \Lambda^k} d_{\boldsymbol{\lambda}} \\ &\text{s.t.} \quad m_{\boldsymbol{\lambda}} \geq \gamma. \end{aligned}$$

An illustrative example. Consider a Markov chain $\{X(t); t \geq 0\}$ of order 2, with state space $A = \{1, 2\}$. For ease of exposition, the two initial states, or intervals, a_1 and a_2 partitioning the state space $[\alpha, \beta]$ of the phenomenon are denoted with 1 and 2, respectively. Assume the following set of time ordered observations of the phenomenon:

$$O = (1, 2, 1, 1, 2, 2, 1).$$

The possible 2-states $\mathbf{a}_{h,2} = (a_{h_2}, a_{h_1}) \in A^2$ are:

$$\mathbf{a}_{1,2} = (1, 1),\ \mathbf{a}_{2,2} = (1, 2),\ \mathbf{a}_{3,2} = (2, 1), \text{ and } \mathbf{a}_{4,2} = (2, 2).$$

Let $\mathcal{M}$ denote the 2-state transition probability matrix of the Markov chain approximating the observed phenomenon. We have:

$\mathcal{M} =$

	states a_z	
2-states $\mathbf{a}_{h,2}$	1	2
$(1,1)$	0	1
$(1,2)$	0.5	0.5
$(2,1)$	1	0
$(2,2)$	1	0

with $z = 1, 2$, $a_z \in A$, $h = 1, ..., 4$.
Our problem is to find the optimal partition $\boldsymbol{\lambda}^*$ of the states for each time lag. We are faced with a number of feasible solutions (i.e. partitions):

- $\boldsymbol{\lambda}^1 = \left(\lambda_2^1, \lambda_1^1\right) = (\{\{1\}, \{2\}\}, \{\{1\}, \{2\}\})$ (states separated at time lag 1 and time lag 2: singleton partition),
- $\boldsymbol{\lambda}^2 = \left(\lambda_2^2, \lambda_1^2\right) = (\{1, 2\}, \{\{1\}, \{2\}\})$ (states separated at time lag 1 and joined at time lag 2),
- $\boldsymbol{\lambda}^3 = \left(\lambda_2^3, \lambda_1^3\right) = (\{\{1\}, \{2\}\}, \{1, 2\})$ (states separated at time lag 2 and joined at time lag 1),
- $\boldsymbol{\lambda}^4 = \left(\lambda_2^4, \lambda_1^4\right) = (\{1, 2\}, \{1, 2\})$ (states joined at time lag 1 and time lag 2: all-comprehensive partition).

The corresponding transition probability matrices are:

$\mathcal{M}^1 = \mathcal{M} =$

	states a_z	
partition classes $\mathbf{A}_{\boldsymbol{\lambda}^1,h,2}$	1	2
$\{\mathbf{a}_{1,2}\}$	0	1
$\{\mathbf{a}_{2,2}\}$	0.5	0.5
$\{\mathbf{a}_{3,2}\}$	1	0
$\{\mathbf{a}_{4,2}\}$	1	0

,

$\mathcal{M}^2 =$

	states a_z	
partition classes $\mathbf{A}_{\boldsymbol{\lambda}^2,h,2}$	1	2
$\{\mathbf{a}_{1,2}, \mathbf{a}_{3,2}\}$	0.5	0.5
$\{\mathbf{a}_{2,2}, \mathbf{a}_{4,2}\}$	0.67	0.33

,

$$\mathcal{M}^3 = \begin{array}{l|cc} & \text{states } a_z & \\ \text{partition classes } \mathbf{A}_{\boldsymbol{\lambda}^3,h,2} & 1 & 2 \\ \hline \{\mathbf{a}_{1,2}, \mathbf{a}_{2,2}\} & 0.33 & 0.67 \\ \{\mathbf{a}_{3,2}, \mathbf{a}_{4,2}\} & 1 & 0 \end{array},$$

and

$$\mathcal{M}^4 = \begin{array}{l|cc} & \text{states } a_z & \\ \text{partition classes } \mathbf{A}_{\boldsymbol{\lambda}^4,h,2} & 1 & 2 \\ \hline \{\mathbf{a}_{1,2}, \mathbf{a}_{2,2}, \mathbf{a}_{3,2}, \mathbf{a}_{4,2}\} & 0.6 & 0.4 \end{array}.$$

Notice that partition $\boldsymbol{\lambda}^3$ is trivially preferred to $\boldsymbol{\lambda}^1$ as it groups the third and fourth 2-states which evolve in the same way. Among the previous partitions, the optimal solution of problem (8)-(9) will be chosen. The partition composed by two classes, the first including 2-state $\mathbf{a}_{1,2}$ and the second containing 2-states $\mathbf{a}_{2,2}$, $\mathbf{a}_{3,2}$, and $\mathbf{a}_{4,2}$ is not a feasible solution of our problem. This partition does not belong to the set Λ^2 defined in Eq. (4), since it does not fulfill the requirement of time dependency.

3 Solution Methodology

First of all, we need to adapt to our case the definition of ϵ-active time lag, which has been introduced in Cerqueti et al. (2009) and will be useful in the Tabu Search algorithm.

Definition 3 *Given $\epsilon \in [0,1]$ and $w \in \{1,\dots,k\}$, a time lag w is said ϵ-active if and only if, for any $a_z \in A$, the following conditions are fulfilled:*

- $|P(a_z|\mathbf{a}_{i,k}) - P(a_z|\mathbf{a}_{j,k})| < \epsilon$, *where $\mathbf{a}_{i,k}$ can differ from $\mathbf{a}_{j,k}$ in all times but $t-w$,for any couple i,j,*
- *ϵ is the smallest number satisfying the previous inequality.*

The smaller the ϵ the more an *ϵ-active* time lag is relevant. More precisely, for an *ϵ-active* time lag w the smaller is ϵ the more the observation of the process in $t-w$ brings a "key information" to determine its evolution at time t.

Solving computationally hard problems like (8)-(9) within reasonable computing times by means of exact solution procedures, such as implicit enumeration methods or Dynamic Programming, is possible only for very small-scale instances. This provides us with the motivation of developing an efficient heuristic procedure to find a near optimal solution in reasonable computing time. To this purpose, we introduce a suitable Tabu Search algorithm. We assume that the reader is already familiar with the Tabu Search framework and we refer to Glover and Laguna (1997) for a detailed description of the metaheuristic.
In this section, we first describe the Tabu Search algorithm and then introduce the procedure used to generate the bootstrapped time series.

3.1 Tabu Search algorithm

The present subsection is organized as follows. Firstly, an intuitive description of the Tabu Search algorithm is provided. Secondly, the behavior of the algorithm in two iterations is shown by means of an illustrative example. Finally, the mechanics of each element of the heuristic are detailed.

The Tabu Search algorithm starts its search from a solution provided by the researcher, as it is detailed in Subsection 3.1.1. Then, the algorithm chooses at each iteration one time lag w and selects the best unidimensional partition λ_w (i.e. the partition of A at time lag w). The selection of w is driven by its ϵ-activeness. Specifically, before the Tabu Search algorithm starts, we compute the value of ϵ for each time lag (see Definition 3). Subsequently, we build up a discrete probability distribution based on the

ϵ-activeness of each time lag, so that the more ϵ-active the time lag, the higher the probability to be selected. Specifically, the probability of time lag $\bar{w}$, denoted as $P(\bar{w})$, is calculated as follows:

$$P(\bar{w}) = \frac{1 - \epsilon_{\bar{w}}}{\sum_{w=1}^{k}(1 - \epsilon_w)}, \tag{10}$$

where $\epsilon_{\bar{w}}$ is the value of ϵ for time lag $\bar{w}$. The rationale of this choice is to give priority to those time lags that bring a key information to determine the evolution of the process. Once a time lag w has been selected, the incumbent partition is modified only in its w-th component (i.e. λ_w), keeping fixed the others. What we obtain is a set of *neighbor* solutions (see in the following), since $\boldsymbol{\lambda}$ is computed modifying only one component, and *feasible*, since we consider only those solutions that do not violate constraint (9). Additionally, those partitions that are tabu are neglected, unless they lead to a solution better than any other already computed (the *aspiration criterion*), as it is further explained in Subsection 3.1.4. The iteration completes choosing the partition minimizing the objective function (8).

An illustrative example. Consider the instance reported in Table 1, where the 2-states of a second order Markov chain with $A = \{1, 2, 3\}$ are partitioned. For ease of exposition, the three initial states, or intervals, a_1, a_2, and a_3 partitioning the state space $[\alpha, \beta]$ of the phenomenon are denoted with 1, 2, and 3, respectively. Assume to solve problem (8)-(9) for $\gamma = 0.975$ and assume that the initial solution is partition $\boldsymbol{\lambda}^1 = \left(\lambda_2^1, \lambda_1^1\right) = (\{1, 2, 3\}, \{1, 2, 3\})$. Partition $\boldsymbol{\lambda}^1$ is shown in the first column of Table 1. Assume that time lag 2 is selected by the Tabu Search algorithm in the first iteration. This means that in the current iteration the algorithm modifies partition $\lambda_2^1 = \{1, 2, 3\}$ while keeping fixed $\lambda_1^1 = \{1, 2, 3\}$. So far no partition is tabu, then all the partitions of states at time lag 2 that belong to the neighborhood are evaluated. Let us assume that the neighborhood is composed of the following unidimensional partitions: $\lambda_2^i = \{\{1, 2\}, \{3\}\}$, $\lambda_2^{ii} = \{\{1, 3\}, \{2\}\}$, and $\lambda_2^{iii} = \{\{1\}, \{2, 3\}\}$. Among these partitions, the one leading to the best feasible multidimensional partition is selected. Assume that partition λ_2^i is selected. This leads to multidimensional partition $\boldsymbol{\lambda}^2 = \left(\lambda_2^i, \lambda_1^1\right) = (\{\{1, 2\}, \{3\}\}, \{1, 2, 3\})$, reported in the second column of Table 1, which becomes the incumbent solution. Then partition $\lambda_2^i = \{\{1, 2\}, \{3\}\}$ is made tabu for a certain number of the following iterations (see Subsection 4.5). In the second iteration, assume that the Tabu Search algorithm selects time lag 1. All the partitions of states at time lag 1 belonging to the neighborhood are evaluated while keeping partition λ_2^i unchanged. Let us assume that the neighbor partitions are $\lambda_1^i = \{\{1, 2\}, \{3\}\}$, $\lambda_1^{ii} = \{\{1, 3\}, \{2\}\}$, and $\lambda_1^{iii} = \{\{1\}, \{2, 3\}\}$. Assume that partition λ_1^{iii} is selected. Hence, the multidimensional partition selected after two iterations is partition $\boldsymbol{\lambda}^3 = \left(\lambda_2^i, \lambda_1^{iii}\right) = (\{\{1, 2\}, \{3\}\}, \{\{1\}, \{2, 3\}\})$, reported in the third column of Table 1, which becomes the incumbent solution for the following iteration.

Table 1: An example of two iterations of the Tabu Search algorithm.

Partition classes $\mathbf{A}_{\boldsymbol{\lambda}^1,h,2}$	Partition classes $\mathbf{A}_{\boldsymbol{\lambda}^2,h,2}$	Partition classes $\mathbf{A}_{\boldsymbol{\lambda}^3,h,2}$
$\{(1,1),$	$\{(1,1),$	$\{(1,1),$
$(1,2),$	$(1,2),$	$(2,1)\}$
$(1,3),$	$(1,3),$	$\{(1,2),$
$(2,1),$	$(2,1),$	$(1,3),$
$(2,2),$	$(2,2),$	$(2,2),$
$(2,3),$	$(2,3)\}$	$(2,3)\}$
$(3,1),$	$\{(3,1),$	$\{(3,1)\}$
$(3,2),$	$(3,2),$	$\{(3,2),$
$(3,3)\}$	$(3,3)\}$	$(3,3)\}$
The initial solution.	*Partitioning time lag* 2.	*Partitioning time lag* 1.

3.1.1 Initial solution

We are interested in computing the whole *efficient frontier approximation*, defined in our Tabu Search framework according to Definition 2. Basically the efficient frontier approximation is the set of solution points $(m_{\boldsymbol{\lambda}}, d_{\boldsymbol{\lambda}})$ such that, setting a value for the multiplicity boundary γ, the heuristic cannot improve below $d_{\boldsymbol{\lambda}}$ (i.e. the distance in the objective function) satisfying the constraint $m_{\boldsymbol{\lambda}} \geq \gamma$. Such representation of the solutions is useful to make a more appropriate choice for $\boldsymbol{\lambda}$ (which will be used in the bootstrap procedure) among the various alternatives which result changing $\gamma \in [0, 1]$. The efficient frontier approximation is computed through a sequence of $n-1$ executions of the Tabu Search algorithm, fixing at each time a specific value of parameter γ and providing an initial solution. In particular, the interval $[0, 1]$ is divided into n subintervals of equal length. By construction we know the two partitions corresponding to the extreme points of the efficient frontier approximation. The singleton partition (i.e. the partition where each distinct k-state $\mathbf{a}_{h,k}$ forms a one element class) corresponds to the point with $m_{\boldsymbol{\lambda}} = 0$ and $d_{\boldsymbol{\lambda}} = 0$. The all-comprehensive partition (i.e. the partition where all the k-states form a unique class) has $m_{\boldsymbol{\lambda}} = 1$, while the value of $d_{\boldsymbol{\lambda}}$ is not known *a priori* but is bounded by 2.
Moving from the starting point, i.e. the all-comprehensive partition, the following points of the frontier are obtained performing the heuristic for each value of γ separating the subintervals. The initial solution considered at any execution is the solution obtained at the previous execution of the Tabu Search algorithm.

3.1.2 Neighborhood

Assume an incumbent solution $\boldsymbol{\lambda}$ and consider the unidimensional partition λ_w, $w = 1, \ldots, k$. Let us denote the subset of unidimensional partitions obtained from λ_w by performing a local change on it as its *neighborhood* $\mathcal{N}(\lambda_w)$. Specifically, the Tabu Search algorithm uses two kinds of moves that define $\mathcal{N}(\lambda_w)$:

1. *1-insertion move*: remove state i from its current class and insert it into another not empty class,
2. *1-creation move*: if state i is not the only element in the class, remove it and create a new class containing i as the only element.

An illustrative example. Consider the unidimensional partition $\lambda = \{\{1, 2\}, \{3\}\}$. From this partition, 3 partitions can be obtained performing 3 different 1-insertion moves. Specifically, one can remove state 1 from the first class and insert it into the second class leading to partition λ^i. Partition λ^{ii} is obtained removing state 2 from the first class and inserting it into the second one. Finally, removing state 3 from the second class and inserting it into the first one leads to partition λ^{iii}. On the other hand, two 1-creation moves are available consisting in removing state 1 or state 2 from its current class and creating a new class, leading to the same partition λ^{iv}.

Table 2: An example about how to construct the neighborhood.

Initial partition	1-insertion moves	1-creation moves
$\lambda = \{\{1, 2\}, \{3\}\}$	$\lambda^i = \{\{2\}, \{1, 3\}\}$ $\lambda^{ii} = \{\{1\}, \{2, 3\}\}$ $\lambda^{iii} = \{\{1, 2, 3\}\}$	$\lambda^{iv} = \{\{1\}, \{2\}, \{3\}\}$

3.1.3 Random selection of time lags

One time lag is selected and evaluated at each iteration by the Tabu Search algorithm. As already mentioned above, the procedure used for selecting randomly the incumbent time lag consists of the generation of an integer random number $w \in \{1, \ldots, k\}$ according to the discrete probability distribution defined in Eq. (10). In this way the procedure modifies the incumbent partition considering preferably

the time lags which have the highest importance (the most ϵ-active time lags have higher probabilities to be selected).

3.1.4 Tabu list

k tabu lists TL_w, $w = 1, \ldots, k$, are used, i.e. one tabu list for each time lag w. Basically, when an unidimensional partition λ_w is selected, then λ_w is inserted in TL_w meaning that it is tabu (forbidden) to select again partition λ_w for a certain number of the following iterations (see Subsection 4.5 for further details).

Aspiration criterion

A unidimensional partition λ_w included in the tabu list can be chosen again (i.e. its tabu status is revoked) if its selection leads to a feasible solution which is better than the best-known solution $\boldsymbol{\lambda}^H$ found so far.

3.1.5 Stopping criterion

The search is stopped after a fixed number of iterations (parameter *MaxIterations*) has been executed.

3.1.6 Tabu Search algorithm

A step-by-step description of our Tabu Search is provided in Algorithm 1.

Algorithm 1 Procedure: TabuSearch.

Input: a feasible partition $\boldsymbol{\lambda}$ of A^k.
Output: the best-known feasible partition $\boldsymbol{\lambda}^H$ of A^k.

1: Set $\boldsymbol{\lambda}^H \leftarrow \boldsymbol{\lambda}$ and $TL_w \leftarrow \varnothing$, $w = 1, ..., k$.
2: **while** iterations $\leq$ *MaxIterations* **do**
3: Select randomly time lag $\bar{w} \in \{1, ..., k\}$.
4: Consider unidimensional partitions $\lambda_{\bar{w}} \in \mathcal{N}(\lambda_{\bar{w}})$ evaluating all the 1-insertion and all the 1-creation moves. Let $\boldsymbol{\lambda} := (\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1)$, where partitions λ_w, $w \neq \bar{w}$, remain unchanged. Choose $\lambda'_{\bar{w}}$ such that $\lambda'_{\bar{w}} \notin TL_{\bar{w}}$ or $d_{\boldsymbol{\lambda}'} < d_{\boldsymbol{\lambda}^H}$, where $\boldsymbol{\lambda}'$ is feasible and $d_{\boldsymbol{\lambda}'}$ is minimized.
5: Set $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}'$.
6: Update $TL_{\bar{w}}$.
7: **if** $d_{\boldsymbol{\lambda}} < d_{\boldsymbol{\lambda}^H}$ **then**
8: Set $\boldsymbol{\lambda}^H \leftarrow \boldsymbol{\lambda}$.
9: Set iterNoImprov$\leftarrow$ 0.
10: **else**
11: Set iterNoImprov$\leftarrow$iterNoImprov+1.
12: **if** iterNoImprov$>$ *MaxIterNoImprov* **then**
13: Call Algorithm 2.
14: Set iterNoImprov$\leftarrow$ 0.
15: **end if**
16: **end if**
17: **end while**

Some initialization tasks are first performed in Step 1. Particularly, the input feasible partition $\boldsymbol{\lambda}$ of A^k is the initial solution provided by the user. Then, the Tabu Search begins in Step 2 and performs the subsequent instructions *MaxIterations* times. Firstly, in Step 3 a time lag $\bar{w}$ is randomly selected. The corresponding unidimensional partition $\lambda_{\bar{w}}$ is then considered in Step 4 to construct the neighborhood $\mathcal{N}(\lambda_{\bar{w}})$. Set $\mathcal{N}(\lambda_{\bar{w}})$ consists of all the unidimensional partitions at time lag $\bar{w}$ that can be obtained performing 1-insertion or 1-creation moves. Any other unidimensional partition λ_w, with $w \neq \bar{w}$, remains unchanged. Among the unidimensional partitions in $\mathcal{N}(\lambda_{\bar{w}})$ only those that provide a feasible solution

$\boldsymbol{\lambda} := (\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1)$ are considered. Then, the unidimensional partition $\lambda'_{\bar{w}}$ leading to the best feasible partition $\boldsymbol{\lambda}' := \left(\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda'_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1\right)$ is selected provided that $\lambda'_{\bar{w}}$ is not tabu or that the aspiration criterion is satisfied. Partition $\boldsymbol{\lambda}'$ becomes the new incumbent solution $\boldsymbol{\lambda}$ (Step 5). Partition $\lambda'_{\bar{w}}$ is made tabu in Step 6. Finally, other update operations are performed from Step 7 to Step 11, whereas in case *MaxIterNoImprov* iterations have been consecutively performed without improvements, the following diversification strategy is performed (Steps 12 to 15).

3.1.7 Diversification strategy

In order to escape from local optima, we have designed jumps that move the search to a different portion of the feasible region once the neighborhood of a solution has been intensively explored. The procedure we have designed for performing jumps is reported in Algorithm 2. It is performed every time that a maximum number of iterations without improvement (parameter *MaxIterNoImprov*) have been consecutively performed.
A jump consists of a sequence of 1-insertion and 1-creation moves. Specifically, choosing a time lag $\bar{w} \in \{1, ..., k\}$ and given the incumbent partition $\boldsymbol{\lambda} := (\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1)$, the algorithm looks for the unidimensional partition $\lambda'_{\bar{w}}$ leading to the best feasible partition $\boldsymbol{\lambda}' := \left(\lambda_k, ..., \lambda_{\bar{w}+1}, \lambda'_{\bar{w}}, \lambda_{\bar{w}-1}, ..., \lambda_1\right)$. Partition $\lambda'_{\bar{w}}$ is obtained performing 1-insertion and 1-creation moves from $\lambda_{\bar{w}}$. This operation is repeated for each time lag $w \in \{1, ..., k\}$. Once the best move has been found for each time lag, an ordered list L of k moves is created by sorting the corresponding objective function values from the smallest to the largest. Subsequently, a random positive integer number δ is generated. The new solution $\boldsymbol{\lambda}'$ is obtained performing iteratively the first δ moves in list L, provided that that specific move does not lead to an infeasible solution. The random number δ is generated in the interval $[\eta, \omega]$, where $\eta > 0$ must be sufficiently large to ensure jumping to a solution reasonably different from the incumbent one, and $\omega \geq \eta$ is a parameter less than or equal to k.

Algorithm 2 Procedure: JUMPING.

Input: the incumbent partition $\boldsymbol{\lambda}$ of A^k.
Output: a new feasible partition $\boldsymbol{\lambda}'$ of A^k.

1: Set $TL_w \leftarrow \varnothing$, $w = 1, ..., k$.
2: For each time lag $\bar{w} \in \{1, ..., k\}$ find the unidimensional partition $\lambda'_{\bar{w}} \in \mathcal{N}(\lambda_{\bar{w}})$ that leads to the best feasible partition $\boldsymbol{\lambda}'$ keeping unchanged the unidimensional partitions λ_w, $w \neq \bar{w}$, of $\boldsymbol{\lambda}$. Set $\mathcal{N}(\lambda_{\bar{w}})$ contains unidimensional partitions obtained by evaluating all the 1-insertion and all the 1-creation moves from $\lambda_{\bar{w}}$.
3: Sort the selected moves in non decreasing order of the corresponding objective function values and create list L.
4: Determine partition $\boldsymbol{\lambda}'$ performing sequentially the first δ moves in list L.
5: **if** $d_{\boldsymbol{\lambda}'} < d_{\boldsymbol{\lambda}^H}$ **then**
6: Set $\boldsymbol{\lambda}^H \leftarrow \boldsymbol{\lambda}'$.
7: **end if**

Notice that in the jumping procedure the tabu lists are reinitialized to empty sets (Step 1) before evaluating the possible moves, i.e. no move is then forbidden.

3.2 Bootstrap procedure

The bootstrap procedure is described in Algorithm 3. The algorithm takes as input the heuristic partition $\boldsymbol{\lambda}^H$ and the corresponding k-th order transition probability matrix $\mathcal{M}^H$, as well as the observed time series $O = (x_1, ..., x_M)$, and returns a bootstrapped series of length lgh. Matrix $\mathcal{M}^H$ is estimated by means of Eq. (7). The core of the algorithm consists of a random choice among the k-states belonging to a class of partition $\boldsymbol{\lambda}^H$. Indeed, the aim of partitioning the k-states of A^k into $\boldsymbol{\lambda}^H$ is to get to a coarser structure of the information, sufficient to guarantee that the bootstrapped series maintain a good statistical similarity

with the original series and keep a satisfactory level of multiplicity.
Let us recall that we can describe a k-state starting at time l by means of its values ($\mathbf{x}_{l,k} = (x_l, ..., x_{l+k-1}) \in O_k$) or by means of its initial states (i.e. the intervals of A which the values belong to, $\tilde{\mathbf{a}}_{l,k} = (a_l, ..., a_{l+k-1}) \in A_k$). We stress that there is a one-to-one correspondence between the values in $\mathbf{x}_{l,k}$ and the intervals in $\tilde{\mathbf{a}}_{l,k}$.
To avoid the rare but not impossible trapping of the bootstrap procedure, the last k-state of A_k is excluded in the case it has never been observed before.
To the sake of readability, we will denote as $\tilde{\mathbf{a}}_{l,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}$ when the sequence of k initial states in $\tilde{\mathbf{a}}_{l,k}$ belongs to partition class $\mathbf{A}_{\boldsymbol{\lambda},h,k}$, i.e. we neglect the starting time l.

Algorithm 3 Procedure: BOOTSTRAP.

Input: a partition $\boldsymbol{\lambda}^H$ of A^k and the corresponding k-th order transition probability matrix $\mathcal{M}^H$; the observed time series $O = (x_1, ..., x_M)$, with $M \geq 2k+1$; a cardinality $lgh \geq k+1$.
Output: a bootstrapped series, i.e. a time-ordered set $\tilde{O} = (\tilde{x}_1, ..., \tilde{x}_{lgh})$.

1: Set $\tilde{\mathbf{x}}_{1,k} = (\tilde{x}_1, ..., \tilde{x}_k) \leftarrow \mathbf{x}_{k+1,k} = (x_{k+1}, ..., x_{2k})$.
2: **for** j=1 **to** $lgh - k$ **do**
3: Set $\tilde{\mathbf{x}}_{j,k} \leftarrow (\tilde{x}_j, ..., \tilde{x}_{j+k-1})$ and let $\tilde{\mathbf{a}}_{j,k} = (a_j, ..., a_{j+k-1})$ be the corresponding k-state.
4: **if** $\tilde{\mathbf{a}}_{j,k} \in A_k$ **then**
5: Let $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ be the partition class of $\tilde{\mathbf{a}}_{j,k}$.
6: Choose uniformly among the k-states $\tilde{\mathbf{a}}_{l,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}$ that have passed the absolute continuity filter. Let $\tilde{\mathbf{a}}_{l^*(j),k} = (a_{l^*(j)}, ..., a_{l^*(j)+k-1})$ be the chosen k-state and $x_{l^*(j)+k}$ the value corresponding to $a_{l^*(j)+k}$.
7: Set $\tilde{x}_{j+k} \leftarrow x_{l^*(j)+k}$.
8: **else**
9: Set $r \leftarrow 1$.
10: Set $\tilde{\mathbf{x}}_{j,k-r} \leftarrow (\tilde{x}_{j+r}, ..., \tilde{x}_{j+k-1})$ and let $\tilde{\mathbf{a}}_{j,k-r} = (a_{j+r}, ..., a_{j+k-1})$ be the corresponding k-state.
11: **if** $\tilde{\mathbf{a}}_{j,k-r} \in A_{k-r}$ **then**
12: Find all the $(k-r)$-states $\tilde{\mathbf{a}}_{l,k-r}$ whose sequence of a's is the same of $\tilde{\mathbf{a}}_{j,k-r}$ and choose uniformly among them. Let $\tilde{\mathbf{a}}_{l^*(j+r),k-r} = (a_{l^*(j+r)}, ..., a_{l^*(j+r)+k-r-1})$ be the chosen $(k-r)$-state and $x_{l^*(j+r)+k-r}$ the value corresponding to $a_{l^*(j+r)+k-r}$.
13: Set $\tilde{x}_{j+k} \leftarrow x_{l^*(j+r)+k-r}$.
14: **else**
15: Set $r \leftarrow r+1$ and go to step 10.
16: **end if**
17: **end if**
18: **end for**

Algorithm 3 generates bootstrapped series of lgh observations, which we denote as $\tilde{O} = (\tilde{x}_1, ..., \tilde{x}_{lgh})$. Notice that the first k bootstrapped values are initialized to the observed values from time $k+1$ to time $2k$, $x_{k+1}, ..., x_{2k}$ (Step 1). Although the estimation of transition probability matrix $\mathcal{M}^H$ has been performed on all the values of the original series, we exclude from the bootstrap procedure the first k observed values in O, $x_1, ..., x_k$. Indeed none of these values can represent the possible evolution of a k-state, because by construction it is not possible to identify a whole k-state preceding each of the values $x_1, ..., x_k$.
The residual $lgh - k$ values are obtained iterating the instructions in Steps 2 to 18.
The standard steps for generating the values of the bootstrapped series are 5 to 7. Such instructions are performed when the last generated k intervals, i.e. $\tilde{\mathbf{a}}_{j,k}$, correspond to a k-state in A_k (Step 4). Firstly, the algorithm looks for the partition class of $\tilde{\mathbf{a}}_{j,k}$. Let that class be $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ (Step 5). Secondly, a random selection that is respectful of the transition probabilities $\hat{P}(a_z|\mathbf{A}_{\boldsymbol{\lambda},h,k})$'s contained in matrix $\mathcal{M}^H$ is done. The selection is performed in Step 6 among those k-states in $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ that fulfill the absolute continuity filter (see in the following for a detailed description). Thirdly, the value that follows the selected k-state is then added to the bootstrapped series (Step 7).
Steps 9 to 16 are designed to manage the possibility of generating k-states $\tilde{\mathbf{a}}_{j,k}$ which have not been observed. In fact, in such cases, relying only on $\mathcal{M}^H$, we would not be able to extract the next value

for the resampled series. To get over such hurdles we design a progressive "reduction of memory". This corresponds to iteratively reducing the order of the Markov chain. The recent history of the generated series is observed reducing progressively the order k (Steps 9 and 10). Set A_{k-r} contains all the observed $(k-r)$-states of intervals $\tilde{\mathbf{a}}_{l,k-r}$. If the last generated $(k-r)$-state $\tilde{\mathbf{a}}_{j,k-r}$ belongs to A_{k-r} (Step 11), a random selection is performed among the $(k-r)$-states of A_{k-r} that show the same sequence of intervals of $\tilde{\mathbf{a}}_{j,k-r}$ (Step 12). The value that follows the selected $(k-r)$-state is then added to the bootstrapped series (Step 13) and the algorithm moves to the next iteration.

3.2.1 Absolute continuity filter

As soon as a k-state is generated, its partition class $\mathbf{A}_{\boldsymbol{\lambda},h,k}$ is found and all the k-states belonging to the class are considered for selection. This class is indeed further refined to exclude those k-states which have transition probabilities not absolutely continuous to those of the currently generated k-state (Step 6). In probability theory, a probability μ is said absolutely continuous with respect to a probability ν if $\mu(B) = 0$ whenever $\nu(B) = 0$ for any event B. We adopt this definition in our context. More precisely, we impose that, given two k-states $\tilde{\mathbf{a}}_{j,k}, \tilde{\mathbf{a}}_{l,k} \in \mathbf{A}_{\boldsymbol{\lambda},h,k}$, where $\tilde{\mathbf{a}}_{j,k}$ is the one currently generated by the procedure, $\tilde{\mathbf{a}}_{l,k}$ is considered for selection if the following condition is satisfied:

$$\hat{P}(a_z|\tilde{\mathbf{a}}_{j,k}) = 0 \Rightarrow \hat{P}(a_z|\tilde{\mathbf{a}}_{l,k}) = 0,\ z = 1, ..., N.$$

This choice should reduce the chances (but not eliminate them) of generating bootstrapped series that evolve following paths which have never been observed.

4 Experimental Environment

4.1 Data description

The solution methodology, which uses a Tabu Search algorithm for the solution of the optimization problem and exploits the resulting partition $\boldsymbol{\lambda}^H$ of A^k to generate bootstrapped series, is now detailed to make clear the choices adopted to develop our application.
In this application we want to generate a family of bootstrapped series starting from the series of prices of two electricity markets. In particular, we consider the series of the daily spot price (in euros) for 1 MWh of the Spanish (Mibel Spanish Electric System Arithmetic Average Price) and German (EEX Phelix Day Base Price) markets in the periods $1/2/1998 - 12/31/2003$ and $6/17/2000 - 5/8/2007$, respectively. The time series of Spain consists of $M = 2,190$ observations, while the time series of Germany has $M = 2,517$ points.
These two series are interesting from a statistical point of view as they have features which make them "hard" to replicate. Fig. 1 and 2 show them.

A look at the figures allows us to detect:
- a weekly seasonality,
- a slightly positive trend,
- stochastic volatility,
- non linear dependency of data,
- two clear regimes of prices: normal trading and occasional short lived spike periods.

Spikes appear with no apparent regularity. Such events can be explained by sudden shortages on the supply side of the electricity system, or by unexpected and temporary increases of the demand (e.g., sudden meteorological events, driving to high consumption). Another regime switch occurs daily and is linked to day-light and night periods, usually referred to as periods of base and peak load. However we do not consider it here, as we use a daily electricity index, which is calculated as an arithmetic average of the 24 delivery hours.
All these features represent a serious challenge to a bootstrap procedure, which is based on classical

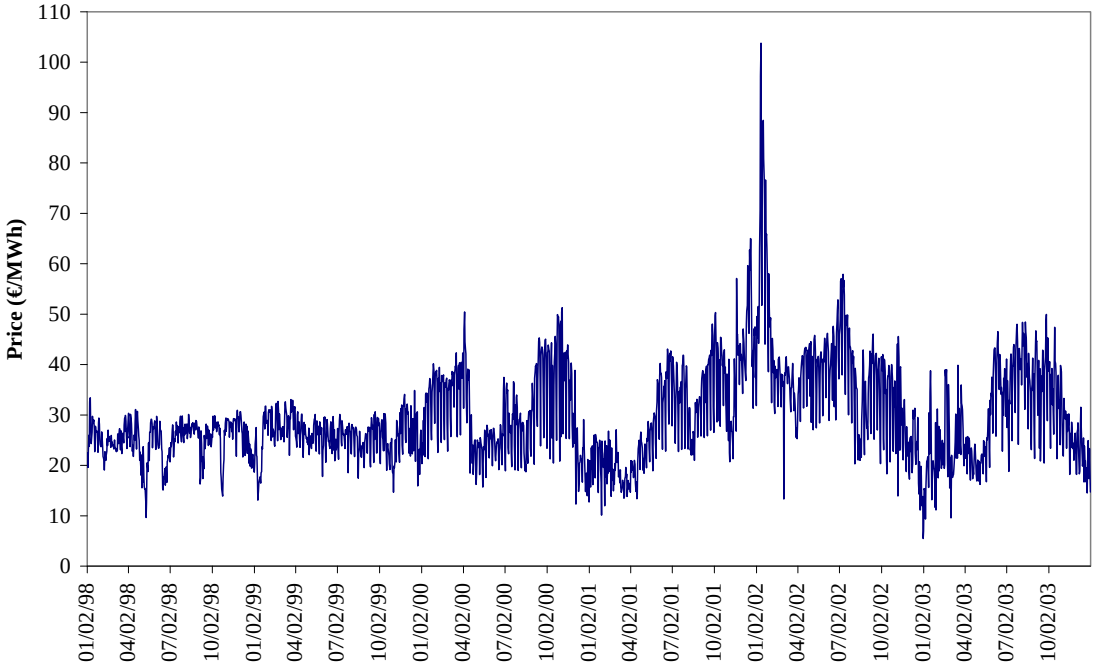

Figure 1: Spanish daily electricity prices.

methods like parametric or block sampling. In particular the parametric bootstrap requires to make a decision about the specification of the model, which is very difficult given the features of the series mentioned above. The block bootstrapping requires the strict stationarity of the original series, which is violated in this case. Data treatment to transform the original series into a stationary process is again very delicate, since the correction for stochastic volatility mixed with that of the occasional spikes is a hard task. On the contrary, such difficulties do not represent a serious challenge to a Markov chain approach to bootstrapping, since, as it is typical of data driven modelling methods, it adapts to any kind of time dependency among the data. Indeed in our application we only remove an exponential trend from the original data, and we re-apply it to the bootstrapped series.

4.2 Trend removal

The removal of an exponential trend rather than a linear one avoids the problem of generating (occasionally) negative prices. In particular the estimation of the exponential trend is based on the following model:

$$x_t^{(c)} = Ce^{rt+\psi_t}, \tag{11}$$

where $x_t^{(c)}$ are the raw original prices. If we assume that the time is continuous and that $\left\{\frac{\psi_t}{\sigma}\right\}_{t\geq 0}$ is a standard Brownian motion, we recognize $\left\{x_t^{(c)}\right\}_{t\geq 0}$ as the geometric Brownian motion with dynamics:

$$dx_t^{(c)} = (r + \frac{1}{2}\sigma^2)dt + \sigma dW_t,\ x_0^{(c)} = C,\ W_t = \frac{\psi_t}{\sigma}.$$

If we take the natural logarithm on both sides of Eq. (11), we obtain the following equation:

$$z_t = u + rt + \psi_t,$$

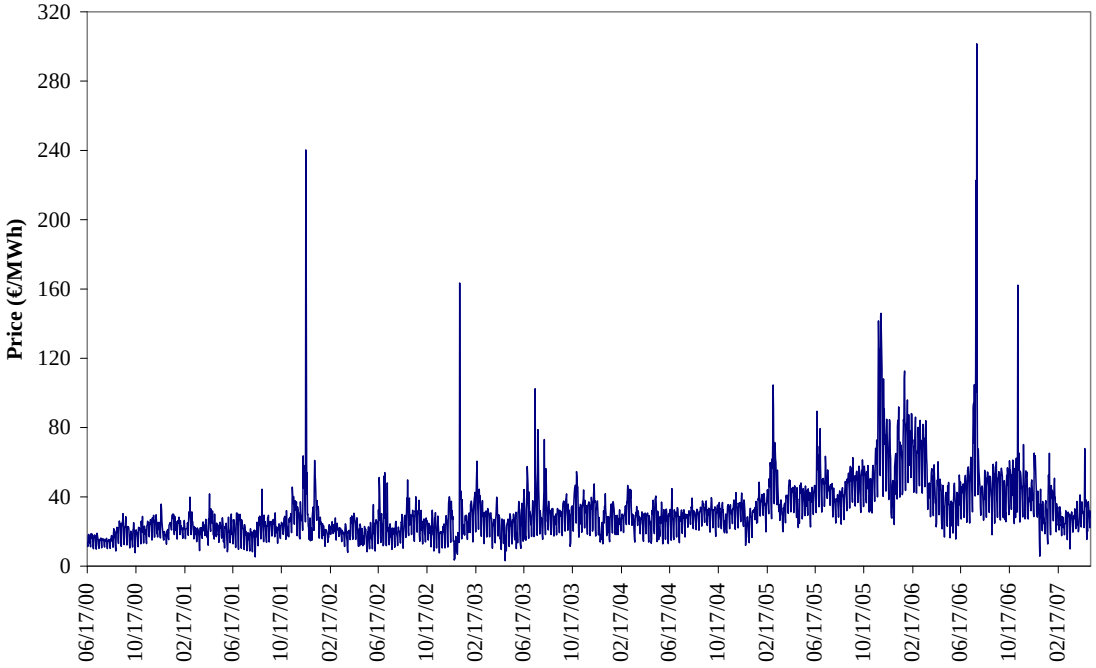

Figure 2: German daily electricity prices.

where $z_t = \ln x_t^{(c)}$, $u = \ln C$ is the intercept, r is the growth rate, and ψ_t are the errors.
For estimation purposes, let us assume that time is discrete and that the usual hypotheses of linear regression on the errors ψ_t hold. We obtain the following OLS estimates of r significant at a level of 5%:

$$\begin{aligned} \hat{r}_S &= 0.0001162404 \\ \hat{r}_G &= 0.0003867605, \end{aligned}$$

where S stands for Spain and G stands for Germany.
To the purpose of removing the exponential trend from our original series, we define the series of prices $O = (x_1, ..., x_M)$, where:

$$x_t = e^{z_t - \hat{r}t},\ t = 1, ..., M.$$

Set O is an input for the bootstrap procedure, while the output is the bootstrapped series $\tilde{O} = (\tilde{x}_1, ..., \tilde{x}_{lgh})$. To re-introduce the exponential trend in $\tilde{O}$, we multiply each point $\tilde{x}_j$ by $e^{\hat{r}j}$, $j = 1, ..., lgh$.
Another simple preliminary data treatment that we could perform is the removal of seasonalities. We have chosen not to handle annual seasonality because of the relatively short time series under analysis, while weekly seasonality has been modeled setting the order k of the Markov chain equal to 7.

4.3 Preliminary segmentation of the state space

The search for the relevant states of a continuous-valued process results from the aggregation of smaller segments of the state space of the process. To this purpose we partition the state space $[\alpha, \beta] \subseteq \mathbb{R}$ into N initial states, or intervals, $a_1, \ldots, a_N$, where N is set to a larger value than it is actually required. We collect the N initial states in set A.
In this application we set $N = 12$, which is a number indeed larger than it is usually expected to model electricity price regimes. At the same time, it was not possible to increase it further, as the transition probability matrix would tend to a 0-1 matrix, and the bootstrapped series would become replications of

the original sample. More precisely, if a k-state is specified through an excessively fine grid, then it will be observed only once in the original series. Consequently the estimate of its transition probability will consist of a 1 for the state to which that k-state has historically evolved and a 0 for all the other states. Let us call these k-states *deterministic k-states*.
To identify the 12 initial states, a univariate cluster procedure based on Ward's minimum-variance method (e.g., see Ward Jr. 1963) has been applied to each time series (after removal of trend). Adopting a Ward clustering favors the formation of clusters around the modal values of a distribution. Fig. 3 and 4 show the two detrended series together with the 12 intervals (separated by horizontal lines). Appendix B details this preliminary segmentation. For ease of exposition, we represent the set of states as $A = \{1, .., 12\}$.

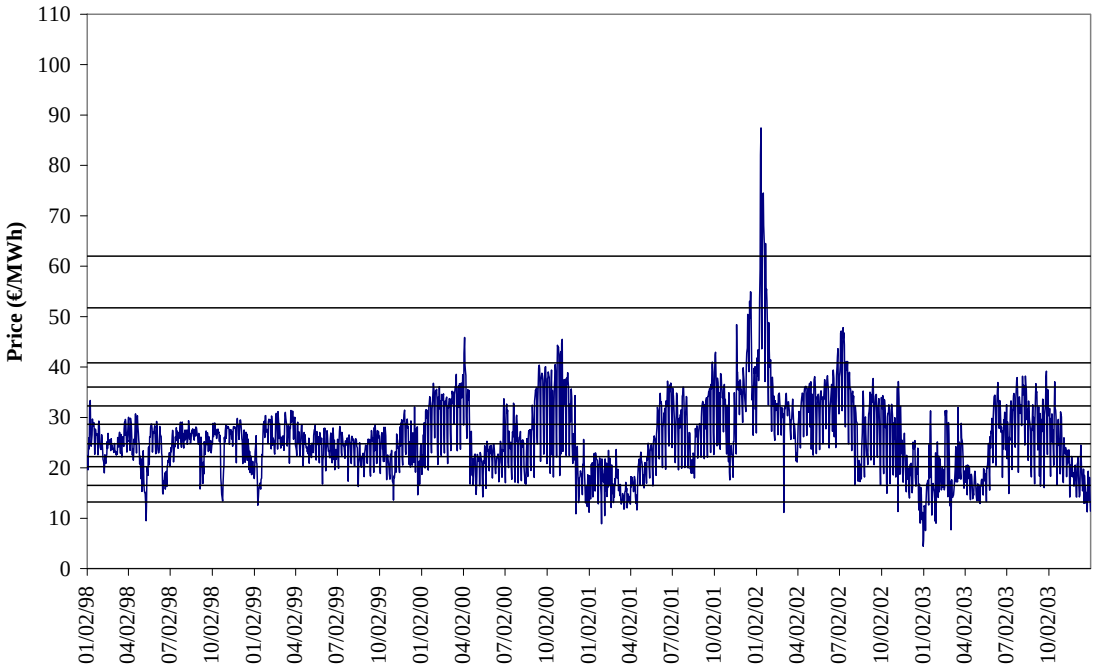

Figure 3: Spanish daily electricity prices without exponential trend and with grid of initial states, or intervals.

Based on such preliminary segmentation, the Tabu Search algorithm possibly joins several initial states into larger ones, to the aim of identifying the significant regimes of the phenomenon.

4.4 Partitions composition

Optimization problem (8)-(9) aims at obtaining both the optimal order k^* of a Markov chain $\{X^*(t); t \geq 0\}$ describing the evolutive phenomenon and the optimal partition $\boldsymbol{\lambda}^*$ providing the clustering of A^{k^*}. It is worth noting that both k^* and $\boldsymbol{\lambda}^*$ depend on the value of γ. As already mentioned at the end of Subsection 4.2, in our application we restrict the optimization problem by fixing $k = 7$. Nevertheless, the optimal k might be found by iterating the exact solution procedure as k varies from 1 to a sufficiently large value $\bar{k}$ and selecting the value of k, namely k^*, corresponding to the best solution.
The 7^{th} order transition probability matrices $\mathcal{M}_S$ and $\mathcal{M}_G$ estimated for the Spanish (S) and German (G) markets are available at the web page "http://chiara.eco.unibs.it/~guastaro/Partitioning/InstancesPartitioning.html". The transition probabilities have been computed using Eq. (3). Recall that the total number of rows of a transition probability matrix of a 7^{th} order Markov chain with 12 states amounts to $\#\left(A^7\right) = [\#(A)]^7 = 12^7 = 35,831,808$. The transition probability matrices $\mathcal{M}_S$ and $\mathcal{M}_G$ have $1,964$ and $2,206$ rows, respectively, as they

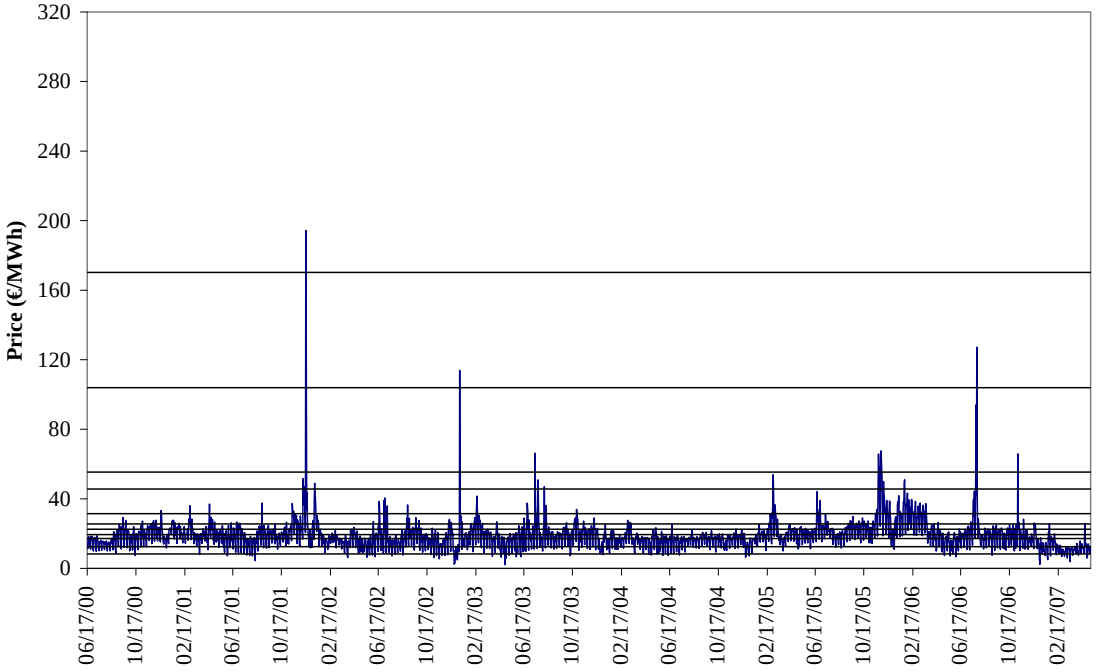

Figure 4: German daily electricity prices without exponential trend and with grid of initial states, or intervals.

include only the rows corresponding to observed 7-states. The missing rows are not reported, because they refer to non observed 7-states whose transition probabilities are null. Notice that in both cases the rows represent a small subset of A^7.

Another empirical feature of our evolutive phenomena is the presence of observed 7-states with deterministic evolution. The set of observed 7-states, A_7, is composed of $2,183$ 7-states for the Spanish instance and $2,510$ 7-states for the German market[2]. The percentage of deterministic 7-states over all the observed 7-states is 89% in the case of Spain and 86% in the case of Germany. The remaining observed 7-states will be called *probabilistic*, because, contrarily to the deterministic 7-states, they have empirically evolved towards more than one of the 12 possible states. In the end, the estimated 7^{th} order transition probability matrix $\mathcal{M}_S$ consists of 90 probabilistic rows and $1,874$ deterministic rows, while for $\mathcal{M}_G$ the numbers are 136 and $2,070$, respectively.

Let us notice that the generation of bootstrapped series should preserve the deterministic transitions of the original series: when a series has shown a nearly unique evolution (e.g., a spike), a bootstrapping procedure should keep the deterministic nature of such trajectory. On the other hand, the probabilistic k-states, denoting a recurrent feature of the phenomenon, represent the points where a bootstrapping methodology should select among the observed evolutions.

Our bootstrapping procedure is designed to take into account the two previous requirements. Specifically, the procedure allows deterministic k-states to be reproduced. On the other hand, by introducing the parameter $\gamma \in [0,1]$ in optimization problem (8)-(9), we control the number of classes aggregating the probabilistic k-states, therefore tuning the trade-off between statistical similarity and multiplicity of the bootstrapped series.

Given the fact that not all the potential k-states have been observed and that many observed k-states are deterministic, we have adopted the following settings for our optimization problem:

1. all the non observed k-states, whose transition probabilities are null, are considered as the members

[2]These two numbers exclude the last observed 7-states, because they have not evolved to a state.

of a unique class,

2. each observed deterministic k-state is the unique member of a one element class.

Given the fact that the number and the composition of the partition classes of points 1. and 2. are the same for all the feasible solutions, we have solved our optimization problem by calculating the distance indicator and the multiplicity measure *only on the partition classes aggregating the probabilistic 7-states*. As a result of the previous settings, all the feasible partitions $\boldsymbol{\lambda}$ are characterized by three kinds of classes:

a) the unique class collecting the non observed 7-states,

b) as many classes as the number of observed deterministic 7-states,

c) the classes aggregating the observed probabilistic 7-states.

4.5 Tabu Search settings

Because of the rapidly increasing number of partitions that have to be considered when solving optimization problem (8)-(9), the exact solution procedure proposed in Cerqueti et al. (2009) can handle only small size instances. Thus, the first goal of the computational experiments is to validate our heuristic comparing the performance of the Tabu Search algorithm with that of the exact solution procedure introduced in Cerqueti et al. (2009) for small size instances. The second goal is to assess the performance of the heuristic when solving large scale instances.
The Tabu Search algorithm has been coded in Java. The computational experiments have been performed on a PC with an Intel Xeon 2.27 GHz processor and 12.00 GB of RAM. To the first goal we have generated 10 instances, as described in Subsection 5.1, whereas to the second goal we have considered the two 7^{th} order transition probability matrices $\mathcal{M}_S$ and $\mathcal{M}_G$.
After extensive preliminary experiments, the parameters of the Tabu Search algorithm have been set to the following values. The Tabu Search algorithm stops after 2000 iterations (parameter *MaxIterations*). The jumping procedure is called after 500 consecutive iterations without any improvement (parameter *MaxIterNoImprov*). Additionally, once a unidimensional partition λ_w is inserted in a tabu list, it remains in the list $round(4 \times k \times U[0,1])$ iterations, where $U[0,1]$ is a uniformly distributed random number between 0 and 1, and $round(\cdot)$ indicates the nearest integer of "$\cdot$". Furthermore, we set the interval $[\eta, \omega]$ for the random choice of the number of jumps equal to $[2, 5]$. Finally, the interval $[0, 1]$ is divided into 40 subintervals (parameter n), each of length equal to 0.025. As a consequence, 39 consecutive executions of the Tabu Search algorithm are performed, one for each value of γ separating the subintervals.
Recall that, moving from the starting point (the all-comprehensive partition), the following points of the frontier are obtained performing the heuristic for each value of γ separating the subintervals. The initial solution considered at any execution is the solution obtained at the previous execution of the Tabu Search algorithm.

5 Results and Discussion

5.1 Computational results: the Tabu Search algorithm

To evaluate the effectiveness of the Tabu Search algorithm, we have generated two groups of instances, each including 5 test problems:

Group 1: the instances refer to 5 Markov chains of order 5 with 3 states. Each corresponding transition probability matrix includes 243 rows. The instances are laboratory tests of (relatively) small size. The number of partitions that can be formed for each instance is $3,125$;

Group 2: this group is similar to group 1, with the only difference that they refer to 5 Markov chains of order 3 with 5 states. Each corresponding matrix includes 125 rows. The number of partitions that can be formed for each instance is $140,608$.

All the previous instances have been generated similarly to the cases analyzed in Cerqueti et al. (2009).

Table 3: Comparison between the optimal efficient frontiers and the efficient frontier approximations calculated for 10 instances.

	Instance details				Exact solution procedure (average values)	Tabu Search algorithm (average values)[e]			
Group	No. of instances[a]	No. of states[b]	No. of time lags[c]	No. of partitions[d]	CPU (secs.)[i]	No. of found/ No. of points[f]	Gap (%)[g]	Gap_{95} (%)[h]	CPU (secs.)[i]
1	5	3	5	3,125	24.968	0.701	3.98%	0.00%	21.235
2	5	5	3	140,608	570.864	0.655	6.66%	0.75%	28.438

[a] Column "No. of instances" shows the number of instances composing each group.
[b,c] The numbers of states and time lags are reported in the following two columns.
[d] Column "No. of partitions" refers to the number of partitions that can formed for each instance of the group.
[e] The last 4 columns concern the average performance of the Tabu Search algorithm.
[f] Column "No. of found/No. of points" reports the average number of solutions found by the Tabu Search algorithm divided by the number of points composing the efficient frontiers.
[g] Column "Gap (%)" shows the average percentage error computed considering only the number of points found by the Tabu Search algorithm. For each point considered, the statistic is computed as $100 \times \frac{d_{\boldsymbol{\lambda}^H} - d_{\boldsymbol{\lambda}^*}}{d_{\boldsymbol{\lambda}^*}}$, where $d_{\boldsymbol{\lambda}^H}$ is the distance of the best-known solution found by the Tabu Search algorithm, and $d_{\boldsymbol{\lambda}^*}$ is the distance of the solution found by the exact solution procedure and having the same multiplicity of the heuristic solution ($m_{\boldsymbol{\lambda}^*} = m_{\boldsymbol{\lambda}^H}$).
[h] Column "Gap_{95} (%)" shows the average percentage error computed by considering only the points within the 95^{th} percentile.
[i] The average computing time spent by the exact solution procedure and Tabu Search algorithm computing the frontiers are reported in CPU (secs.). Computing times for the Tabu Search algorithm include the (negligible) time spent computing the value of ϵ for each time lag (see Subsection 3.1).

Table 3 summarizes the details and the computational results of the instances in groups 1 and 2. The first 4 columns show the key features of each group: the number of instances composing the group, the order and the states of the Markov chains, and the number of partitions. The following columns report the performances of the exact solution procedure (column 5) and of the Tabu Search algorithm (columns 6 to 9).

We can separate the comments on efficiency (CPU times) from those concerning effectiveness (quality of the solutions).

Starting with the CPU times, we notice that the computing time spent by the Tabu Search algorithm is neatly smaller than that needed by the exact solution procedure, especially for group 2 including "bigger" instances than those in group 1. The computing times show that the resources needed by the exact solution procedure increase sharply with the dimension of the instances and justify the use of our heuristic approach to solve large scale instances.

Concerning the quality of the solutions, the number of solutions found by the Tabu Search algorithm which correspond to those found by the exact solution procedure have ranged from 65.5% (group 2) to 70.1% (group 1). Statistic "Gap (%)" is equal to 3.98% and to 6.66% for groups 1 and 2, respectively. Nevertheless, it must be noticed that the statistic values have been largely driven by the errors made in few cases. For example, the largest error (260%) among the 5 instances in group 1 is due to an apparently small difference between the partitions found by the exact solution procedure and the Tabu Search algorithm, as shown in Table 4.

Table 4: Difference between partition $\boldsymbol{\lambda}^*$ found by the exact solution procedure and the corresponding partition, $\boldsymbol{\lambda}^{\mathbf{H}}$, found by the Tabu Search algorithm and generating the largest error (260%) among the 5 instances in group 1.

Time lag k	Exact solution procedure partition $\boldsymbol{\lambda}^* = (\lambda_5^*, ..., \lambda_1^*)$	Tabu Search algorithm partition $\boldsymbol{\lambda}^H = (\lambda_5^H, ..., \lambda_1^H)$
4	$\lambda_4^* = \{\{1,2,3\}\}$	$\lambda_4^H = \{\{1,2\},\{3\}\}$
3	$\lambda_3^* = \{\{1\},\{2,3\}\}$	$\lambda_3^H = \{\{1,2,3\}\}$

For time lags 5, 2, and 1, the partitions are identical.

This behavior suggests that even small changes in the composition of the solution can cause a dramatic deterioration of the objective function. Removing the outliers (i.e. the errors between the 95^{th} and the 99^{th} percentiles) from the computation, it can be noticed that the Tabu Search algorithm found the optimal or a near-optimal solution (see statistic "Gap_{95} (%)").

For the sake of brevity, we report the comparison between the optimal efficient frontier (Optimal) and the efficient frontier approximation (Heuristic) of one instance for each group (see Fig. 5 and 6). It is worth noting that the efficient frontier approximations replicate their optimal counterparts, with the exception of few cases where the algorithm has not been able to find the optimal solution.

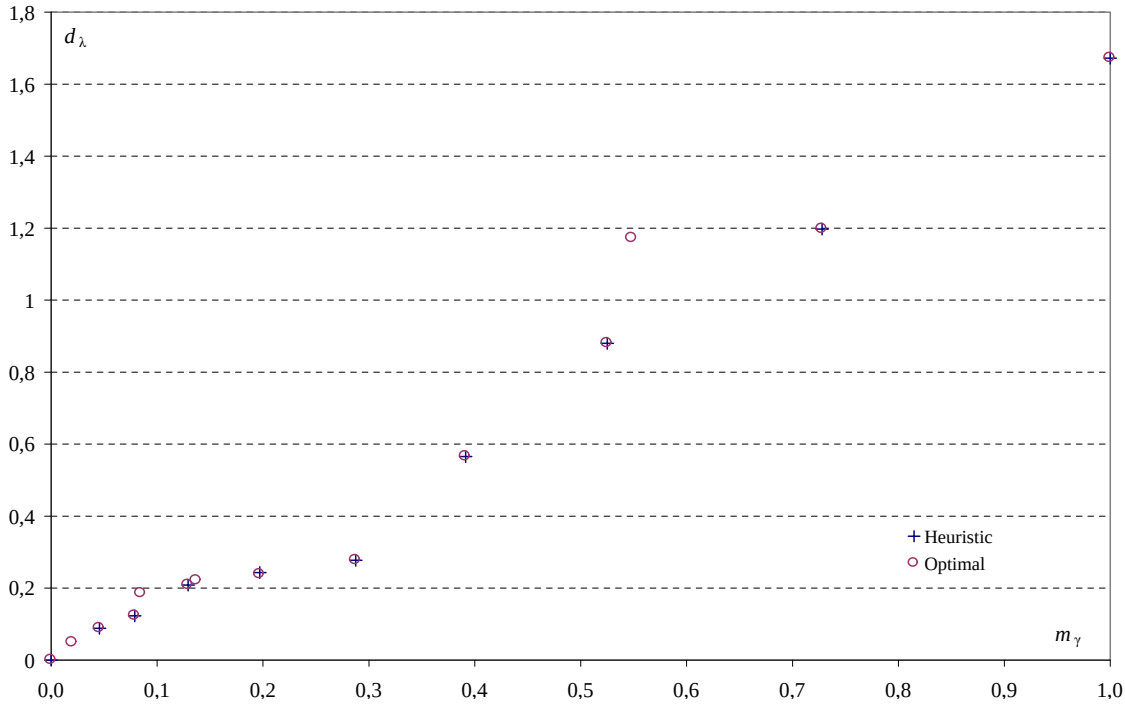

Figure 5: Comparison between the optimal efficient frontier and the efficient frontier approximation based on instance 1 of group 1. The frontiers are overlapping except 4 points (only circle), where the heuristic has not found the optimal solution.

The efficient frontier approximations computed by the Tabu Search algorithm for the matrices $\mathcal{M}_S$ and $\mathcal{M}_G$ are shown in Fig. 7. To the sake of completeness, we report in Appendix C the coordinates of the points computed by the Tabu Search algorithm. By construction we know the two partitions corresponding to the extreme points of the efficient frontier approximation, i.e. the singleton partition and the all-comprehensive partition (see Subsection 3.1.1). Therefore the efficient frontier approximations reported in Fig. 7 represent the 39 solutions found by the Tabu Search algorithm plus the two extreme partitions.

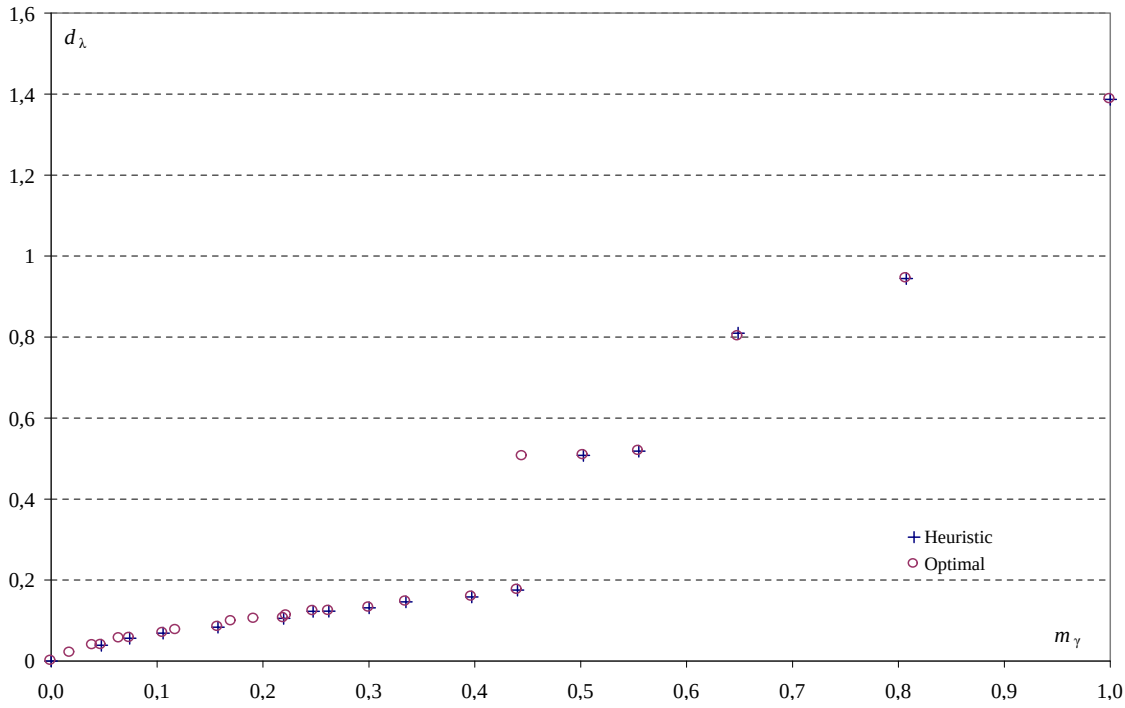

Figure 6: Comparison between the optimal efficient frontier and the efficient frontier approximation based on instance 4 of group 2. The frontiers are overlapping except 8 points (only circle), where the heuristic has not found the optimal solution.

Finally, the Tabu Search algorithm computed the efficient frontier approximation for the Spanish instance in 211.550 secs. (on average it took 5.424 secs. to compute each solution), whereas it required 619.669 secs. to compute the efficient frontier approximation for the German instance (on average 15.889 secs. to compute each solution).
In Fig. 7, three points catch the eye. These points represent the 4 partitions we have chosen to perform our bootstrap procedure: two partitions, characterized by the minimum of multiplicity and the maximum of similarity, to be compared with two other partitions, showing a sufficiently high level of multiplicity and an acceptable level of similarity.
The points at the origin of the axes represent the singleton partitions of Spain and Germany; we denote these partitions as $\boldsymbol{\lambda}_S$ and $\boldsymbol{\lambda}_G$, respectively. The other two points represent the heuristic solutions of optimization problem (8)-(9) with $\gamma = 0.4$ and $\gamma = 0.275$, respectively for Spain and Germany. We call these two partitions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$. The 7^{th} order transition probability matrices associated to $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$ are $\mathcal{M}_S^H$ and $\mathcal{M}_G^H$. The matrices report the probabilities of each partition class as estimated through Eq. (7) and are available at the web page "http://chiara.eco.unibs.it/˜guastaro/Partitioning/InstancesPartitioning.html". Partition $\boldsymbol{\lambda}_S^H$ consists of 45 classes aggregating the observed 90 probabilistic 7-states, $1,874$ classes of observed deterministic 7-states, and one further class containing all the non observed 7-states. Partition $\boldsymbol{\lambda}_G^H$ has 57 classes aggregating the 136 observed probabilistic 7-states, $2,070$ deterministic classes, and a class collecting the non observed 7-states.
Tables 5 and 6 detail partitions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$ in terms of unidimensional partitions $\lambda_{S,w}^H$ and $\lambda_{G,w}^H$, with $w = 1, \ldots, 7$ (see also Eq. (4)).

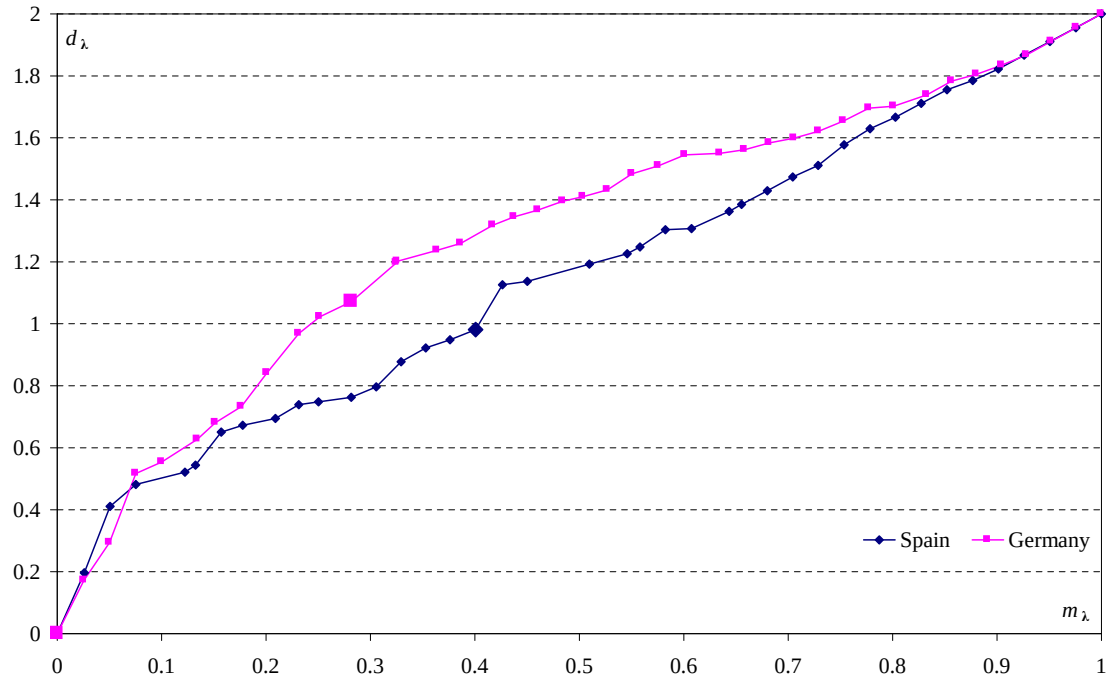

Figure 7: Efficient frontier approximations of Spain (diamond mark) and Germany (square mark) based on matrices $\mathcal{M}_S$ and $\mathcal{M}_G$. The bigger marks represent the two partitions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$ found by the Tabu Search algorithm and the two singleton partitions $\boldsymbol{\lambda}_S$ and $\boldsymbol{\lambda}_G$. These 4 partitions have been chosen to evaluate the performance of our methodology.

Table 5: The 7 unidimensional partitions composing the heuristic multidimensional partition $\boldsymbol{\lambda}_S^H$.

Time lag k	$\boldsymbol{\lambda}_S^H = (\lambda_{S,7}^H, ..., \lambda_{S,1}^H)$						
7	$\lambda_{S,7}^H = \{\{1,8,9\},$	$\{2\},$	$\{3,4,5,6,7\},$		$\{10\},$	$\{11\},$	$\{12\}\}$
6	$\lambda_{S,6}^H = \{\{1,3,4,5,6,7,9\},$	$\{2,8\},$			$\{10\},$	$\{11\},$	$\{12\}\}$
5	$\lambda_{S,5}^H = \{\{1,2,4,9\},$	$\{3,8\},$	$\{5,6,7\},$		$\{10\},$	$\{11\},$	$\{12\}\}$
4	$\lambda_{S,4}^H = \{\{1,4,5,6,9\},$	$\{2,3,7\},$	$\{8\},$		$\{10\},$	$\{11\},$	$\{12\}\}$
3	$\lambda_{S,3}^H = \{\{1,2\},$	$\{3\},$	$\{4\},$	$\{5,6,7,8,9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
2	$\lambda_{S,2}^H = \{\{1,2,8,9\},$	$\{3,4\},$	$\{5,6,7\},$		$\{10\},$	$\{11\},$	$\{12\}\}$
1	$\lambda_{S,1}^H = \{\{1,4,5,6,7,9\},$	$\{2,3,8\},$			$\{10\},$	$\{11\},$	$\{12\}\}$

Table 6: The 7 unidimensional partitions composing the heuristic multidimensional partition $\boldsymbol{\lambda}_G^H$.

Time lag k	$\boldsymbol{\lambda}_G^H = (\lambda_{G,7}^H, ..., \lambda_{G,1}^H)$							
7	$\lambda_{G,7}^H = \{\{1,2,3,4,5,8\},$	$\{6,7\},$			$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
6	$\lambda_{G,6}^H = \{\{1,6\},$	$\{2\},$	$\{3,4,5,7,8\},$		$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
5	$\lambda_{G,5}^H = \{\{1,6\},$	$\{2,7,8\},$	$\{3,4,5\},$		$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
4	$\lambda_{G,4}^H = \{\{1,6,7\},$	$\{2\},$	$\{3,4,8\},$	$\{5\},$	$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
3	$\lambda_{G,3}^H = \{\{1,6,8\},$	$\{2,3,4,5\},$	$\{7\},$		$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
2	$\lambda_{G,2}^H = \{\{1,5\},$	$\{2,3,4\},$	$\{6\},$	$\{7,8\},$	$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$
1	$\lambda_{G,1}^H = \{\{1,2,3,4\},$	$\{5,7\},$	$\{6,8\},$		$\{9\},$	$\{10\},$	$\{11\},$	$\{12\}\}$

We can make the following comments on the two tables:

- the "highest" states, namely states 10 to 12 for the Spanish instance and states 9 to 12 for Germany, are intervals of high prices and have been kept separated for all the time lags. This result depends on the fact that all the 7-states composed by at least one of these states are deterministic, preventing that such states could be aggregated at some time lags,

- the other states, namely states 1 to 9 for the Spanish market and states 1 to 8 for Germany, have been somehow aggregated at each time lag. Such aggregations indicate that at some time lags the preliminary segmentation of the state space has turned out to be fulsome to describe the phenomenon evolution,

- what is surprising about the clustering of states is the non hierarchical feature of the unidimensional partitions: indeed the aggregation of some states at a time lag does not maintain, or even become finer, at closer time lags. For example, time lag 1 of Spain is characterized by a coarser partition of states than the other time lags (except for time lag 6). Similar observations can be made for the German instance. This result contrasts with the hierarchical aggregation of states implied by the Variable Length Markov Chain bootstrap advanced in Bühlmann and Wyner (1999),

- finally, the "key information" about the phenomenon evolution seems to be brought by time lag 3 for Spain and time lags 2 and 4 in the German case: indeed, the unidimensional partitions of these time lags include the highest number of classes, meaning that keeping states separated at these time lags matters to describe the phenomenon evolution.

5.2 Computational results: the bootstrapped series

For each market (Spain and Germany), we have evaluated the performance of our bootstrap procedure in two different scenarios:

i. a "conservative" scenario, where we consider the two partitions $\boldsymbol{\lambda}_S$ and $\boldsymbol{\lambda}_G$, which are characterized by the minimum multiplicity and the maximum similarity among the bootstrapped series,

ii. a "progressive" scenario, where we consider the two (half-way) partitions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$, which are expected to generate higher diversification and lower similarity among the bootstrapped series than $\boldsymbol{\lambda}_S$ and $\boldsymbol{\lambda}_G$.

We therefore have generated 4 sets of $5,000$ bootstrapped series, one set for each partition, with length $lgh = 2,183$ for the Spanish cases and $lgh = 2,510$ for the German ones. The lengths of the bootstrapped series are equal to the lengths of the corresponding original series (net of the values required to initialize

the procedure) to allow for a fairer comparison[3]. Indeed, in the following we analyze the statistical properties of the bootstrapped series in order to compare them with the ones of the original series.
Before introducing the statistics, we want to give a taste of how bootstrapped series look like. Fig. 8 and 9 report two bootstrapped series, one for the Spanish market and one for the German instance. The two series are based on partitions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$. The bootstrapped series include the exponential trend initially removed from the original samples (see Subsection 4.2). Each value of the series is classified as deterministic (thin mark) or probabilistic (thick mark).

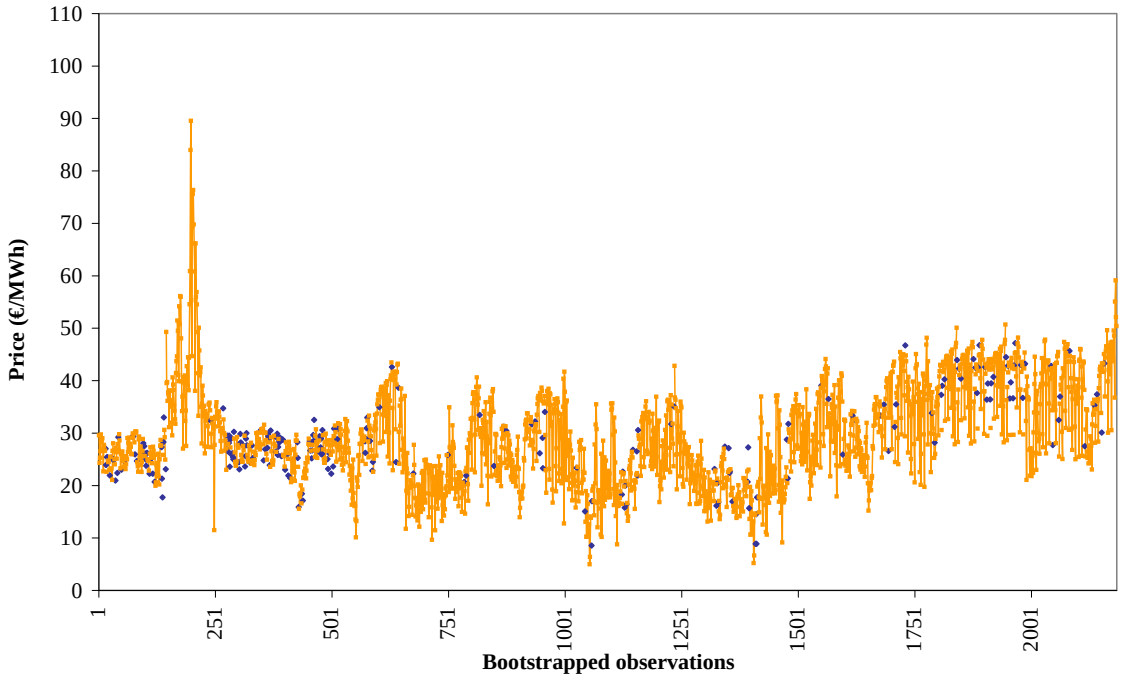

Figure 8: Bootstrapped Spanish electricity prices. The thick mark indicates that the selected value is the ending point of an observed probabilistic 7-state, the thin mark indicates that the bootstrapped values are the last points of observed deterministic 7-states.

We can make the following remarks:

- both the bootstrapped series in Fig. 8 and 9 reproduce the spikes observed in the original series (see Fig. 1 and 2),
- also the normal trading regime appears satisfactorily reproduced. Indeed the two series take values in a range strongly overlapping with that of their original counterparts,
- weekly seasonalities are clearly distinguishable, as well as a slightly positive trend,
- the frequencies of probabilistic values are 11% and 17%, respectively for the Spanish and German cases, similar to the values in the original series (i.e. 11% for Spain and 14% for Germany). The majority of times the bootstrap procedure reproduces deterministically sequences of the original series. Such segments are interleaved occasionally by probabilistic values, as we wanted to obtain. The higher percentage of probabilistic values in the German case may be due to the larger number of probabilistic classes of $\boldsymbol{\lambda}_G^H$ with respect to $\boldsymbol{\lambda}_S^H$ (57 against 45).

[3]See Subsection 3.2 for the initialization steps of the bootstrap procedure.

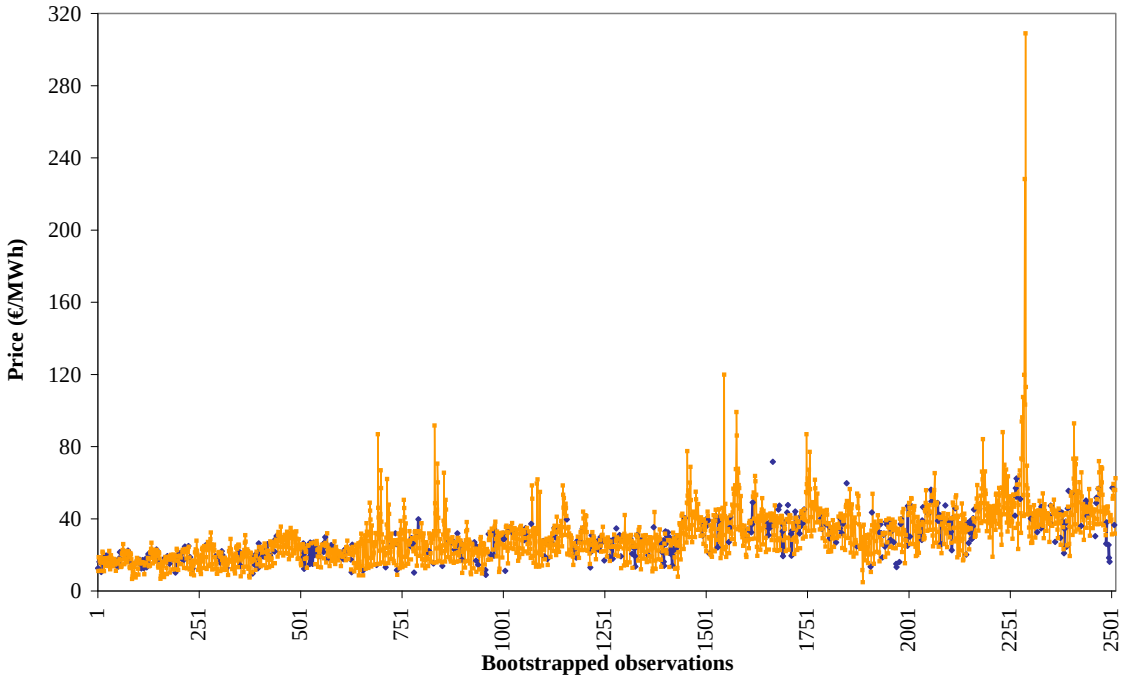

Figure 9: Bootstrapped German electricity prices. The thick mark indicates that the selected value is the ending point of an observed probabilistic 7-state, the thin mark indicates that the bootstrapped values are the last points of observed deterministic 7-states.

To evaluate more rigorously the statistical properties of the bootstrapped series with respect to their original counterparts, we have calculated the following statistics on each bootstrapped series:

1. average,
2. standard deviation,
3. skewness,
4. kurtosis,
5. minimum,
6. maximum,
7. autocorrelation at lag k, $k = 1, ..., 8$,
8. linear regression slope, b, with $\tilde{x}_t = a + bt + e_t$, $t = 1, ..., lgh$.

The statistics from 1. to 6. are concerned with the distribution of prices, while the statistics in 7. and 8. are more concerned with the dynamic structure of the series. The autocorrelation at lag 8 is observed to check if the similarity between the original and the bootstrapped series is kept beyond the order $k = 7$ used to define the driving process.
Tables 7 and 8 report (for Spain and Germany, respectively) the 5^{th} and 95^{th} percentiles of the distributions of the mentioned statistics, together with the actual value observed in the original series. To evaluate these distributions, we also report the percentile rank, i.e. the percentage of cases for which the statistic value is smaller than or equal to the original observed one.
We can make the following remarks:

- for all the four partitions $\boldsymbol{\lambda}_S$, $\boldsymbol{\lambda}_S^H$, $\boldsymbol{\lambda}_G$, and $\boldsymbol{\lambda}_G^H$, all the statistics computed for the original series take values in the middle percentiles. This is true also for the autocorrelation at lag 8,

- the percentile ranks are more fluctuating in the German scenarios than in the Spanish cases: the lowest percentage of Germany is 41 (see row "autocorrelation at lag 7" of partition $\boldsymbol{\lambda}_G^H$ in Table 8), against 57 for Spain (see row "average" of partition $\boldsymbol{\lambda}_S^H$ and row "kurtosis" of partition $\boldsymbol{\lambda}_S$ in Table 7). The highest percentages in the Spain and Germany cases are, respectively, 74 and 77, and display a more negligible difference,

- nearly all the 5-95 intervals of Germany related to partition $\boldsymbol{\lambda}_G^H$ (the half-way scenario) show wider limits than the corresponding intervals calculated on the distributions of $\boldsymbol{\lambda}_G$ (the conservative scenario). Such result denotes that in the German case passing from the conservative scenario to a progressive one allows for an increase of the multiplicity among the bootstrapped series. The case of Spain, on the contrary, is much more stable: there seems to be no remarkable difference between the 5-95 intervals generated with the two partitions $\boldsymbol{\lambda}_S$ and $\boldsymbol{\lambda}_S^H$. This implies that the bootstrapped series obtained with the two partitions are rather stable. Again, the explanation of this result may be attributed to the higher number of probabilistic 7-states observed in the German case,

- although we do not give evidence of the complete distribution of the maximum, we report that not all the $5,000$ series generated in each setting have shown a spike. This feature reflects the desirable property that a rare event does not have to appear regularly.

Table 7: Percentiles of some statistics distributions computed out of the $5,000$ bootstrapped series of Spain for two scenarios: the conservative scenario (partition $\boldsymbol{\lambda}_S$) and the progressive scenario (partition $\boldsymbol{\lambda}_S^H$).

Statistics	Spain - $\boldsymbol{\lambda}_S$				Spain - $\boldsymbol{\lambda}_S^H$			
	5^{th} percentile	95^{th} percentile	Value of original series	Percentile rank of original series value	5^{th} percentile	95^{th} percentile	Value of original series	Percentile rank of original series value
average	26.522	31.889	29.692	62	26.574	32.074	29.692	57
standard dev.	6.844	10.906	9.57	69	7.113	10.979	9.57	68
skewness	0.091	2.108	1.381	60	0.114	2.033	1.381	66
kurtosis	−0.518	10.205	5.081	57	−0.571	9.327	5.081	63
minimum	4.563	11.579	5.469	59	4.546	11.754	5.469	58
maximum	50.237	111.33	103.758	70	50.971	111.382	103.758	71
aut. at lag 1	0.737	0.861	0.818	60	0.737	0.859	0.818	62
aut. at lag 2	0.579	0.773	0.706	59	0.579	0.772	0.706	61
aut. at lag 3	0.545	0.746	0.706	62	0.547	0.745	0.706	63
aut. at lag 4	0.526	0.732	0.667	62	0.529	0.733	0.667	63
aut. at lag 5	0.513	0.729	0.661	62	0.52	0.73	0.661	62
aut. at lag 6	0.604	0.762	0.721	68	0.614	0.764	0.721	65
aut. at lag 7	0.704	0.825	0.802	74	0.728	0.829	0.802	68
aut. at lag 8	0.565	0.724	0.683	69	0.581	0.727	0.683	64
lin. regr. slope	−0.001	0.007	0.004	64	−0.001	0.007	0.004	63

Table 8: Percentiles of some statistics distributions computed out of the 5,000 bootstrapped series of Germany for two scenarios: the conservative scenario (partition $\boldsymbol{\lambda}_G$) and the progressive scenario (partition $\boldsymbol{\lambda}_G^H$).

Statistics	Germany - $\boldsymbol{\lambda}_G$				Germany - $\boldsymbol{\lambda}_G^H$			
	5^{th} percentile	95^{th} percentile	Value of original series	Percentile rank of original series value	5^{th} percentile	95^{th} percentile	Value of original series	Percentile rank of original series value
average	29.237	34.775	32.261	65	29.224	34.63	32.261	67
standard dev.	13.051	22.806	18.4	73	12.684	22.949	18.4	77
skewness	1.294	7.966	3.974	67	1.09	8.324	3.974	70
kurtosis	3.879	119.625	35.573	60	2.696	129.849	35.573	64
minimum	2.122	4.941	3.117	69	2.122	5.261	3.117	65
maximum	137.416	457.285	301.542	75	124.443	457.462	301.542	76
aut. at lag 1	0.597	0.798	0.716	63	0.607	0.808	0.716	55
aut. at lag 2	0.387	0.661	0.572	66	0.386	0.67	0.572	61
aut. at lag 3	0.359	0.597	0.506	69	0.364	0.609	0.506	63
aut. at lag 4	0.32	0.584	0.476	65	0.329	0.598	0.476	59
aut. at lag 5	0.326	0.611	0.486	60	0.336	0.626	0.486	53
aut. at lag 6	0.381	0.712	0.585	55	0.376	0.724	0.585	48
aut. at lag 7	0.422	0.793	0.644	49	0.417	0.807	0.644	41
aut. at lag 8	0.319	0.687	0.544	52	0.313	0.7	0.544	45
lin. regr. slope	0.009	0.017	0.013	58	0.009	0.017	0.013	59

Given these results, it can be said that the original series (both the Spanish and the German one) can be considered as a trajectory sampled from the same process which has generated the bootstrapped series.

6 Conclusions

In this paper, we develop a method to apply a Markov chain bootstrapping for discrete time continuous-valued processes. The search of the "optimal" transition probability matrix has been solved through an application of a Tabu Search algorithm. This heuristic has shown satisfactory performances where the optimal solution of a test problem was known. At the same time it has allowed to solve a real life case, i.e. the bootstrapping of the price series of the German and Spanish electricity markets, finding the (sub-optimal) efficient set of the solutions in a time ranging from 0.5 to 1.5 hours running on a PC with an Intel Pentium 4 3.00 GHz processor and 2.00 GB of RAM. The heuristic solution of the small problems considered in Cerqueti et al. (2009) took at most 4.5 minutes of computing time, providing a slightly sub-optimal solution.

An even more relevant observation is concerned with the quality of the bootstrapped series. Standard statistics like the average, the standard deviation, the autocorrelation coefficients, and the trend observed on the resulting series have distributed nicely around the corresponding value of the original series. Such consistency cannot be arguably expected for alternative methods of bootstrapping, with particular reference to parametric methods, and block resampling methods. In the former case, the difficulties of assuming and estimating the parameters for a satisfactory model for the electricity prices can be overwhelming. Similarly for the latter case are the difficulties required to transform the original data into a stationary process. With respect to other Markov chain bootstrap methods, our proposal cannot be fully compared, since the main focus of this approach is to "find out" the relevant discretization of a continuous state space, while in the standard Markov chain bootstrap methods a finite number of relevant states is taken as given.

Ranging from lower to higher values of the multiplicity boundary has proven to impact on the quality of the bootstrapped series. The dispersion of the distributions of the statistics observed on the resulting series tends to increase as such constraint becomes more restricting, as it was expected. At the same time, the same distributions maintain a remarkable centrality with respect to the corresponding parameter values of the original series.
It is finally worth to mention that even more compelling cases can be approached through the method advanced here. Bootstrapping of multivariate Markov processes has been rarely faced in the literature, because of the significant additional difficulties of such processes. The components of a multivariate Markov process can be arbitrarily different and yet show significant dependencies. In such cases the complications arising for the block or the parametric bootstrapping can become simply unbearable. On the contrary, these processes can be easily modeled through a multivariate discretization of their state spaces in a natural extension of the single valued case developed here, and the design of the optimal transition probability matrix reduced to a problem of acceptable computational effort.

Acknowledgements

The authors would like to express their appreciation for the insightful comments made by two anonymous reviewers.

References

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

Anily, S. and Federgruen, A.: 1991, Structured partitioning problems, *Operations Research* **39**(1), 130–149.

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.: 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.

Cerqueti, R., Falbo, P. and Pelizzari, C.: 2009, Optimal dimension of transition probability matrices for markov chain bootstrapping, *Quaderni del Dipartimento di Istituzioni Economiche e Finanziarie dell'Università degli Studi di Macerata 53*, Università degli Studi di Macerata, Macerata, Italy, http://webhouse.unimc.it/economia/repo/quaderni/QDief53-2009.pdf.

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.

Courtois, P. J.: 1977, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, NY, USA.

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

Efron, B. and Tibshirani, R. J.: 1993, *An Introduction to the Bootstrap*, Chapman & Hall, New York, NY, USA.

Ehrgott, M., Klamroth, K. and Schwehm, C.: 2004, An mcdm approach to portfolio optimization, *European Journal of Operational Research* **155**(3), 752770.

Forgy, E. W.: 1965, Cluster analysis of multivariate data: Efficiency vs. interpretability of classification - abstract, *Biometrics* **21**(3), 768–769.

Freedman, D.: 1984, On bootstrapping two-stage least-squares estimates in stationary linear models, *The Annals of Statistics* **12**(3), 827–842.

Freedman, D. A. and Peters, S. C.: 1984, Bootstrapping a regression equation: Some empirical results, *Journal of the American Statistical Association* **79**(385), 97–106.

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.

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

Ma, P. C. H., Chan, K. C. C., Yao, X. and Chiu, D. K. Y.: 2006, An evolutionary clustering algorithm for gene expression microarray data analysis, *IEEE Transactions on Evolutionary Computation* **10**(3), 296–314.

MacQueen, J. B.: 1967, Some methods for classification and analysis of multivariate observations, *in* L. M. LeCam and J. Neyman (eds), *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, University of California Press, Berkeley, CA, USA, pp. 281–297.

Spears, W. M.: 1998, A compression algorithm for probability transition matrices, *SIAM Journal on Matrix Analysis and Applications* **20**(1), 60–77.

Sung, C. S. and Jin, H. W.: 2000, A tabu-search-based heuristic for clustering, *Pattern Recognition* **33**(5), 849–858.

Takahashi, Y.: 1984, Weak d-markov chain and its application to a queuing network, *in* G. Iazeolla, P.-J. Courtois and A. Hordijk (eds), *Mathematical Computer Performance and Reliability*, North-Holland Publishing Co., Amsterdam, The Netherlands, pp. 153–165.

Trejos, J., Murillo, A. and Piza, E.: 2004, Clustering by ant colony optimization, *in* D. Banks, L. House, F. R. McMorris, P. Arabie and W. Gaul (eds), *Classification, Clustering, and Data Mining Applications - Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15-18 July 2004*, Springer, Berlin, Germany, pp. 25–32.

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 Trck, S.: 2004, Modeling electricity prices: Jump diffusion and regime switching, *Physica A: Statistical Mechanics and its Applications* **336**(12), 39–48.

Woodside-Oriakhi, M., Lucas, C. and Beasley, J. E.: 2011, Heuristic algorithms for the cardinality constrained efficient frontier, *European Journal of Operational Research* **213**(3), 538550.

Wu, Z. and Leahy, R.: 1993, An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(11), 1101–1113.

Zhu, J., Hong, J. and Hughes, J. G.: 2002, Using markov chains for link prediction in adaptive web sites, *in* D. W. Bustard, W. Liu and R. Sterritt (eds), *Soft-Ware 2002: Computing in an Imperfect World. Lecture Notes in Computer Science, Volume 2311*, Springer, Berlin, Germany, pp. 60–73.

Appendix A

Table 9 summarizes some notation introduced in the paper.

Table 9: Some notation with a brief explanation.

Notation	Brief explanation
$[\alpha, \beta] \subseteq \mathbb{R}$	Real interval representing the state space of the continuous-valued process modelling the evolutive phenomenon
$\#(Y)$	Cardinality of set Y
$O = (x_1, \ldots, x_M)$	Original sample of the evolutive phenomenon
$A = \{a_1, \ldots, a_N\}$	N initial states, or intervals, $a_1, \ldots, a_N$, partitioning $[\alpha, \beta]$
$\{X(t); t \geq 0\}$	Markov chain of order k with state space A
$\mathbf{x}_{l,k} = (x_l, \ldots, x_{l+k-1})$	k-state, composed of observed values in O starting at time l
O_k	Set of k-states $\mathbf{x}_{l,k}$ included in O
$\tilde{\mathbf{a}}_{l,k} = (a_l, \ldots, a_{l+k-1})$	k-state, composed of observed intervals in A starting at time l. The component a_t is an element of A assigned to time t, with $t = l, ..., l + k - 1$
A_k	Set of k-states $\tilde{\mathbf{a}}_{l,k}$. Set A_k biunivocally corresponds to O_k
$\mathbf{a}_{h,k} = (a_{h_k}, a_{h_{k-1}}, ..., a_{h_1})$	Possible k-state of intervals in A without any reference to the starting time. The index $h_w = 1, ..., N$ points to an element of A for time lag w of k-state h
$F_{a_z \mid \mathbf{a}_{h,k}} = \{\tilde{\mathbf{a}}_{l,k} : \tilde{\mathbf{a}}_{l,k} \equiv \mathbf{a}_{h,k}, a_{l+k} \equiv a_z\}$	Set of observed k-states whose sequence of intervals of A is the same as the sequence of intervals of the possible k-state $\mathbf{a}_{h,k}$, and evolving to state $a_z \in A$ at time $l + k$
A^k	Set of k-states $\mathbf{a}_{h,k}$. The set is defined as the Cartesian product of A with itself k times
$\lambda = \{A_{\lambda,1}, \ldots, A_{\lambda,\#(\lambda)}\}$	Partition of nonempty subsets of A
Λ	Set of partitions λ of A
$\boldsymbol{\lambda} = (\lambda_k, \lambda_{k-1}, \ldots, \lambda_1)$	Multidimensional partition of A^k. Partition λ_w, $w = 1, \ldots, k$, is the unidimensional partition of A referred to time lag w
Λ^k	Set of multidimensional partitions $\boldsymbol{\lambda}$ of A^k
$\mathbf{A}_{\boldsymbol{\lambda},h,k} = (A_{\boldsymbol{\lambda}_k,h_k}, A_{\boldsymbol{\lambda}_{k-1},h_{k-1}}, ..., A_{\boldsymbol{\lambda}_1,h_1})$	h-th class of multidimensional partition $\boldsymbol{\lambda}$. The class is expressed as an ordered sequence of classes of partitions $\lambda_k, \lambda_{k-1}, \ldots, \lambda_1$
$\{X^*(t); t \geq 0\}$	Markov chain of order k^* with multidimensional state space A^{k^*} partitioned by $\boldsymbol{\lambda}^*$. Optimization problem (8)-(9) is introduced to choose k^* and $\boldsymbol{\lambda}^*$
$\tilde{O} = (\tilde{x}_1, \ldots, \tilde{x}_{lgh})$	Series of lgh observations bootstrapped with Algorithm 3

Appendix B

Table 10 reports the 12 initial states partitioning the state space of the detrended series of Spain and Germany. Each interval is identified with the label used in the paper. In both cases, the upper limit of the twelfth interval, i.e. β, represents a high enough value such that no price can be reasonably thought

to be greater than it. For example, in our experiment we could take $\beta = 1,000,000$.

Table 10: Initial states, or intervals, partitioning the state space of the detrended series of electricity prices of Spain and Germany.

Interval label	Interval of prices	
	Spain	Germany
1	$[0, 13.21)$	$[0, 8.26)$
2	$[13.21, 16.53)$	$[8.26, 12.43)$
3	$[16.53, 20.22)$	$[12.43, 17.18)$
4	$[20.22, 22.21)$	$[17.18, 19.41)$
5	$[22.21, 24.79)$	$[19.41, 22.45)$
6	$[24.79, 28.63)$	$[22.45, 25.58)$
7	$[28.63, 32.28)$	$[25.58, 31.49)$
8	$[32.28, 36.04)$	$[31.49, 45.62)$
9	$[36.04, 40.81)$	$[45.62, 55.40)$
10	$[40.81, 51.74)$	$[55.40, 103.98)$
11	$[51.74, 61.99)$	$[103.98, 170.17)$
12	$[61.99, \beta]$	$[170.17, \beta]$

Appendix C

Table 11 reports the coordinates of the 39 points computed by the Tabu Search algorithm. Notice that for the Spanish instance it happened that in two consecutive executions the Tabu Search algorithm found the same solution (see $\gamma = 0.475$ and $\gamma = 0.5$). The underlined coordinates represent the two solutions $\boldsymbol{\lambda}_S^H$ and $\boldsymbol{\lambda}_G^H$ chosen for our analysis.

Table 11: The points composing the two efficient frontier approximations of Spain and Germany. For each value of γ, we report the coordinates of the points found by the Tabu Search algorithm, i.e. the multiplicity measure $m_{\boldsymbol{\lambda}^H}$ and the distance indicator $d_{\boldsymbol{\lambda}^H}$.

γ	Spain		Germany	
	$m_{\boldsymbol{\lambda}^H}$	$d_{\boldsymbol{\lambda}^H}$	$m_{\boldsymbol{\lambda}^H}$	$d_{\boldsymbol{\lambda}^H}$
0.025	0.026	0.196	0.025	0.172
0.05	0.051	0.411	0.05	0.294
0.075	0.075	0.481	0.075	0.517
0.1	0.122	0.521	0.1	0.554
0.125	0.132	0.544	0.134	0.627
0.15	0.157	0.65	0.151	0.681
0.175	0.178	0.673	0.176	0.733
0.2	0.209	0.695	0.201	0.841
0.225	0.231	0.739	0.231	0.969
0.25	0.25	0.748	0.252	1.023
0.275	0.282	0.763	0.282	1.072
0.3	0.306	0.796	0.325	1.197
0.325	0.329	0.878	0.326	1.202
0.35	0.353	0.922	0.364	1.237
0.375	0.376	0.948	0.386	1.259
0.4	0.401	0.981	0.417	1.318
0.425	0.426	1.126	0.438	1.345
0.45	0.45	1.137	0.46	1.367
0.475	0.51	1.193	0.484	1.396
0.5	0.51	1.193	0.504	1.411
0.525	0.546	1.226	0.527	1.433
0.55	0.558	1.248	0.551	1.484
0.575	0.582	1.304	0.576	1.51
0.6	0.607	1.307	0.601	1.545
0.625	0.644	1.363	0.635	1.55
0.65	0.655	1.385	0.658	1.562
0.675	0.68	1.43	0.682	1.584
0.7	0.704	1.474	0.706	1.599
0.725	0.729	1.511	0.729	1.621
0.75	0.754	1.578	0.753	1.654
0.775	0.778	1.63	0.777	1.696
0.8	0.803	1.667	0.801	1.702
0.825	0.827	1.711	0.833	1.739
0.85	0.852	1.756	0.857	1.783
0.875	0.877	1.785	0.88	1.805
0.9	0.901	1.822	0.905	1.835
0.925	0.926	1.867	0.928	1.868
0.95	0.951	1.911	0.952	1.912
0.975	0.975	1.956	0.976	1.956