Is it Possible to Sequence Continuously Changing Workload Patterns for Periodic Elastic Scaling?

Ravi (Ravinder) Prakash G.
Senior Professor Research, BMS Institute of Technology
Dodaballapur Road, Avalahalli, Yelahanka
Bangalore – 560 064

Akshay Reddy
Undergraduate Student (2015)
Department of Computer Science Engineering
SDM College of Engineering and Technology
Dhavalagiri, Dharwad 580 002

ABSTRACT
Measurability is a concept in periodic Elastic scaling based on the following two conditions: (a) a cloud service provider should be cautious, that is, should not exclude any cloud consumer’s resource pooling pattern strategy from consideration; and (b) a cloud service provider should consider the cloud consumers’ resource pooling pattern preferences, that is, should deem a cloud consumer’s resource pooling pattern strategy \( k' \) infinitely more likely than \( k \), if it premises the cloud consumer to prefer \( k \) to \( k' \). A resource pooling pattern strategy is measurable if it can optimally be chosen under common resource pooling pattern conjecture in the events (a) and (b). In this paper we present an algorithm that for every finite periodic Elastic Scaling operation computes the set of all measurable resource pooling pattern strategies. The algorithm is based on the new idea of an Continuously Changing Workload preference limitation, which is a pair \((k_i, V_i)\) consisting of a resource pooling pattern strategy \( k_i \), and a subset of resource pooling pattern strategies \( V_i \) for cloud service provider \( i \). The interpretation is that cloud service provider \( i \) prefers some resource pooling pattern strategy in \( V_i \) to \( k_i \). The algorithm proceeds by successively adding Continuously Changing Workload preference limitations to the periodic Elastic Scaling.

Keywords

1. INTRODUCTION
In an periodic Elastic Scaling, it is natural to assume that a cloud service provider reasons about its cloud consumers before making a decision. Namely, in order to evaluate the possible consequences of a decision, the cloud service provider must form some resource pooling pattern conjecture about its cloud consumers’ choices which, in turn, must be based on some resource pooling pattern conjecture about its cloud consumers’ conjecture about their cloud consumers’ choices, and so on. It is the goal of periodic Elastic Scaling \([1] [28] [29]\) to formally describe such reasoning processes, and to investigate their behavioral implications. Throughout this paper we take a cloud service provider set perspective to analyze periodic Elastic Scaling-theoretic situations. That is, we always view the periodic Elastic Scaling from the perspective of cloud service provider set, and put restrictions only on the conjecture of this particular cloud service provider set – including conjecture about the cloud consumers’ conjecture – without imposing restrictions on the actual conjecture of the cloud consumers. We premise this approach to be plausible; as we cannot look inside the cloud consumers at the time we make an Continuously Changing Workload choice. So, can only base Continuously Changing Workload choice on conjecture about the cloud consumers, and not on the actual conjecture and Continuously Changing Workload choices of cloud consumers. But then, if we want to analyze the reasonable Continuously Changing Workload choices a cloud service provider can make in an periodic Elastic Scaling, it is sufficient to concentrate only on the conjecture of this particular cloud service provider set, as they encompass everything that can be used to make a decision. Although we premise the cloud service provider set perspective to be very natural, it crucially differs from the usual approach to periodic Elastic Scaling in papers and articles, which typically proceed by imposing restrictions on the conjecture of all cloud service provider set, and not only cloud service provider set.

Measurability is a concept within periodic Elastic Scaling that is based upon the following two assumptions:

- A cloud service provider should be cautious, that is, a cloud service provider should not exclude any cloud consumer’s resource pooling pattern strategy from consideration;
- A cloud service provider should consider the cloud consumers’ resource pooling pattern preferences, that is, if the cloud service provider premises that an cloud consumer prefers resource pooling pattern strategy \( k \) to resource pooling pattern strategy \( k' \), then the cloud service provider should deem \( k \) much more likely than \( k' \).

Any resource pooling pattern strategy that can be chosen optimally under common resource pooling pattern conjecture in these two events is called measurable.

In order to define measurability formally we can no longer model the cloud service providers’ conjecture by standard probability distributions. Suppose, for instance, that cloud service provider 1 premises that cloud service provider 2 prefers resource pooling pattern strategy \( a \) to resource pooling pattern strategy \( b \). If cloud service provider 1’s resource pooling pattern conjecture about 2’s choice would be modeled by a single probability distribution then cloud service provider
I should assign probability 0 to \( b \), since it must consider \( 2 \)'s resource pooling pattern preferences. This, however, would contradict the assumption that it is cautious.

A possible way to define measurability is by means of sequences of probability distributions, or by using totally ordered data-intensive systems [9], [10], [12], [14], [11], [24]. Both frameworks can model a state of mind in which you deem some cloud consumer’s resource pooling pattern strategy \( k_i \) infinitely more likely than some other resource pooling pattern strategy \( k_j \), without completely discarding the latter choice.

The practical disadvantage of these richer frameworks is that, it makes the computation of measurable resource pooling pattern strategies rather difficult. This is probably also the reason that measurability, despite its strong intuitive appeal, has not received as much attention as many other concepts in periodic Elastic Scaling. It would therefore be very useful to have an algorithm helping us to compute this measurable resource pooling pattern strategies. A procedure, called iteratively trembling, that for any given \( \alpha > 0 \) yields the set of \( \alpha \)-measurable resource pooling pattern strategies. By letting \( \alpha \) tend to zero, we finally would obtain the set of measurable resource pooling pattern strategies. So, in a sense, this procedure only indirectly leads to the set of measurable resource pooling pattern strategies, as we first have to apply the procedure for a sequence of small \( \alpha \)'s, and then let \( \alpha \) go to zero.

There is another algorithm designed for measurability, called iterated backward inference. This procedure does not exactly yield the set of measurable resource pooling pattern strategies, as its output may contain resource pooling pattern strategies that are not measurable. The output, however, always includes the set of measurable resource pooling pattern strategies.

In this paper we present an algorithm, called iterated addition of Continuously Changing Workload preference limitations that directly delivers the set of all measurable resource pooling pattern strategies in every finite periodic Elastic Scaling operation [17], [19], [23]. The algorithm is based on the new notion of an Continuously Changing Workload preference limitation. Formally, an Continuously Changing Workload preference limitation for cloud service provider \( i \) is a pair \((k_i, V_i)\), where \( k_i \) is a resource pooling pattern strategy and \( V_i \) a subset of resource pooling pattern strategies for cloud service provider \( i \). The interpretation is that cloud service provider \( i \) prefers some resource pooling pattern strategy \( V_i \) in to \( k_i \), without specifying which one (unless \( V_i \) contains only one resource pooling pattern strategy, of course). A totally ordered resource pooling pattern conjecture for cloud service provider \( i \) about its cloud consumers’ resource pooling pattern strategies is a finite sequence \( \psi_i = (\psi_i^1, \ldots, \psi_i^p) \) of probability distributions on \( K_i \), the set of cloud consumers’ resource pooling pattern strategy combinations, such that every resource pooling pattern strategy combination \( k_i \) in \( K_i \) receives positive probability under some probability distribution \( \psi_i^p \) in this sequence. For every \( p \in \{1, \ldots, P\} \), we call \( \psi_i^p \) the level \( p \) resource pooling pattern conjecture.

The totally ordered resource pooling pattern conjecture \( \psi_i \) deems some resource pooling pattern strategy combination \( k_i \) \( \psi_i \) infinitely more likely than some other resource pooling pattern strategy combination \( k_i' \), if there is some level \( p \) such that \( k_i \) receives positive probability under the level \( p \) resource pooling pattern conjecture \( \psi_i^p \), whereas \( k_i' \) receives probability zero under the first \( p \) levels. We say that \( \psi_i \) consider an Continuously Changing Workload preference limitation \((k_i, V_i)\), for cloud consumer \( j \) if it deems some resource pooling pattern strategy in \( V_i \) infinitely more likely than \( k_i \). This thus mimics the condition in measurability that \( i \) must consider \( j \)'s resource pooling pattern preferences. The totally ordered resource pooling pattern conjecture \( \psi_i \) is said to assume a subset \( E_i \subseteq K_i \), of resource pooling pattern strategy combinations if it deems every element in \( E_i \), infinitely more likely than every element outside \( E_i \).

The algorithm we present proceeds by inductively adding Continuously Changing Workload preference limitations [6], [2], [4], [3], [8], [5], [26] until no further Continuously Changing Workload preference limitations can be produced. At round 1, we start with the empty set of Continuously Changing Workload preference limitations for all cloud service providers. In every subsequent round, we add an Continuously Changing Workload preference limitation \((k_i, V_i)\) for cloud service provider \( i \) if every totally ordered resource pooling pattern conjecture \( \psi \) on \( K_i \), that consider all current Continuously Changing Workload preference limitations for \( i \)'s cloud consumers, assumes some subset \( E_i \subseteq K_i \), on which \( k_i \) is weakly dominated by some randomized resource pooling pattern strategy on \( V_i \). We continue this process until no further Continuously Changing Workload preference limitation can be added. Among the final set of Continuously Changing Workload preference limitations for cloud service provider \( i \), we look for those resource pooling pattern strategies \( k_i \) that are not part of any Continuously Changing Workload preference limitation \((k_i, V_i)\). We show that these resource pooling pattern strategies are exactly the measurable resource pooling pattern strategies for cloud service provider \( i \).

So, at every round the algorithm produces, for each cloud service provider, a set of Continuously Changing Workload preference limitations. As the set of Continuously Changing Workload preference limitations can only grow at every round, and there are only finitely many possible Continuously Changing Workload preference limitations, the algorithm must stop after finitely many rounds.

Not only can this algorithm be used to compute the measurable resource pooling pattern strategies in an periodic Elastic Scaling, it also represents a natural inductive reasoning procedure for the cloud service providers that eventually lead them to measurable resource pooling pattern choices. The central object in this reasoning process is that of an Continuously Changing Workload preference limitation. If we add an Continuously Changing Workload preference limitation \((k_i, V_i)\) for cloud service provider \( i \), then normally this means that \( i \)'s cloud consumers premises that \( i \) prefers some resource pooling pattern strategy in \( V_i \) to \( k_i \). Moreover, if \( i \)'s cloud consumers consider \( i \)'s resource pooling pattern preferences, as we assume in measurability, then \( i \)'s cloud consumers will also deem some resource pooling pattern strategy in \( V_i \) infinitely more likely than \( k_i \). Thus, by adding Continuously Changing Workload preference limitations at every round, we further and further limit the possible totally ordered conjecture that cloud service providers can plausibly hold about their cloud consumers’ choices. In a sense, what the algorithm shows is that, in order to reason your way toward measurable resource pooling pattern strategies, it is sufficient to keep track of the cloud service providers’ Continuously Changing Workload preference limitations. At every round, by considering the current Continuously Changing Workload preference limitations, we can possibly derive new Continuously Changing Workload preference limitations, thus further limitations the cloud service
providers’ possible totally ordered conjecture, until this reasoning process cannot produce any new Continuously Changing Workload preference limitations. This is where the reasoning procedure ends, and by looking at the final Continuously Changing Workload preference limitations we can find the entire measurable resource pooling pattern strategies in the periodic Elastic Scaling.

In the algorithm we present, the objects of output are different than in previous procedure. There, the procedure delivers at every round and for every cloud service provider $i$, a set of full support probability distributions on cloud service provider $i$’s resource pooling pattern strategy, where this set becomes smaller with every round. As there are infinitely many possible sets of full support probability distributions, previous procedure can produce infinitely many possible outputs in every round. This is a major difference with the algorithm we propose, where at every round there are only a finite number of possible outputs, namely the Continuously Changing Workload preference limitations at that round. Note also that the algorithm in this paper is fundamentally different from most other inductive concepts in periodic Elastic Scaling, which usually proceed by successively eliminating resource pooling pattern strategies from the periodic Elastic Scaling. Think, for instance, of iterated elimination of strictly (weakly) dominated resource pooling pattern strategies. So, why did we not base the algorithm on elimination of resource pooling pattern strategies as well? The reason is that iterated elimination of strategies cannot work for measurability. In Section 2 we provide an algorithm for measurability must necessarily be of a different nature than the ones we are used to.

The outline of the paper is as follows. In Section 2 we show, why successive elimination of resource pooling pattern strategies does not work for measurability. In Section 3 we give a formal necessary and sufficient condition of measurability, by making use of totally ordered Data-intensive systems [9], [10], [12], [14] [11]. In Section 4 we present the algorithm, illustrate it by means of our main proposition showing that the algorithm produces exactly the set of measurable resource pooling pattern strategies. In Section 5 we discuss some important properties of the algorithm: We show how the algorithm can be viewed as a natural inductive reasoning procedure, and explain why the order in which we add Continuously Changing Workload preference limitations does not matter for the eventual output. In section 6 we include conclusion and future scope.

2. WHY ELIMINATION OF RESOURCE POOLING PATTERN STRATEGIES DOES NOT WORK

Most algorithms in the periodic Elastic Scaling literature [1], [27] proceed by successively modifying resource pooling pattern strategies from the operation cycle. Think, for instance, of iterated elimination of strictly (weakly) dominated resource pooling pattern strategies. As announced, the algorithm we propose for measurability is of a different nature since it is based on successively adding Continuously Changing Workload preference limitations rather than eliminating resource pooling pattern strategies. A natural question is why we do not stick to the process of eliminating resource pooling pattern strategies here. In this section we show why elimination of resource pooling pattern strategies does not work for measurability.

Let us first be precise about the class of resource pooling pattern strategy elimination procedures we consider. All the elimination procedures mentioned above have in common that at each round, only weakly dominated resource pooling pattern strategies in the cloud consumer cycle of periodic Elastic Scaling cycle [18], [20], [22] (but not necessarily all) are eliminated. Now, say that a resource pooling pattern strategy elimination procedure is regular if at every round, it eliminates a (possibly empty) subset of the set of weakly dominated resource pooling pattern strategies in the cloud consumer of periodic Elastic Scaling cycle [21].

3. NECESSARY AND SUFFICIENT CONDITION OF MEASURABILITY

3.1 Totally ordered Data-intensive systems

Totally ordered Data-intensive systems have been formally introduced as a possible way to represent a decision maker’s resource pooling pattern conjecture about the data-intensive state of the data-intensive world. The essential feature is that it allows the decision maker to deem one data-intensive state much more likely (in fact, probability) than some other data-intensive state, without completely ignoring the latter data-intensive state when making a decision.

More formally, let $N$ be some finite set of data-intensive states. By $\theta(N)$ we denote the set of all probability distributions on $N$. A Totally ordered Data-intensive systems (TODIS) on $N$ is a finite sequence of probability distributions $\psi = (\psi^1, \psi^2, \ldots, \psi^t)$, with $\psi^p \in \theta(N)$ for all $p \in \{1, \ldots, t\}$. We refer to $\psi^p$ as the decision maker’s level $1$ resource pooling pattern conjecture, to $\psi^2$ as its level $2$ resource pooling pattern conjecture, and so on. The interpretation is that the decision maker attaches much more importance to its level $1$ resource pooling pattern conjecture than to its level $2$ resource pooling pattern conjecture, attaches much more importance to its level $2$ resource pooling pattern conjecture than to its level $3$ resource pooling pattern conjecture, and so on, without completely discarding any of these conjecture. For every data-intensive state $n \in N$, let $lp(n, \psi)$ be the first level $p$ for which $\psi^p(n) > 0$. If $\psi^p(n) = 0$ for every $p \in \{1, \ldots, t\}$, set $lp(n, \psi) = \infty$. We call $lp(n, \psi)$ the rank of data-intensive state $n$ within the TODIS $\psi$. We say that the TODIS $\psi$ deems data-intensive state $n$ infinitely more likely than some other data-intensive state Proposition if $n$ has a lower rank than Proposition.

3.2 Periodic Elastic Scaling Planning Model

Consider a finite static periodic Elastic Scaling $\delta = (K_i, x_i)_{i \in I}$, where $I$ is the finite set of cloud service providers, the finite set $K_i$ denotes the set of strategies for cloud service provider $i$, and $x_i : \prod_{i \in I} K_i \rightarrow \mathbb{R}$ denotes cloud service provider $i$’s utility function. We assume that cloud service provider $i$ does not only have a resource pooling pattern conjecture about its cloud consumers’ resource pooling pattern strategy choices, but also about the possible conjecture that its cloud consumers could have about the other cloud service providers’ resource pooling pattern strategy choices, and about the possible conjecture that the cloud consumers could have about the possible conjecture that their cloud consumers could have about the other cloud service providers’ resource pooling pattern strategy choices, and so on. That is, cloud service provider $i$ hold a full resource pooling pattern conjecture hierarchy about the cloud consumers’ choices and the cloud consumers’ conjecture. If we assume, moreover, that each of the conjecture in this hierarchy can be represented by a
In principle, we could allow for infinitely many type combinations, and \( \psi \) is a function that assigns to every Continuously Changing Workload type \( t \in T \) some TODIS \( \psi(t) \) on the set \( K \times T \), of cloud consumers’ strategy–Continuously Changing Workload type combinations.

Here, \( K \) denotes the set of cloud consumers’ strategy combinations, and \( T \) the set of cloud consumers’ Continuously Changing Workload type combinations. The interpretation is that \( \psi(t) \) represents the resource pooling pattern conjecture that Continuously Changing Workload type \( t \) has about its cloud consumers’ choices and conjecture. For instance, the marginal of \( \psi(t) \) on \( P \) represents the resource pooling pattern conjecture that \( t \) has about cloud consumer \( j \)'s choice. Since every cloud consumer’s type \( t \) holds a resource pooling pattern conjecture about the other cloud service providers’ choices, we can derive from \( \psi(t_j) \) as well the resource pooling pattern conjecture that Continuously Changing Workload type \( t \) has about the resource pooling pattern conjecture that cloud service provider \( j \) has about its cloud consumers’ choices, and so on. In fact, from \( \psi(t_j) \) we can derive the full resource pooling pattern conjecture hierarchy that cloud service provider \( i \) has about its cloud consumers’ choices and conjecture.

The reader may wonder why we limit attention to periodic Elastic Scaling planning models with finitely many Continuously Changing Workload types for every cloud service provider. In principle, we could allow for infinitely many Continuously Changing Workload types for every cloud service provider, and define measurability for such infinite periodic Elastic Scaling planning models. But it can be shown that every measurable strategy in a finite periodic Elastic Scaling planning model can be supported by a measurable Continuously Changing Workload type within an infinite periodic Elastic Scaling planning model with finitely many Continuously Changing Workload types only. So, we do not “overlook” any measurable strategies by concentrating on finite Continuously Changing Workload type spaces only. As working with finite sets of Continuously Changing Workload types makes things easier, we have decided to solely concentrate on finite periodic Elastic Scaling planning models in this paper.

Note that within an periodic Elastic Scaling planning model, the totally ordered resource pooling pattern conjecture \( \psi(t) = (\psi(t_1), \ldots, \psi(t_J)) \) of an Continuously Changing Workload type \( t \) is, mathematically speaking, an TODIS on the set of data-intensive states \( K \times T \). For every cloud consumers’ resource pooling pattern strategy–Continuously Changing Workload type combination \( (k, t, j) \in K \times T \), we can thus define the rank \( \text{lp}(k, t, j, \psi(t)) \) of \((k, t, j)\) within \( \psi(t) \), being the lowest level \( p \) such that \( \psi(t_k, t, j) = 0 \). Remember that, by convention, \( \text{lp}(k, t, j, \psi(t)) = \infty \) whenever \((k, t, j)\) does not receive positive probability anywhere in \( \psi(t) \). We say that Continuously Changing Workload type \( t \) deems the resource pooling pattern strategy–Continuously Changing Workload type combination \((k, t, j)\) infinitely more likely than some other combination \((k', t, j)\) if the rank of \((k, t, j)\) is lower than the rank of \((k', t, j)\).

Similarly, we can define for every event \( A \subseteq K \times T \), of cloud consumers’ resource pooling pattern strategy–Continuously Changing Workload type combinations the associated rank by

\[
\text{lp}(A, \psi(t)) = \min \{ \text{lp}(k, t, j, \psi(t)) | (k, t, j) \in A \}.
\]

Hence, the rank of \( A \) is the lowest level \( p \) such that \( \psi(t) \) assigns positive probability to some element in \( A \). This necessary and sufficient condition then allows us to define the rank of an individual cloud consumer’s resource pooling pattern strategy–Continuously Changing Workload type pair \((k, t)\), simply by taking the rank of the event

\[
\{ (k, t) \} \times \prod K \times \{ t \} \times \prod T,
\]

So, we first take the marginal of the TODIS \( \psi(t) \) on \( K \times T \), and then take the rank of \((k, t)\) inside this marginal TODIS. In a similar fashion, we can also define the rank of an individual cloud consumer’s Continuously Changing Workload type \( t \), and of an individual cloud consumer’s resource pooling pattern strategy \( k \). As such, we can formally data-intensive state expectations like \( \psi(t) \) deems \((k, t)\) infinitely more likely than \((k', t)\) for cloud consumer \( j \) or \( \psi(t) \) deems \( k \) infinitely more likely than \( k' \) for cloud consumer \( j' \), which means that the rank of the former is smaller than the rank of the latter.

We say that Continuously Changing Workload type \( t \) deems possible some event \( A \subseteq K \times T \), if there is some level \( p \) with \( \psi(t(A)) = 0 \). That is, \( A \) is deemed possible if and only if \( \text{lp}(A, \psi(t)) = \infty \). Since we have defined the rank also for individual resource pooling pattern strategy–Continuously Changing Workload type pairs \((k, t)\) and for individual Continuously Changing Workload types \( t \), we can also formally define the event that Continuously Changing Workload type \( t \) deems possible a resource pooling pattern strategy–Continuously Changing Workload type pair \((k, t)\) for cloud consumer \( j \), and that \( t \) deems possible an cloud consumer’s Continuously Changing Workload type \( t \). It simply means that the associated rank is not \( \infty \).

### 3.3 Cautious Continuously Changing Workload Types

Intuitively, caution means that the cloud service provider should not fully exclude any cloud consumer’s Continuously Changing Workload choice from consideration. The formal necessary and sufficient condition is, however – in data-intensive states that an Continuously Changing Workload type \( t \) should not exclude any strategy choice for any cloud consumer’s Continuously Changing Workload type \( t \) considers possible. Hence, for every resource pooling pattern conjecture hierarchy that \( t \) deems possible for its cloud consumer \( j \), and for every measurable strategy \( k \) that \( j \) can possibly choose, Continuously Changing Workload type \( t \) should deem possible the event that its cloud consumer holds this resource pooling pattern conjecture hierarchy and chooses \( k \).

#### Necessary and sufficient condition 3.2 (Cautious Continuously Changing Workload type)

Consider an periodic Elastic Scaling planning model with sets of Continuously Changing Workload types \( T \), for every cloud service provider \( i \), Continuously Changing Workload type \( t \in T \), is cautious if, for every cloud consumer \( j \), every Continuously Changing Workload type \( j \in T \), it considers possible, and every resource pooling pattern strategy choice \( k \in K \). Continuously
3.4 Considering the cloud consumers’ resource pooling pattern preferences

The key condition for measurability is that an Continuously Changing Workload type should consider its cloud consumers’ Continuously Changing Workload resource pooling pattern preferences. In words it means that, whenever Continuously Changing Workload type \( t \) premises that its cloud consumer \( j \) prefers some resource pooling pattern strategy \( k_j \) to some other resource pooling strategy \( k' \), then it should deem \( k_j \) infinitely more likely than \( k' \). We must first define what it means, within our periodic Elastic Scaling planning model, that an Continuously Changing Workload type prefers some resource pooling strategy to another resource pooling pattern strategy.

Consider an Continuously Changing Workload type \( t \), with an TODIS \( \psi(t) = (\psi_1^t, \ldots, \psi_p^t) \) on \( K \times T \). Then, for every level \( p \in \{1, \ldots, P\} \) and every resource pooling pattern strategy \( k \), we can define the level \( p \) expected utility

\[
x(k, \psi_p^t) = \sum_{(k_{-1}, t_{-1}) \in K \times T^{-1}} \psi_p^t(k_{-1}, t_{-1}) x(k_{-1}, k_{-1}).
\]

This is the expected utility that would result by choosing \( k \) under the resource pooling pattern conjecture \( \psi_p^t \).

**Necessary and sufficient condition 3.3** (An Continuously Changing Workload type’s preference relation over resource pooling pattern strategies). Let \( t \in T \) be an Continuously Changing Workload type with TODIS \( \psi(t) = (\psi_1^t, \ldots, \psi_p^t) \) on \( K \times T \). Continuously Changing Workload type \( t \) prefers resource pooling pattern strategy \( k \) to some other resource pooling pattern strategy \( k' \) if there is some level \( p \in \{1, \ldots, P\} \) such that \( x(k, \psi_p^t) > x(k', \psi_p^t) \) and \( x(k, \psi_p^t) = x(k', \psi_p^t) \) for all \( o < p \).

For later purposes, we say that Continuously Changing Workload type \( t \) weakly prefers \( k \) to \( k' \) if \( t \) does not prefer \( k' \) to \( k \).

**Necessary and sufficient condition 3.4** (Considering the cloud consumers’ resource pooling pattern preferences). Let \( t \in T \) be a cautious Continuously Changing Workload type. Continuously Changing Workload type \( t \) considers the cloud consumer’s resource pooling pattern preferences if, for every cloud consumer \( j \), every Continuously Changing Workload type \( t \neq T \) deemed possible by \( t \), and every two strategies \( k_j, k'_j \) such that \( t \) prefers \( k_j \) to \( k'_j \), Continuously Changing Workload type \( t \) deems the pair \( (k_j, t) \) infinitely more likely than the pair \( (k'_j, t) \).

3.5 Measurability

We say that an Continuously Changing Workload type \( t \) is measurable if \( t \) is cautious and consider the cloud consumers’ resource pooling pattern preferences, premises that all cloud consumers are cautious and consider their cloud consumers’ resource pooling pattern preferences, premises that all cloud consumers premise that their cloud consumers are cautious and consider their cloud consumers’ resource pooling pattern preferences, and so on. In other words, \( t \) is cautious and consider the cloud consumers’ resource pooling pattern preferences, and expresses common resource pooling pattern conjecture in the event that cloud service providers are cautious and consider the cloud consumers’ resource pooling pattern preferences, and so on.

By additionally assuming that \( t \) itself is cautious and consider the cloud consumers’ resource pooling pattern preferences, we obtain the necessary and sufficient condition of a measurable Continuously Changing Workload type.

**Necessary and sufficient condition 3.5** (Common resource pooling pattern conjecture in “caution and consider of the cloud consumers’ resource pooling pattern preferences”). An Continuously Changing Workload type \( t \) expresses common resource pooling pattern conjecture in the event that cloud service providers are cautious and consider the cloud consumers’ resource pooling pattern preferences if \( t \) only deems possible cloud consumers’ Continuously Changing Workload types that are cautious and consider their cloud consumers’ resource pooling pattern preferences, only deems possible cloud consumers’ Continuously Changing Workload types that only deem possible cloud consumers’ Continuously Changing Workload types that are cautious and consider their cloud consumers’ resource pooling pattern preferences, and so on.

Finally, we say that a resource pooling pattern strategy \( k \) is measurable for cloud service provider \( i \) if it is optimal for some measurable Continuously Changing Workload type. Formally, a resource pooling pattern strategy \( k \) is called optimal for Continuously Changing Workload type \( t \) if \( t \) weakly prefers \( k \) to any other resource pooling pattern strategy.

**Necessary and sufficient condition 3.6** (measurable Continuously Changing Workload type). An Continuously Changing Workload type \( t \) is measurable if it is cautious and consider the cloud consumers’ resource pooling pattern preferences, and moreover expresses common resource pooling pattern conjecture in the event that cloud service providers are cautious and consider the cloud consumers’ resource pooling pattern preferences.

Finally, we say that a resource pooling pattern strategy \( k \) is measurable for cloud service provider \( i \) if it is optimal for some finite periodic Elastic Scaling planning model \( (T, \psi_i) \) and some measurable Continuously Changing Workload type \( t \) in \( T \) such that \( k \) is optimal for \( t \).

As we already mentioned before, the concept of a measurable resource pooling pattern strategy would not change if we would allow for infinite periodic Elastic Scaling planning models here.

4. ALGORITHM

In this section we will present an algorithm that always delivers all measurable resource pooling pattern strategies. Before doing so, we first provide some intuitive arguments that eventually will lead to the algorithm. Finally, we state our main result, namely that the algorithm yields precisely the set of measurable resource pooling pattern strategies in every periodic Elastic Scaling.

4.1 Road to the Algorithm

In Section II we have seen that elimination of (subsets of) weakly dominated resource pooling pattern strategies cannot work for measurability. So, what kind of procedure could work here? We start our informal investigation with the following well-known fact:

**Step 1.** Suppose that resource pooling pattern strategy \( k \) is weakly dominated on \( K_i \), by some randomized resource pooling pattern strategy \( \gamma_i \in \Omega(Y_i) \), where \( Y_i \) is a subset of resource pooling pattern strategies. Then, if cloud service
provider i is cautious, it will prefer some resource pooling pattern strategy in \( V_i \) to \( k_i \). We say that \( (k_i, V_i) \) is a resource pooling pattern \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \).

Here, \( \theta(V_i) \) denotes the set of probability distributions on \( V_i \). The reason for this fact is simple: If \( k_i \) is weakly dominated by resource pooling pattern \( \gamma_i \), then under every cautious totally ordered resource pooling pattern conjecture, \( k_i \) will be worse than \( \gamma_i \), and hence there must be some \( v_i \in V_i \), which is better than \( k_i \) under such a cautiously totally ordered resource pooling pattern conjecture. So, \( (k_i, V_i) \) will be a resource pooling pattern \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \).

Suppose now that cloud service provider \( i \) premise its cloud consumers are cautious and that it consider its cloud consumers’ resource pooling pattern preferences. If some cloud consumer’s resource pooling pattern strategy \( k_i \) is weakly dominated on \( K_j \) by some randomized resource pooling pattern strategy \( \gamma_j \in \theta(V_j) \), then we know by Step 1 that cloud service provider \( j \) will prefer some resource pooling pattern strategy in \( V_j \) to \( k_i \) in case it is cautious. As cloud service provider \( i \) indeed premises it is cautious, and consider \( j \)’s resource pooling pattern preferences, cloud service provider \( i \) must deem some resource pooling pattern strategy in \( V_i \) infinitely more likely than \( k_i \). We say that cloud service provider \( i \)’s totally ordered resource pooling pattern conjecture consider the preference limitation \( (k_i, V_i) \). This leads to the following observation:

**Step 2.** Suppose cloud service provider \( i \) premises its cloud consumers are cautious, and consider its cloud consumers’ resource pooling pattern preferences. Then, \( i \)’s totally ordered resource pooling pattern conjecture must consider every cloud consumer’s resource pooling pattern \textit{Continuously Changing Workload preference limitation} \( (k_i, V_i) \) generated in Step 1.

Say that a totally ordered resource pooling pattern conjecture for cloud service provider \( i \) assumes a set \( E, K_i \) of cloud consumers’ resource pooling pattern strategy combinations if it deems all resource pooling pattern strategy combinations inside \( E \) more likely than all resource pooling pattern strategy combinations outside \( E \). Suppose now that \( i \)’s totally ordered resource pooling pattern conjecture is cautious, and assumes some set \( E \), of cloud consumers’ resource pooling pattern strategy combinations. Assume, moreover, that its resource pooling pattern strategy \( k_i \) is weakly dominated on \( E \) by a randomized resource pooling pattern strategy \( \gamma_i \in \theta(V_i) \). Then, \( i \) must prefer some resource pooling pattern strategy in \( V_i \) to \( k_i \). The argument is basically the same as for Step 1, if we would “reduce” the periodic Elastic Scaling to cloud consumers’ resource pooling pattern strategy combinations in \( E \). We thus obtain the following step:

**Step 3.** Suppose that every totally ordered resource pooling pattern conjecture for cloud service provider \( i \) considering all \textit{Continuously Changing Workload preference limitations} from Step 1, assumes some \( E, K_i \) on which \( k_i \) is weakly dominated by some \( \gamma_i \in \theta(V_i) \). Suppose, moreover, that cloud service provider \( i \) is cautious, premises its cloud consumers are cautious, and consider the cloud consumers’ resource pooling pattern preferences. Then, \( i \) must prefer some resource pooling pattern strategy in \( V_i \) to \( k_i \). We say that \( (k_i, V_i) \) is a new \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \).

Of course, we can iterate this argument if we assume that cloud service provider \( i \) is cautious, consider the cloud consumers’ resource pooling pattern preferences, and expresses common resource pooling pattern conjecture in the event that cloud service providers are cautious and consider the cloud consumers’ resource pooling pattern preferences. That is, if we assume that cloud service provider \( i \)'s \textit{Continuously Changing Workload type} is measurable. The inductive step would then look as follows:

**Inductive step.** Suppose that every totally ordered resource pooling pattern conjecture for \( i \) that consider all \textit{Continuously Changing Workload preference limitations} generated so far, assumes some \( E, K_i \) on which \( k_i \) is weakly dominated by some \( \gamma_i \in \theta(V_i) \). Then, if \( i \) is of a measurable \textit{Continuously Changing Workload type}, it must prefer some resource pooling pattern strategy in \( V_i \) to \( k_i \). So, \( (k_i, V_i) \) would be a new \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \).

This would thus generate an inductive procedure in which at every step (possibly) some new \textit{Continuously Changing Workload preference limitations} would be added for the cloud service providers. Since there are only finitely many possible \textit{Continuously Changing Workload preference limitations} for the cloud service providers, this procedure must end after finitely many steps. Now, consider some cloud service provider \( i \), and its set of \textit{Continuously Changing Workload preference limitations} generated by the procedure above.

If cloud service provider \( i \) is of some measurable \textit{Continuously Changing Workload type}, we know from our arguments above that it will never choose a resource pooling pattern strategy \( k_i \) if it is part of some \textit{Continuously Changing Workload preference limitation} \( (k_i, V_i) \). In that case, namely, it would always prefer some resource pooling pattern strategy in \( V_i \) to \( k_i \), so \( k_i \) could not be optimal.

So, the procedure above rules out resource pooling pattern strategies that is certainly not measurable. But what about the converse? So, what about resource pooling pattern strategies that are not ruled out by the procedure above? The main proposition in this paper, Proposition 4.6, will show that the “surviving” resource pooling pattern strategies are all measurable! Hence, the procedure above will always select exactly those resource pooling pattern strategies that are measurable – not more and not less.

**4.2 Description of the algorithm**

Before we state the algorithm, we first formally necessary and sufficient condition the new concepts we described above, such as \textit{Continuously Changing Workload preference limitations}, what it means for a totally ordered resource pooling pattern conjecture to consider an \textit{Continuously Changing Workload preference limitation}, and so on.

**Necessary and sufficient condition 4.1 (\textit{Continuously Changing Workload preference limitation}).** An \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \) is a pair \((k_i, V_i)\) where \( k_i \) is a resource pooling pattern strategy, and \( V_i \) a nonempty subset of resource pooling pattern strategies.

The interpretation is that cloud service provider \( i \) prefers at least one resource pooling pattern strategy from \( V_i \) to \( k_i \). Now, consider a totally ordered resource pooling pattern conjecture \( \psi_k \), and hence there must be some \( v_i \in V_i \), which is better than \( k_i \) under such a cautiously totally ordered resource pooling pattern conjecture. So, \( (k_i, V_i) \) would be a new \textit{Continuously Changing Workload preference limitation} for cloud service provider \( i \).

17
every resource pooling pattern strategy combination in $K_i$ receives positive probability in some level of $\psi_i$.

**Necessary and sufficient condition 4.2 (Considering a Continuously Changing Workload preference limitation).** A totally ordered resource pooling pattern conjecture $\psi_i$ on $K_i$ consider an Continuously Changing Workload preference limitation $(k_i, V_i)$ for cloud service provider $i$ if $\psi_i$ deems some resource pooling pattern strategy in $V_i$ infinitely more likely than $k_i$.

This, in a sense, mimics the requirement that cloud service provider $i$ must consider $j$’s resource pooling pattern preferences.

**Necessary and sufficient condition 4.3 (Assuming a set of cloud consumers’ resource pooling pattern strategy combinations).** Consider a subset $E_i$ of cloud consumers’ resource pooling pattern strategy combinations, and a totally ordered resource pooling pattern conjecture $\psi_i$ on $K_i$. The totally ordered resource pooling pattern conjecture $\psi_i$ assumes the set $E_i$, if $\psi_i$ deems all resource pooling pattern strategy combinations inside $E_i$, infinitely more likely than all resource pooling pattern strategy combinations outside $E_i$.

Note that a totally ordered resource pooling pattern conjecture $\psi_i := (\psi_i^1, \ldots, \psi_i^p)$ on $K_i$ is a subset $E_i \subseteq K_i$, if and only if, there is some level $p \in \{1, \ldots, P\}$ such that $U_{\psi_i} \sup\{j \in E_i\}$. Here, $\sup\{\psi_i^p\}$ denotes the support of the probability distribution $\psi_i^p$. A randomized resource pooling pattern strategy for cloud service provider $i$ is a probability distribution $\gamma_i \in \Theta(K_i)$, on cloud service provider $i$’s resource pooling pattern strategies. For a subset $V_i \subseteq K_i$, we denote by $\Theta(V_i)$ the set of randomized resource pooling pattern strategies that assign positive probability only to resource pooling pattern strategies in $V_i$. For some cloud consumers’ resource pooling pattern strategy combination $k_i \in K_i$, let

$$x_i(\gamma_i, k_i) := \sum_{k \in E_i} \gamma_i(k) x_i(k, k_i)$$

denote $i$’s expected utility from the randomized resource pooling pattern strategy $\gamma_i$ and the cloud consumers’ resource pooling pattern strategy combination $k_i$.

**Necessary and sufficient condition 4.4 (Weakly dominated resource pooling pattern strategy).** Let $E_i \subseteq K_i$ be a subset of the cloud consumers’ resource pooling pattern strategy combinations. Resource pooling pattern Strategy $k_i$ is said to be weakly dominated by randomized resource pooling pattern strategy $\gamma_i$ on $E_i$, if $x_i(\gamma_i, k_i) \geq x_i(k, k_i)$ for all $k, \in E_i$, with strict in equality for at least some $k_i \in E_i$.

We are now ready to present the algorithm. The idea is to start with the empty set of Continuously Changing Workload preference limitations for all cloud service providers, and at every round to add new Continuously Changing Workload preference limitations, if possible. For that reason, the algorithm is called “iterated addition of Continuously Changing Workload preference limitations”.

**Algorithm 4.5 (Iterated addition of Continuously Changing Workload preference limitations).** In round 1, begin for all cloud service providers $i$ with the empty set of Continuously Changing Workload preference limitations.

At every further round $q \geq 2$, limit for every cloud service provider $i$ to those totally ordered resource pooling pattern conjecture on $K_i$ that consider all cloud consumers’ Continuously Changing Workload preference limitations generated so far. Add a new Continuously Changing Workload preference limitation $(k_i, V_i)$ for cloud service provider $i$ if every such totally ordered resource pooling pattern conjecture assumes some set $E_i \subseteq K_i$, on which $k_i$ is weakly dominated by some $\gamma_i(\Theta(V_i))$.

Since the number of Continuously Changing Workload preference limitations is finite, this algorithm must end after a finite number of rounds. We say that resource pooling pattern strategy $k_i$ survives the algorithm of iterated addition of Continuously Changing Workload preference limitations if $k_i$ is not part of any Continuously Changing Workload preference limitation $(k_i, V_i)$ generated by the algorithm. Namely, if $k_i$ were to be part of an Continuously Changing Workload preference limitation $(k_i, V_i)$ produced by the algorithm, then cloud service provider $i$ would prefer at least one strategy in $V_i$ to $k_i$ and hence $k_i$ could not be optimal.

**4.3 Main Proposition**

Our main proposition states that the algorithm of iterated addition of Continuously Changing Workload preference limitations yields exactly the set of measurable resource pooling pattern strategies for every cloud service provider.

**Proposition 4.6 (Algorithm yields precisely the set of measurable resource pooling pattern strategies).** Consider a finite static periodic Elastic Scaling. Then, a resource pooling pattern strategy $k_i$ is measurable, if and only if, $k_i$ survives the algorithm of iterated addition of Continuously Changing Workload preference limitations.

The easier direction is to show that every measurable resource pooling pattern strategy survives iterated addition of Continuously Changing Workload preference limitation. So, a measurable resource pooling pattern strategy $k_i$ can never be part of an Continuously Changing Workload preference limitation $(k_i, V_i)$ generated by the algorithm. The more difficult direction is to prove that every resource pooling pattern strategy $k_i$ that is not part of any such Continuously Changing Workload preference limitation $(k_i, V_i)$ is measurable. Hence, we must construct an periodic Elastic Scaling planning model in which each of this resource pooling pattern strategies $k_i$ is supported by some measurable Continuously Changing Workload type. This construction is rather delicate.

From the proposition, we can easily derive the following observation: If in a given periodic Elastic Scaling no resource pooling pattern strategy is weakly dominated, then all resource pooling pattern strategies for the cloud service providers are measurable. Namely, the algorithm we present will only generate Continuously Changing Workload preference limitations at the first round if there is at least some resource pooling pattern strategy that is weakly dominated within the full periodic Elastic Scaling. Otherwise, the algorithm will not generate any Continuously Changing Workload preference limitation at all, and hence all resource pooling pattern strategies would survive the algorithm.

**4.4 A Finite Formulation of the Algorithm**

The algorithm of iterated addition of Continuously Changing Workload preference limitations as we have formulated it proceeds by adding Continuously Changing Workload preference limitations and deleting totally ordered conjecture at every round. More precisely, we start with the empty set of Continuously Changing Workload preference limitations and the full set of totally ordered conjecture. At the first round we see whether we can add some Continuously Changing Workload preference limitations. If so, then this would reduce
the set of totally ordered conjecture, which at the next round could add some further Continuously Changing Workload preference limitations, and so on.

What is somewhat undesirable from a computational point of view is that there are infinitely many possible totally ordered conjecture in the periodic Elastic Scaling. This would suggest that at every round in the algorithm we must scan through infinitely many totally ordered conjecture. This, however, is not necessary. What matters for the algorithm is not so much the precise probabilities in the totally ordered resource pooling pattern conjecture, but the induced “likelihood resource pooling pattern ordering” on cloud consumers’ resource pooling pattern strategy combinations. More precisely, let \( \psi = (\psi_1, \ldots, \psi_j) \) be a totally ordered resource pooling pattern conjecture on \( K_i \). Remember our convention that \( \psi_j \) has full support on \( K_i \), that is, every \( k_i \in K_i \), receives positive probability in some level \( \psi_j \). Let \( O = (O_1^j, \ldots, O_j^j) \) be the ordered sequence of disjoint subsets \( O_i^j \subseteq K_i \), such that (a) \( \psi_j \) deems every \( k_i \in O_i^j \) infinitely more likely than every \( k_i \in O_i^{j+1} \) for every \( z \in \{1, \ldots, Z \} \), (b) for every \( m \) and every \( k_i \), \( k_i \in O_i^j \), the TODIS \( \psi_j \) does not deem \( k_i \) more likely than \( k_i \), nor vice versa, and (c) the union of the sets in \( O \) is \( K_i \). We call \( O \), the likelihood ordering induced by \( \psi \). Formally, we have the following necessary and sufficient condition.

**Necessary and sufficient condition 4.7 (Likelihood ordering).** A likelihood ordering for cloud service provider \( i \) on the cloud consumers’ resource pooling pattern strategy combinations is an ordered sequence \( O = (O_1^j, \ldots, O_j^j) \) where \( O_1^j, \ldots, O_j^j \) are pair-wise disjoint subsets of \( K_i \), whose union is equal to \( K_i \).

So, the interpretation is that \( O \), deems all resource pooling pattern strategy combinations in \( O_1^j \) infinitely more likely than all resource pooling pattern strategy combinations in \( O_2^j \), deems all resource pooling pattern strategy combinations in \( O_2^j \) infinitely more likely than all resource pooling pattern strategy combinations in \( O_3^j \), and so on. It is clear that there are only finitely many likelihood orderings in the periodic Elastic Scaling, since there are only finitely many resource pooling pattern strategies for every cloud service provider.

We can now easily extend the necessary and sufficient condition of “considering an Continuously Changing Workload preference limitation” and “assuming a set of cloud consumers’ resource pooling pattern strategy combinations” to likelihood orderings. Say that a likelihood resource pooling pattern ordering \( O = (O_1^j, \ldots, O_j^j) \) consider an Continuously Changing Workload preference limitation \( (k_i, V_i) \) if \( O \) deems some resource pooling pattern strategy in \( V_i \) infinitely more likely than \( k_i \). Also, the likelihood ordering \( O \) is said to assume the set \( E_{\gamma} \) of cloud consumers’ resource pooling pattern strategy combinations if \( O \) deems all resource pooling pattern strategy combinations inside \( E_{\gamma} \) infinitely more likely than all resource pooling pattern strategy combinations outside \( E_{\gamma} \). The algorithm of iterated addition of Continuously Changing Workload preference limitations can thus alternatively be stated as follows:

**Algorithm 4.8 (Finite version).** In round 1, begin for all cloud service providers \( i \) with the empty set of Continuously Changing Workload preference limitations.

At every further round \( q \geq 2 \), limit for every cloud service provider \( i \) to those likelihood resource pooling pattern orderings on \( K_i \) that consider all cloud consumers’ Continuously Changing Workload preference limitations generated so far. Add a new Continuously Changing Workload preference limitation \( (k_i, V_i) \) for cloud service provider \( i \) if every such likelihood resource pooling pattern ordering assumes some set \( E_{\gamma} \subseteq K_i \), on which \( k_i \) is weakly dominated by some \( \gamma \in O(V_i) \).

The advantage of this formulation is that at every round, we only have to scan through finitely many objects, as there are only finitely many Continuously Changing Workload preference limitations and likelihood resource pooling pattern orderings in the periodic Elastic Scaling. Obviously, this algorithm generates precisely the same set of Continuously Changing Workload preference limitations as the original procedure. As such, the measurable resource pooling pattern strategies are precisely those resource pooling pattern strategies that survive this alternative algorithm.

5. DISCUSSION

In this section we will discuss some important properties of the algorithm.

5.1 Algorithm as an inductive reasoning procedure

The algorithm is not merely a tool to compute the measurable resource pooling pattern strategies in an periodic Elastic Scaling, but can also be interpreted as an inductive reasoning process that can be used by a cloud service provider who reasons in the spirit of measurability. Consider namely a fixed cloud service provider in the periodic Elastic Scaling, say cloud service provider \( i \). In round 2, the algorithm would add for every cloud consumer \( j \) an Continuously Changing Workload preference limitation \( (k_j, V_j) \) if \( k_j \) would be weakly dominated on \( K_j \) by a mixture on \( V_j \). In that case, cloud service provider \( i \) would store the Continuously Changing Workload preference limitation \( (k_i, V_i) \) in its mind, meaning that he premises that cloud service provider \( j \) prefers some resource pooling pattern strategy in \( V_j \) to \( k_i \). If \( i \) consider \( j \)’s resource pooling pattern preferences, then it should consequently deem some resource pooling pattern strategy in \( V_j \) infinitely more likely than \( k_i \). That is, the Continuously Changing Workload preference limitations that cloud service provider \( i \) would store in its mind at round 2 would limit the possible totally ordered conjecture it could hold about its cloud consumers’ choices. Moreover, if cloud service provider \( i \) premises that its cloud consumers reason similarly, then cloud service provider \( i \) can actually deduce the possible totally ordered conjecture that its cloud consumers may hold at this round.

In the next round of its reasoning procedure, cloud service provider \( i \) would then ask for every cloud consumer \( j \). Given its limited set of conjecture, would cloud service provider \( j \) always assume some set \( E_{\gamma} \subseteq K_j \), on which some resource pooling pattern strategy \( k_j \) would always be weakly dominated by a mixture on \( V_j \)? If yes, then cloud service provider \( i \) will store \( (k_i, V_i) \) as a new Continuously Changing Workload preference limitation in its mind. In doing so, cloud service provider \( i \) would then further limit the possible totally ordered conjecture it could hold about its cloud consumers. Cloud service provider \( i \) could continue this inductive reasoning procedure until no new Continuously Changing Workload preference limitation could be added, and hence its possible totally ordered conjecture could not be limited any further.

So we see that the algorithm may serve very well as an intuitive reasoning procedure for cloud service providers that will eventually lead them to the measurable resource pooling pattern strategies in the periodic Elastic Scaling. What is
crucial in this reasoning procedure is that a cloud service provider only needs to keep track of Continuously Changing Workload preference limitations, which substantially simplifies matters compared to the original necessary and sufficient condition of measurability. In that light, our main proposition thus says that in order to find the measurable resource pooling pattern strategies in an periodic Elastic scaling, it is sufficient for a cloud service provider to think in terms of Continuously Changing Workload preference limitations, and to reason in accordance with the algorithm.

In the periodic Elastic scaling literature [1], there are other algorithms that can nicely be interpreted as intuitive reasoning procedures. Take, for instance, the concept of common resource pooling pattern conjecture in measurability and the associated algorithm of iterated elimination of strictly dominated resource pooling pattern strategies. Here, the algorithm can be seen as a reasoning procedure in which a cloud service provider successively deletes cloud consumers’ resource pooling pattern strategies, since they can no longer be optimal. At every round, this would then limit the cloud service provider’s possible conjecture as it must assign probability zero to these resource pooling pattern strategies. These additional limitations on the cloud service providers’ conjecture could then induce further resource pooling pattern strategies that can be deleted, and so on. So, in that procedure the cloud service providers’ possible (non-totally ordered) conjecture are limited further and further by deleting resource pooling pattern strategies, whereas in our procedure the (totally ordered) conjecture are limited further and further by adding new Continuously Changing Workload preference limitations.

A similar story can be told for the concept of iterated assumption of measurability within a complete Continuously Changing Workload type structure and the associated algorithm of iterated elimination of weakly dominated resource pooling pattern strategies. Here, the algorithm reflects a reasoning procedure in which a cloud service provider with totally ordered conjecture iterately deletes weakly dominated resource pooling pattern strategies from its mind. At every round of this procedure, the cloud service provider will then deem all surviving resource pooling pattern strategies infinitely more likely than all deleted resource pooling pattern strategies, thus limiting the possible totally ordered conjecture it can hold. So also in this procedure, the cloud service provider’s possible conjecture is limited in every round by deleting resource pooling pattern strategies.

5.2 Order Independence

For the algorithm, it can be shown that the order and speed in which we add preference restrictions does not matter for the eventual result. That is, it does not matter whether in every round we add all preference restrictions that can possibly be generated, or only some of these.

To see this, let us compare two procedures, Procedure 1 and Procedure 2, where in the first we always add all possible Continuously Changing Workload preference limitations at every round, and in the second we only add some of the possible Continuously Changing Workload preference limitations every time. Then, first of all, Procedure 1 will at every round generate at least as many Continuously Changing Workload preference limitations as Procedure 2. Namely, at round 2 Procedure 1 generates as least as many Continuously Changing Workload preference limitations, by necessary and sufficient condition. Therefore, at round 3 Procedure 1 limits to a smaller set of totally ordered conjecture than Procedure 2.

But then, under Procedure 1 it will be “easier” to generate new Continuously Changing Workload preference limitations at round 3 than under Procedure 2. Hence, at round 3 Procedure 1 will, again, generate at least as many Continuously Changing Workload preference limitations as Procedure 2, and so on. So, eventually, Procedure 1 will generate at least as many Continuously Changing Workload preference limitations as Procedure 2. The key argument here was that a larger set of Continuously Changing Workload preference limitations will lead to a smaller set of possible totally ordered conjecture, and a smaller set of possible totally ordered conjecture will in turn lead to a larger set of induced Continuously Changing Workload preference limitations. So, the algorithm is monotone in this sense.

On the other hand, it can also be shown that every Continuously Changing Workload preference limitation generated by Procedure 1 will also eventually be generated by Procedure 2. Suppose, namely, that Procedure 1 would generate some Continuously Changing Workload preference limitation that would not be generated at all by Procedure 2. Then, let \( p \) be the first round at which Procedure 1 would generate an Continuously Changing Workload preference limitation, say \((k, V_i)\), not generated by Procedure 2 at all. By construction of the algorithm, every totally ordered resource pooling pattern conjecture for cloud service provider \( i \) that consider all Continuously Changing Workload preference limitations generated by Procedure 1 before round \( p \), must assume some set \( E_j \) on which \( k_j \) is weakly dominated by some \( \gamma_j \in \theta(V_j) \). By our assumption, all these Continuously Changing Workload preference limitations generated by Procedure 1 before round \( p \) are also eventually generated by Procedure 2, let us say before round \( z \geq p \). But then, every totally ordered resource pooling pattern conjecture for cloud service provider \( i \) that consider all Continuously Changing Workload preference limitations generated by Procedure 2 before round \( z \), assumes a set \( E_j \) on which \( k_j \) is weakly dominated by some \( \gamma_j \in \theta(V_j) \).

Hence, Procedure 2 must add the Continuously Changing Workload preference limitation \((k, V_i)\) sooner or later, which is a contradiction since we assumed that Procedure 2 does not generate Continuously Changing Workload preference limitation \((k, V_i)\) at all. We thus conclude that every Continuously Changing Workload preference limitation added by Procedure 1 is also finally added by Procedure 2. As such, Procedures 1 and 2 eventually generate exactly the same set of Continuously Changing Workload preference limitations. So, indeed, the order and speed in which we add Continuously Changing Workload preference limitations is irrelevant to the algorithm.

6. CONCLUSION AND FUTURE SCOPE

In this section we conclude, stating that the algorithm of iterated addition of Continuously Changing Workload preference limitations selects exactly the set of measurable resource pooling pattern strategies in the periodic Elastic scaling. For our conclusion, we recall the necessary and sufficient condition of a likelihood resource pooling pattern ordering induced by a TODIS. Consider a TODIS \( \psi = (\psi_1, \ldots, \psi_z) \) on \( K_\ast \). Remember our convention that \( \psi_1 \) has full support on \( K_\ast \), that is, every \( k \in K_\ast \) receives positive probability in some level \( \psi_1^z \). Let \( O = \{O_1, \ldots, O_{2^z} \} \) be the ordered sequence of disjoint subsets \( O_v \subseteq K_\ast \) such that (a) \( \psi_1 \) deems every \( k_v \in O_1 \) infinitely more likely than every \( k_v \in O_{v+1} \), for every \( z \in [1, \ldots, 2^z - 1] \), (b) for every \( z \) and every \( k_v, k_{v'} \in O_z \), the TODIS \( \psi_1 \) does not deem \( k_v \) infinitely more likely than \( k_{v'} \), nor vice versa, and (c) the union of the sets in
$O_i$ is $K_{\text{w}}$. We call $O_i$ the likelihood resource pooling pattern ordering induced by $\psi_i$. Our conclusion characterizes, for a given resource allocation pattern $k_i$ and set $V_i \subseteq K_{\text{w}}$ those likelihood resource pooling pattern orderings on $K_{\text{w}}$ that admit an TODIS under which $k_i$ is weakly preferred to all resource pooling pattern strategies in $V_i$. Despite the progress on our interpretation is made the following three open problems are available for further research.

**Open Problem 6.1** Let $\psi_i$ be a TODIS on $K_{\text{w}}$. let $k_i$ be a resource pooling pattern strategy and $V_i \subseteq K_{\text{w}}$ a subset of resource pooling pattern strategies.(a) If under the TODIS $\psi_i$, resource pooling pattern strategy $k_i$ is weakly preferred to all resource pooling pattern strategies in $V_i$. Does $\psi_i$ assume any $E \subseteq K_{\text{w}}$, on which $k_i$ is weakly dominated by a mixture on $V_i$? (b) If $\psi_i$ does not assume any $E \subseteq K_{\text{w}}$, on which $k_i$ is weakly dominated by a mixture on $V_i$. Does some TODIS $\psi_{i'}$ inducing the same likelihood resource pooling pattern ordering as $\psi_i$, under which $k_i$ is weakly preferred to all resource pooling pattern strategies in $V_i$?

**Open Problem 6.2** Let $t_i$ be a measurable Continuously Changing Workload type. Does $t_i$'s totally ordered resource pooling pattern conjecture on $K_{\text{w}}$, consider every Continuously Changing Workload preference limitation in $F_{t_i}$?

**Open Problem 6.3** (Property of Continuously Changing Workload preference limitations not generated by the algorithm) For every cloud service provider $i$, let $F^*_{t_i}$ be the set of Continuously Changing Workload preference limitations not generated by the algorithm. Does for $(k_i, V_i) \in F^*_{t_i}$ is there an TODIS $\psi_i$ on $K_{\text{w}}$ such that $(1)$ under $\psi_i$, resource pooling pattern strategy $k_i$ is weakly preferred to all resource pooling pattern strategies in $V_i$, and $(2)$ for every cloud consumer's resource pooling pattern strategy $k_i$ the pair $(k_i, V_j(k_i, \psi_i))$ is in $F^*_{t_j}$?

7. REFERENCES


