

Forecasting Duration Intervals of Scientific Workflow Activities based on Time-Series Patterns

Xiao Liu, Jinjun Chen, Ke Liu, Yun Yang

*Faculty of Information and Communication Technologies, Swinburne University of Technology
Hawthorn, Melbourne, Australia 3122
{xliu, jchen, kliu, yyang}@swin.edu.au*

Abstract

In scientific workflow systems, time related functionalities such as workflow scheduling and temporal verification normally require effective forecasting of activity durations due to the dynamic nature of underlying resources such as Web or Grid services. However, most existing strategies cannot handle well the problems of limited sample size and frequent turning points which are typical for the duration series of scientific workflow activities. To address such problems, we propose a novel pattern based time-series forecasting strategy which utilises a periodical sampling plan to build representative duration series, and then conducts time-series segmentation to discover the smallest pattern set and predicts the activity duration intervals with pattern matching results. The simulation experiment demonstrates the excellent performance of our segmentation algorithm and further shows the effectiveness of our strategy in the prediction of activity duration intervals, especially the ability of handling turning points.

1. Introduction

Scientific workflow usually underlies in many complex e-science applications such as climate modelling, disaster recovery simulation, astrophysics and high energy physics [4][17]. In reality, scientific and business processes are normally subjected to some temporal constraints, e.g. relative durations or fixed deadlines. Hence, scientific workflow functionalities such as workflow scheduling and temporal verification are provided to control and monitor the overall temporal correctness of scientific workflow execution [3][9]. These functionalities often demand a strategy for the prediction of activity duration intervals, namely the upper bound and lower bound of activity completion time, in order to make effective decisions.

In both scientific and business fields, time-series models are probably the most widely used statistical ways for formulating and forecasting the dynamic

behaviour of complex systems. A time series is a set of observations made sequentially through time [2]. Some representative time series, including marketing time series, temperature time series and quality control time series, are effectively applied in various scientific and business domains [6][7]. Similarly, a scientific workflow activity duration time series, or duration series for short, is composed of ordered duration samples obtained from scientific workflow system logs or other forms of historical data. Current forecasting strategies for computation tasks mainly reside on the prediction of CPU load [1][15], however, this is quite different from the prediction of scientific workflow activity durations. Activity durations cover the time intervals from the initial submission to the final completion of each workflow activity [3]. Hence, besides the exact execution time on scheduled resources, they also consist of extra time, i.e. scientific workflow overheads. According to [11], there exist four main categories of scientific workflow overheads, i.e. middleware overhead, data transfer overhead, loss of parallelism overhead and activity related overhead. Evidently, scientific workflow activity durations involve much more affecting factors than that of conventional computation tasks which are dominated by high performance computing resources.

In this paper, we conduct univariate time-series analysis which analyses the behaviour of time series itself to build a model for the correlation between its neighbouring samples [2]. However, two problems of the duration series, i.e. limited sample size and frequent turning points, potentially hinder the effectiveness of conventional time-series models. Limited sample size evidently impedes the fitting of time-series models [10]. Meanwhile, frequent turning points where dramatic deviations from the previous time-series sequences occur significantly deteriorate the overall accuracy of time-series forecasting [19]. To address such two problems, we propose a pattern based time-series forecasting strategy. First, an effective periodical sampling plan is conducted to build representative duration series. Second, a pattern recognition process utilises a novel time-series segmentation algorithm to discover the minimum number of potential patterns

which are further validated and associated with specified turning points. Third, given the latest duration sequences, pattern matching and interval forecasting are performed to make predictions based on the statistical features of the best-matched patterns. Meanwhile, concerning the occurrences of turning points, three types of duration sequences are identified and then handled with different pattern matching results. The simulation experiment conducted in our SwinDeW-G (**Swinburne Decentralised Workflow for Grid**) scientific workflow system [16] verifies that our time-series segmentation algorithm is capable of discovering the smallest potential pattern set compared with three generic algorithms, i.e. Sliding Windows, Top-Down and Bottom-Up [8]. It further demonstrates the performance of our strategy in the prediction of high confidence duration intervals and the handling of turning points by the comparison to two intuitive yet practical methods of MEAN and LAST [6][10].

The remainder of the paper is organised as follows. Section 2 presents the related work and problem analysis. Section 3 defines the duration-series patterns and proposes our pattern based time-series forecasting strategy. Section 4 presents the algorithms designed for the strategy. Section 5 demonstrates a comprehensive simulation experiment to evaluate the effectiveness of our strategy. Finally, Section 6 addresses our conclusion and future work.

2. Related work and problem analysis

2.1. Related work

Five service performance estimation approaches, i.e. simulation, analytical modelling, historical data, on-line learning and hybrid, are proposed in [17]. In general, time-series forecasting belongs to historical data based approaches. Currently, most work focuses on the prediction of CPU load since it is the dominant factor for the durations of computation intensive activities. Typical linear time-series models such as AR, MA and Box-Jenkins are fitted to perform host load prediction in [6] where it claims that most time-series models are sufficient for host load prediction and recommend AR(16) which consumes miniscule computation cost. [18] proposes a one-step-ahead and low-overhead time-series forecasting strategy which tracks recent trends by giving more weight to recent data. A hybrid model which integrates the AR model with confidence interval is proposed in [15] to perform n-step-ahead load prediction. A different strategy proposed in [13] predicts application duration with historical information where search techniques are employed to determine those application characteristics that yield the best definition of similarity. The work in [1] investigates extended forecast of both CPU and network load on computation

grid to achieve effective load balancing. The authors in [19] mention the problem of turning points which brings large prediction errors to time-series models. However, few solutions have been proposed to investigate more affecting factors and handle turning points.

Time-series patterns can be employed for complex prediction tasks, such as the trend analysis and value estimation in stock market, product sales and weather forecast [2][5]. The major two tasks in pattern based time-series forecasting are pattern recognition which discovers time-series patterns according to the pattern definition and pattern matching which searches for the similar patterns with given query sequences [7]. To facilitate complex time-series analysis such as time-series clustering and fast similarity search, time-series segmentation is often employed to reduce the variances in each segment so that they can be described by simple linear or non-linear models [5][8]. In general, most segmentation algorithms can be classified into three generic algorithms with some variations, i.e. Sliding Windows, Top-Down and Bottom-Up. The process of Sliding Windows is like sequential scanning where the segments keep increasing with each new point until some thresholds are violated. The basic idea of the Top-Down algorithm is the repetition of the splitting process where the original time series is equally divided until all segments meet the stopping criteria. In contrast to Top-Down, the basic process of the Bottom-Up algorithm is merging where the finest segments are merged constantly until some thresholds are violated. However, hybrid algorithms which take advantages of the three generic algorithms by modifications and combinations are often more practical [7][8][14].

2.2. Problem analysis

As we mentioned earlier, the following two problems of duration series may hinder the implementation of conventional linear univariate time-series models.

- 1) Limited sample size. Current work on the prediction of CPU load such as AR(16) demands a large sample size to fit the model and at least 16 prior points to make one prediction. However, this becomes a real issue for scientific workflow activity duration series. Unlike CPU load which reflects an instant state and the observations can be measured at any time with high sampling rates such as 1Hz [6][19], scientific workflow activity durations denote discrete long-term intervals since activity instances only occur occasionally and take several minutes or more than a couple of hours to complete. Meanwhile, due to the limits of available resources, the number of concurrent activity instances is usually constrained. Therefore, the duration sample size is often limited and linear time-series models cannot be fitted effectively. However, time-series patterns are not restricted by this problem since their lengths are very scalable according to the behaviour of the time series [7].

2) Frequent turning points. A significant problem which deteriorates the effectiveness of linear time-series models is the occurrence of turning points where large deviations from the previous duration sequences take place [19]. This problem resides not only in the prediction of CPU load, but also in the forecasting of activity durations. Actually, due to the uneven distribution of activity instances, the system environment is usually of great differences even between neighbouring duration-series segments. For example, the average number of concurrent activity instances observed in the unit of (10:00am~11:00am) may well be bigger than that of (9:00am~10:00am) due to the normal schedule of working hours. Therefore, activity instances are unevenly distributed and frequent turning points are hence often expected. This further emphasises the importance of effective turning points discovery and handling in the forecasting of scientific workflow activity durations.

Based on the above analysis, rather than fitting conventional linear time-series models in this paper, we investigate the idea of pattern based time-series forecasting. To our best knowledge, this is the first time to investigate time-series patterns to predict scientific workflow activity durations.

3. Pattern based forecasting strategy

3.1. Duration-series patterns

The motivation for pattern based time-series forecasting comes from the observation that for those duration-series segments where the number of concurrent activity instances is similar, these activity durations reveal comparable statistical features. It is obvious that when the number of concurrent activity instances increases, the average resources that can be scheduled for each activity decrease while the overall workflow overheads increase, and vice versa. Accordingly, the duration series behaves up and down though it may not necessarily follow a linear relationship. This phenomenon is especially evident when scientific workflow systems adopt market-driven or trust-driven scheduling strategies that give preferences to a limited range of resources [17]. Furthermore, if these segments reappear frequently during a duration series, they can be deemed as potential patterns which represent unique behaviour of the duration-series under certain circumstances. Therefore, if we are able to define and discover these typical patterns, the intervals of future durations can be estimated with the statistical features of the closest patterns by matching the latest duration sequences. Based on this motivation, here, we present the definition of duration-series patterns as follows.

Definition (Duration-series patterns): For a specific duration series $DS = \{X_1, X_2 \dots X_n\}$ which consists of n

observations, its pattern set is $Patterns = \{P_1, P_2 \dots P_m\}$

where $\sum_{i=1}^m Length(P_i) \leq n$. For pattern P_i of length k , it is a recurrent segment which has unique statistical features of sample mean μ and sample standard deviation σ , where:

$$\mu = \frac{\sum_{i=1}^k X_i}{k}, \quad \sigma = \sqrt{\frac{\sum_{i=1}^k (X_i - \mu)^2}{k-1}}.$$

3.2. Time-series forecasting strategy

Our pattern based time-series forecasting strategy consists of four consecutive steps as depicted in the outlier of Fig. 1, i.e. duration series building, duration pattern recognition, duration pattern matching and duration interval forecasting. The inner three circles stand for the three basic factors concerned with activity durations, i.e. characteristics of the scientific activity itself, high level scientific workflow specifications and low level resources performance.

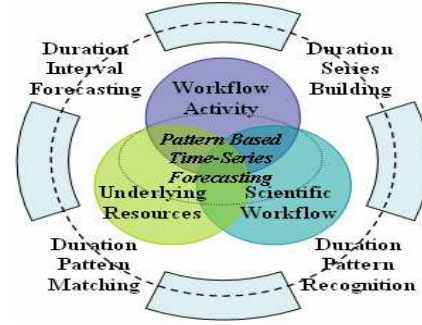


Fig. 1 Pattern based time-series forecasting strategy

Here, we illustrate the process of each step while leaving the detailed algorithms to Section 4.

1) Duration series building. In our strategy, we adopt a periodical sampling plan where the samples having their submission time belonging to the same observation unit of each period are treated as samples for the same unit. Afterwards, a representative duration series is built with the sample mean of each unit. This periodical sampling plan effectively solves the problem of limited sample size and uneven distribution, since samples are assembled to increase the density of each observation unit. Meanwhile, recurrent frequencies of duration-series segments are statistically guaranteed. Therefore, we only need to analyse the representative duration series, and if a pattern is identified for this representative duration series, it is also statistically a pattern for the entire historic duration series.

2) Duration pattern recognition. Duration In our strategy, the pattern recognition process contains two parts. The first part is to identify a potential pattern set. For this purpose, we design a novel non-linear time-

series segmentation algorithm named K-MaxSDev to achieve better performance compared with the three generic algorithms in discovering the minimum number of potential duration-series patterns which reflect the fundamental behaviour patterns of duration series. The second part is to check the validity of these segments by the minimum length threshold and specify the turning points for each valid pattern. The minimum length threshold is set to filter those segments which have not enough samples for pattern matching. Meanwhile, we emphasise the proactive identification of turning points so as to eliminate large prediction errors. In our strategy, turning points are those on the edge of two adjacent segments where the testing criterion is violated.

3) Duration pattern matching. Given the latest duration sequence, duration pattern matching is to find out the closest pattern with similarity search based on absolute differences of their sample means. However, due to the occurrences of turning points, three types of duration sequences, i.e. the type where the sequence contains internal turning points, the type where the next value of the sequence is probably a turning point and the type for the remainder of the sequences, are first identified based on their statistical features.

4) Duration interval forecasting. With the pattern matching results, duration intervals are specified accordingly with a given confidence under the assumption that the samples within a pattern follow the normal distribution. Note that in this paper, we focus on one-step-ahead prediction. But since the predicted value can also be deemed as new samples, multi-step-ahead prediction can be performed as well.

4. Algorithms for forecasting strategy

4.1. Time-series segmentation algorithm

In this section, we propose K-MaxSDev, a non-linear time-series segmentation algorithm which is designed to formulate the duration series with the minimum number of segments, i.e. potential duration-series patterns. Here, K is the initial value for equal segmentation and MaxSDev is the **Maximum Standard Deviation** specified as the testing criterion. The intuition for the initial K equal segmentation is an efficiency enhancement to the generic Bottom-Up algorithm which normally starts from the finest equal segments with the length of 2. This modification can save significant computation time at the initial stage. MaxSDev comes directly from the requirement for precise interval forecasting where the variances in each segment need to be controlled within a maximum threshold in order to guarantee an accurate small bound of the predicted durations.

K-MaxSDev is a hybrid algorithm which can be described as follows. It starts from Bottom-Up with K initial equal segmentation, followed by the repetitive

process of Sliding Windows and Top-Down with the testing criterion of MaxSDev. Here, the basic merging element of Sliding Windows is not a single point but a segment. Meanwhile, the windows can slide both forward and backward to merge as many segments as possible. As for Top-Down, it equally splits those segments which cannot be merged with any neighbours. This repetitive process stops when all individual segments cannot be merged with their neighbours any more. Evidently, the aim here is to discover the minimum number of segments. The purpose of designing K-MaxSDev rather than employing existing algorithms is to take full advantage of these three generic algorithms to achieve a better overall performance in our scenario. The pseudo-code for the K-MaxSDev algorithm is presented in Table 1 and its notations are listed in Table 2.

TABLE 1 K-MaxSDev time-series segmentation algorithm

Algorithm: K-MaxSDev	
Input: Duration series $T = \{X_1, X_2, X_3, \dots, X_n\}$, Const K , Const MaxSDev	
Output: Segmented duration series $D = \{Seg_1, Seg_2, Seg_3, \dots, Seg_m\}$	
InitialSeg(K);	//Bottom-Up
for ($j=2, j \leq K-1, j++$)	
if $SDev(Seg_{j-1}, Seg_j) \leq \text{MaxSDev}$	//Sliding Windows-Backward
$Seg_j = \text{Merge}(Seg_{j-1}, Seg_j)$;	
Delete(Seg_{j-1}); $K=K-1$; Update the index;	
end	
if $SDev(Seg_j, Seg_{j+1}) \leq \text{MaxSDev}$	//Sliding Windows-Forward
$Seg_{j+1} = \text{Merge}(Seg_j, Seg_{j+1})$;	
Delete(Seg_j); $K=K-1$; Update the index;	
end	
if $SDev(Seg_{j-1}, Seg_j) > \text{MaxSDev}$ and $SDev(Seg_j, Seg_{j+1}) > \text{MaxSDev}$	
Split(Seg_j); $K=K+1$; Update the index; //Top-Down	
end	
end	

TABLE 2 Notations used in K-MaxSDev

Seg_j	The j^{th} segment of time series
$SDev(Seg_p, Seg_q)$	The standard deviation of all the samples in the p^{th} and q^{th} segments
InitialSeg(K)	The initial K equal segmentation for time series
Split(Seg_j)	Split the j^{th} segment into two equal parts
Merge(Seg_p, Seg_q)	Merge the p^{th} and q^{th} segments
Delete(Seg_j)	Delete the j^{th} segment of time series T

The empirical function for choosing K is based on the equal segmentation tests. The candidates of K are defined to be in the form of $2i$ and should be a value of no bigger than $n/4$ where i is a natural number and n is the length of the duration series. Otherwise, K-MaxSDev is turning into Bottom-up with low efficiency. The process is that we first choose some of the candidates and perform equal segmentation. We calculate the mean for each segment and rank each candidate according to the average of the absolute differences between the means of neighbouring segments. The intuition is that initial segmentation should ensure large differences between the means of

segments, so as to guarantee the efficiency of the segmentation algorithm. The formula for the empirical function is denoted as follows:

$$Rank(k) = \frac{\sum_{i=1}^{k-1} |Mean_{i+1} - Mean_i|}{k-1} \quad (1)$$

The theoretical basis for MaxSDev is the Tchebysheff's theorem [12] which has been widely used in the statistics theory. According to the theorem, given a number d greater than or equal to 1 and a set of n samples, at least $\left[1 - \left(\frac{1}{d}\right)^2\right]$ of the samples will lie within d standard deviations of their mean, no matter what the actual probability distribution is. For example, 88.89% of the samples will fall into the interval $(\mu - 3\sigma, \mu + 3\sigma)$. The value of μ and σ can be estimated from the sample mean and sample standard deviation respectively. If it happens to be a normal distribution, the probability increases to 99.73%. Therefore, if we control the deviation to be less than $m\%$ of the mean, then $3\sigma \leq m\% \times \mu$ is ensured. We can thus specify MaxSDev by the following formula:

$$MaxSDev = \frac{m\%}{3} \times \mu \quad (2)$$

TABLE 3
Pattern validation and turning points discovery algorithm

<p>Algorithm: Pattern validation and turning points discovery Input: Segmented duration series $D = \{Seg_1, Seg_2, \dots, Seg_m\}$, Const Min_pattern_length Output: Duration patterns Pattern[] and associated turning points</p> <pre> Pattern[] = {}; for (i=1, i ≤ D.length, i++) // Checking Patterns Pattern[i] = Seg_i; Pattern[i].mean = Seg_i.mean; Pattern[i].sdev = Seg_i.sdev; if Seg_i.length < Min_pattern_length Pattern[i].valid = false; else Pattern[i].valid = true; end end for (j=1, j ≤ Pattern.length, j++) // Specifying Turning Points if Pattern[j].valid = true if Pattern[j+1].valid = true Pattern[j].turningpoint = Pattern[j+1].firstvalue else Pattern[j].turningpoint = Pattern[j+1].mean end end end end </pre>
--

With the discovered potential duration-series patterns, we further check their validity with the specified minimum length threshold defined as Min_pattern_length which should be normally larger than 3 so as to get effective statistics. Afterwards, we identify those turning points associated with each valid pattern. Specifically, turning points are defined as the points where the testing criterion of MaxSDev is violated. Since our segmentation algorithm actually ensures that the violations of MaxSDev only occur on the edge of two adjacent segments, as shown in Table 3, turning points are either specified by the mean of the

next invalid pattern or the first value of the next valid pattern. Evidently, this specification of turning points captures the locations where large prediction errors mostly tend to happen.

4.2. Algorithm for pattern matching and interval forecasting

As we mentioned in Section 3, the most important issue for pattern matching and interval forecasting is to identify the type of the latest duration sequence. Specifically, we define three types of duration sequences, i.e. the type where the sequence contains internal turning points, the type where the next value of the sequence is probably a turning point and the type for the remainder of the sequences.

TABLE 4
Pattern matching and duration interval forecasting algorithm

<p>Algorithm: Pattern matching and duration interval forecasting Input: Latest duration sequence DS, Duration patterns Pattern[], Const ms=MaxSDev, Const Min_pattern_length, Const λ (α% confidence percentile) Output: Matched pattern MP and duration interval DI(min, max)</p> <pre> if DS.sdev ≥ ms // The latest duration sequence contains turning point in itself While (DS.sdev ≥ ms) // remove the part before and include the turning point DS = DS - DS.firstvalue; end DI(min, max) = (DS.mean - λ * ms, DS.mean + λ * ms); else // λ is the α% confidence percentile for normal distribution if DS.length ≥ Min_pattern_length // ensure a valid pattern matching MP = Min(Abs(Pattern.mean - DS.mean)) // Min() returns the pattern with minimum absolute difference of mean if MP.sdev < DS.sdev < ms // The next value is highly possible to be a turning point DI(min, max) = (MP.turningpoint - λ * ms, MP.turningpoint + λ * ms); else if DS.sdev ≤ MP.sdev DI(min, max) = (MP.mean - λ * MP.sdev, MP.mean + λ * MP.sdev) end else // Sample size is too small to match a valid pattern DI(min, max) = (DS.mean - λ * ms, DS.mean + λ * ms); end end </pre>

As presented in Table 4, the types of sequences are identified based on standard deviations. For the first type of duration sequence which has a standard deviation larger than MaxSDev, it cannot be matched with any valid patterns and must contain at least one turning point. Therefore, we need to first locate the turning points and then conduct pattern matching with the remainder of the duration sequence. However, to ensure the effectiveness of the prediction, if the length of the remainder duration sequence is smaller than the minimum pattern length, the duration interval is specified with the mean of the duration sequence while its standard deviation is substituted by MaxSDev. Note that the matched pattern is defined as the valid pattern of which the statistical mean has the minimum absolute difference with that of the latest duration sequence. As for the second type, it is identified when the standard deviation of the latest duration sequence is larger than that of the matched pattern but smaller than MaxSDev. In this case, the next value of the sequence will probably be a turning point since it is on the edge of two different patterns according to our pattern recognition process. Therefore, the mean of the next value is specified with the associated turning

point of its matched pattern and the standard deviation is specified with MaxSDev. For the third type of sequence where its standard deviation is smaller than its matched pattern, the next value is predicted with the statistical features of the matched patterns as a refinement to that of the latest duration sequence. As presented in Table 4, we illustrate the specification of duration intervals with normal distribution which is a kind of symmetric probability distribution. For example, given λ which is the $\alpha\%$ confidence percentile for normal distribution, the predicted $\alpha\%$ confidence interval is $(\mu - \lambda\sigma, \mu + \lambda\sigma)$, where μ and σ stands for the statistical sample mean and standard deviation respectively. However, if the samples of a pattern follow non-normal distribution models, such as Gamma distribution, lognormal distribution or Beta distribution, changes can be made accordingly [10].

5. Evaluation

5.1. Simulation environment

SwinDeW-G is a peer-to-peer based scientific workflow system running on the SwinGrid (Swinburne service Grid) platform [16]. An overall picture of SwinGrid is depicted in Fig. 2 (bottom plane) which contains many grid nodes distributed in different places. Each grid node contains many computers including high performance PCs and/or supercomputers composed of significant number of computing units. The primary hosting nodes include the Swinburne CITR (Centre for Information Technology Research) Node, Swinburne ESR (Enterprise Systems Research laboratory) Node, Swinburne Astrophysics Supercomputer Node, and Beihang CROWN (China R&D environment Over Wide-area Network) Node in China. They are running Linux, GT4 (Globus Toolkit) or CROWN grid toolkit 2.5 where CROWN is an extension of GT4 with more middleware, hence compatible with GT4. Currently, SwinDeW-G is deployed at all primary hosting nodes as exemplified in Fig. 2 (top plane). In SwinDeW-G, a scientific workflow is executed by different peers that may be distributed at different grid nodes. As shown in Fig. 2, each grid node can have a number of peers, and each peer can be simply viewed as a grid service.

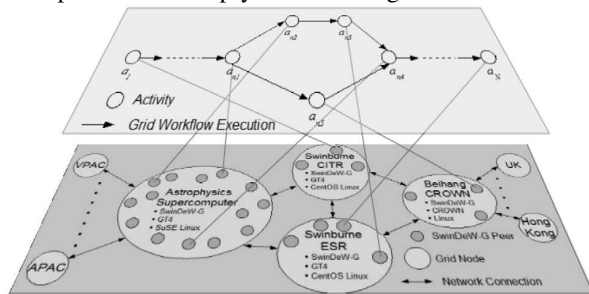
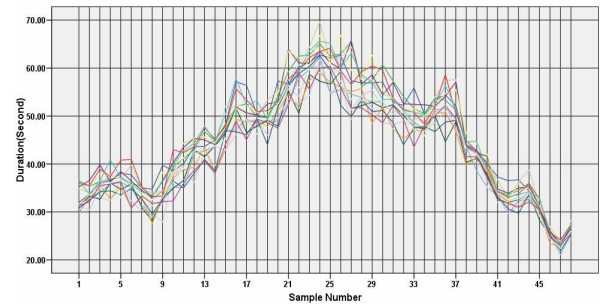


Fig. 2 Overview of SwinDeW-G environment

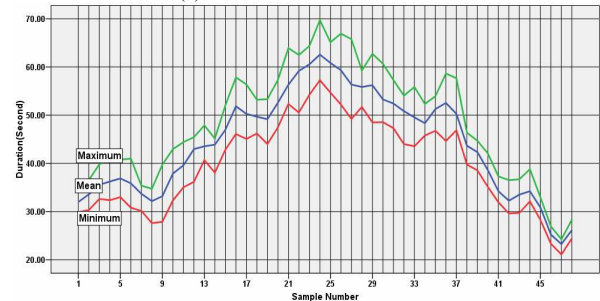
5.2. Simulation experiment and results

The activity we analysed here is for solving complex thermodynamic equations which occurs frequently on a daily basis within many scientific workflow instances and simulation experiments for meteorological research. This activity is quite representative since it demands high performance computing resources for computation as well as network and I/O facilities for data collection and data transfer. The simulation data and related materials could be found online at www.swinflow.org/docs/TimeSeries.zip. Now, we demonstrate the simulation experiment with each step addressed in Section 3.2.

1) Duration series building. The first step is to build the duration series. Here, we set the basic observation time unit as 15 minutes and the overall observation window as a typical working day of (8:00am~8:00pm). As depicted in Fig. 3(a), with the period of a day, 15 duration series are built by random sampling without replacement in each observation unit. Here, null values are replaced by the unit sample means plus white noise [2], i.e. a random number with the mean of 0 and standard deviation of 1. Eventually, the representative duration series is built by the composition of the sample mean in each observation unit sequentially through the time. As depicted in Fig. 3(b), the representative duration series which is to be further analysed lies between the upper and the lower duration series which are composed of the maximum duration and minimum duration of each observation unit respectively. In Fig. 3, the vertical axis stands for the activity durations with the basic time unit of second.



(a) 15 historical duration series



(b) The representative duration series

Fig. 3 Building duration series

2) Duration pattern recognition. In this step, we first conduct the K-MaxSDev time-series segmentation algorithm to discover the potential pattern set. Here, we need to assign the value for two parameters, i.e. K and MaxSDev. Based on our empirical function, we choose the candidates of 2, 4, 8, and 12. Based on Formula (1) in Section 4, 4 is ranked the best. As for MaxSDev, based on Formula (2) with the overall sample mean μ as 44.11 and having the candidates of m as 15, 20, 25, and 30, they are specified as 2.21, 2.94, 3.68 and 4.41 respectively. We conduct K-MaxSDev and compare with the three generic segmentation algorithms. With different MaxSDev, the number of segments for the four rounds of test is presented in Fig. 4. Evidently, with 11, 8, 7 and 5 as the minimum number of segments in each round of test, K-MaxSDev achieves the best performance in terms of discovering the smallest potential pattern set.

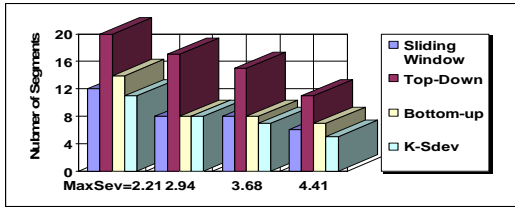


Fig. 4 Comparison with three generic segmentation algorithms

Moreover, with MaxSDev as 2.21 and Min_pattern_length as 3 which is the lowest bound, the result for segmentation and pattern validation is presented as an example in Table 5 where 8 valid patterns are identified from a total of 11 segments. The valid patterns and their associated turning points are listed in Table 6. Note that since our observation period is a whole working day, the turning point of Pattern 8 is actually defined as the first value of Pattern 1 to make it complete as a cycle. Here, we also trace back the number of concurrent activities according to the observation units covered by each pattern. The result shows that when the average number of concurrent activities increases with the schedule of working hours, the sample means of activity durations also follow an

increasing trend. However, they are evidently not in a linear relationship. Therefore, this observation indeed supports our motivation.

3) Duration pattern matching and 4) Duration interval forecasting. Since these two steps are highly correlated, we demonstrate them together. Here, we randomly select 30 duration sequences from the system logs where the lengths range from 4 to 6 and the last value of each sequence is chosen as the target to be predicted. In our experiment, we adopt the normal distribution model to specify the duration intervals with a high confidence of 95% which denotes the corresponding confidence percentile to be 1.96 [12]. As shown in Fig. 5, for 25 cases, the target value lies within the predicted duration intervals. As for the rest of the 5 cases, the target values deviate only slightly from the intervals. Therefore, the performance of interval forecasting is effective.

TABLE 5 Results for segmentation and pattern validation

Segments	Segments Description			Covered Observation Time Units	Pattern Validation
	Mean	SDev	Length		
Segment1	34.7	2.05	10	(8:00am-10:30am)	Valid
Segment2	39.58	0	1	(10:30am-10:45am)	Invalid
Segment3	44.47	0.47	3	(10:45am-11:30am)	Valid
Segment4	50.10	2.00	6	(11:30am-1:00pm)	Valid
Segment5	59.81	2.10	6	(1:00pm-2:30pm)	Valid
Segment6	55.43	1.45	4	(2:30pm-3:30pm)	Valid
Segment7	50.75	1.51	7	(3:30pm-5:15pm)	Valid
Segment8	43.00	0.94	2	(5:15pm-5:45pm)	Invalid
Segment9	38.55	0	1	(5:45pm-6:00pm)	Invalid
Segment10	33.01	1.45	5	(6:00pm-7:15pm)	Valid
Segment11	24.88	1.49	3	(7:15pm-8:00pm)	Valid

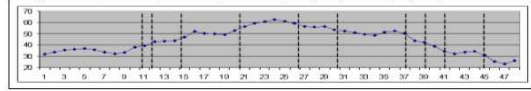


TABLE 6 Pattern description and associated turning points

Patterns	Patterns Description			Concurrent Activities
	Mean	SDev	Turning Points	
Pattern 1	34.7	2.05	39.58	(2, 4)
Pattern 2	44.47	0.47	46.98	(3, 6)
Pattern 3	50.10	2.00	56.33	(4, 8)
Pattern 4	59.81	2.10	56.35	(7, 11)
Pattern 5	55.43	1.45	52.42	(6, 10)
Pattern 6	50.75	1.51	43.00	(4, 9)
Pattern 7	33.01	1.45	25.22	(2, 5)
Pattern 8	24.88	1.49	31.99	(1, 3)

Furthermore, two intuitive yet general methods of MEAN and LAST [6][10] which adopt the sample mean or the last value of the latest sequences respectively as the predicted value are implemented for comparison. We compare them with our predicted means since they can

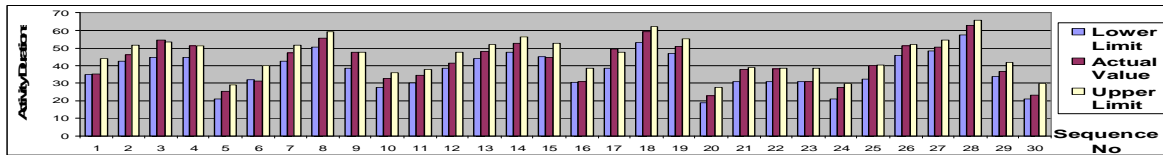


Fig. 5 Predicted duration intervals

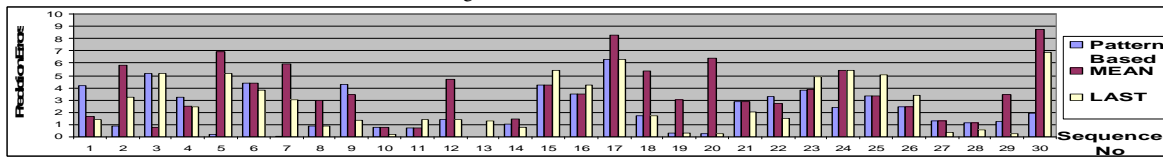


Fig. 6 Comparison of prediction errors (individual cases)

only predict point values. Here, the prediction error is defined as the absolute difference between the predicted mean and the target value. As seen from Fig. 6, our pattern based strategy behaves better than MEAN and LAST for most individual cases, especially for the sequences such as numbers 2, 5 and 7 where the target value is a turning point and the sequences such as numbers 18, 20 and 30 where the sequences contain turning points in themselves. To get an overall view of the performance, we also depict the sum of errors for each of 5 cases in Fig. 7. It evidently shows that our strategy behaves more stable with smaller errors.

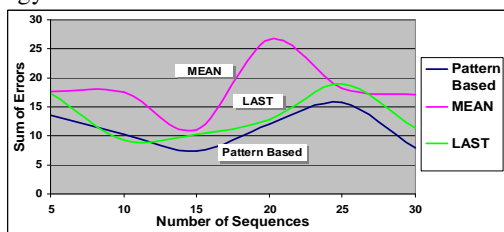


Fig. 7 Comparison of prediction errors (overall view)

6. Conclusion and future work

The performance of conventional linear time-series models can be seriously deteriorated due to the problems limited sample size and frequent turning points. To address such problems, in this paper, a novel pattern based time-series forecasting strategy has been proposed. Our strategy can adapt to small sample size with a periodical sampling plan and handle turning points with an effective process of pattern recognition and pattern matching. Specifically, the K-MaxSDev time-series segmentation algorithm has been presented to discover the minimum number of potential patterns. Concerning the occurrences of turning points, three types of duration sequences were identified and specified with different pattern matching results. With that, accurate duration intervals can be predicted. The simulation experiment has demonstrated that our K-MaxSDev segmentation algorithm performs better than three generic algorithms, i.e. Sliding Windows, Top-Down and Bottom-Up, in terms of discovering the smallest potential pattern set. Furthermore, the prediction results have shown that our strategy can effectively handle the occurrences of turning points and specify high confidence duration intervals with relatively smaller errors than the intuitive yet practical methods of MEAN and LAST.

In the future, we will further investigate time-series patterns in more scientific workflow scenarios, such as workflow scheduling and temporal verification.

ACKNOWLEDGEMENT

The research work reported in this paper is partly supported by Australian Research Council under DP0663841 and LP0669660, Swinburne Dean's

Collaborative Grant Scheme 2007-2008, and Swinburne Research Development Scheme 2008.

REFERENCES

- [1] S. Akioka and Y. Muraoka, "Extended Forecast of CPU and Network Load on Computation Grid", *Proc. of 2004 IEEE International Symposium on Cluster Computing and the Grid*, pp.765-772, 2004.
- [2] C. Chatfield, *The Analysis of Time Series: An Introduction* (Sixth Edition), Chapman and Hall/CRC, 2004.
- [3] J. Chen and Y. Yang, "Adaptive Selection of Necessary and Sufficient Checkpoints for Dynamic Verification of Temporal Constraints in Grid Workflow Systems". *ACM Trans. on Autonomous and Adaptive Systems*, Vol. 2, Issue 2, Article 6, June 2007.
- [4] J. Chen and Y. Yang, "Temporal Dependency based Checkpoint Selection for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems", *Proc. of 30th International Conference on Software Engineering (ICSE2008)*, pp. 141-150, Leipzig, Germany, May 2008.
- [5] F. L. Chung, T. C. Fu, V. Ng and W. P. Luk, "An Evolutionary Approach to Pattern-Based Time Series Segmentation", *IEEE Trans. on Evolutionary Computation*, Vol. 8, pp.471-488, 2004.
- [6] P. A. Dinda and D. R. O'Hallaron, "Host Load Prediction Using Linear Models", *Cluster Computing*, Vol. 3, pp. 265-280, 2000.
- [7] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques* (Second Edition), Elsevier, 2006.
- [8] E. Keogh, S. Chu, D. Hart and M. Pazzani, "An Online Algorithm for Segmenting Time Series", *Proc. of 2001 IEEE International Conference on Data Mining*, pp. 289-296, 2001.
- [9] K. H. Kim, R. Buyya and J. Kim, "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters", *Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, pp.541-548, 2007.
- [10] A. M. Law and W. D. Kelton, *Simulation Modelling and Analysis* (Fourth Edition), McGraw-Hill, 2007.
- [11] R. Prodan and T. Fahringer, "Overhead Analysis of Scientific Workflows in Grid Environments", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 19, No. 3, pp. 378-393, Mar. 2008.
- [12] K. A. Stroud, *Engineering Mathematics* (Sixth Edition), Palgrave Macmillan, New York, 2007.
- [13] W. Smith, I. Foster and V. Taylor, "Predicting Application Run Times with Historic Information", *Journal of Parallel and Distributed Computing*, Vol.64, No.9, pp.1007-1016, Sept. 2004.
- [14] S. Tirthapura, B. J. Xu and C. Busch, "Sketching asynchronous streams over a sliding window", *Proc. of the 25th symposium on Principles of distributed computing*, pp. 82-91, 2006.
- [15] Y. W. Wu, Y. L. Yuan, G. W. Yang and W. M. Zheng, "Load prediction Using Hybrid Model for Computational Grid", *Proc. of 2007 International Conference on Grid Computing*, pp. 235 – 242, Sept. 2007.
- [16] Y. Yang, K. Liu, J. Chen, J. Lignier and H. Jin, "Peer-to-Peer Based Grid Workflow Runtime Environment of SwinDeW-G", *Proc. of the 3rd International Conference on e-Science and Grid Computing (e-Science07)*, pp. 51-58, Bangalore, Dec. 2007.
- [17] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", *Journal of Grid Computing*, No. 3, pp. 171-200, Sept. 2005.
- [18] L. Y. Yang, I. Foster and J. M. Schopf, "Homeostatic and Tendency-based CPU load Predictions", *Proc. of 2003 International Parallel and Distributed Processing Symposium (IPDPS'03)*, pp. 9-17, Apr. 2003.
- [19] Y. Y. Zhang, W. Sun and Y. Inoguchi, "Predicting Running Time of Grid Tasks based on CPU Load Predictions", *Proc. of 2006 IEEE International Conference on Grid Computing*, pp. 286-292, Sept. 2006.