Supply chain management by means of FLM-rules

Nicolas Le Normand, Julien Boissière, Nicolas Méger, Lionel Valet

LISTIC Laboratory - Polytech'Savoie
Université de Savoie
B.P. 80439 — F-74944 Annecy-Le-Vieux, France.
{Nicolas.Le-Normand|Julien.Boissiere|Nicolas.Meger|Lionel.Valet}@univ-savoie.fr

Abstract. Efficient supply chain management is nowadays considered as a major source of competitiveness. In this paper, we aim at managing a supply chain by forecasting problems that might arise in the future. To achieve this goal, we propose to model our supply chain using $FLM-rules$ and to use such a model for forecasting problems. This approach is tested using synthesized data that originate from a supply chain simulation. Results are encouraging as 93% of the problems are predicted.

Key words: episode rules, FLM-rules, forecast problems, supply chain.

1 Introduction

Information technology tools such as ERPs (Enterprise Resource Planning) allow firms to gather huge quantities of data that originate from several business processes. For example, when dealing with supply chains, dataflows about inventories levels, orders, and deliveries can be collected for different facilities over large periods of time. If a model is inducted from this data, then it becomes possible to forecast problems that may arise in the future. Symeonidis *et al* [1] uses such an approach to enable an agent to anticipate bidding strategy in a supply chain. See also Chen *et al* [2] for a more general framework. In this paper, we propose to forecast problems in supply chains by handling the whole data as a single time stamped event sequence. As operations strongly relate to each other, accumulated data can be described using temporal dependencies, such as *episodes rules*, as defined and proposed in [3]. Finding episodes rules may provide interesting insight to experts in various domains. In particular, it has been shown to be very useful for alarm log analysis in the context of the TASA project [4]. As episodes rules extraction gives many rules, it has been proposed in [4] to browse them interactively so that the user focuses on episodes rules that are interesting with respect to his/her knowledge. In this paper, we intend to decrease the number of episode rules by selecting conclusions. We also aim at bringing additional temporal information by extracting $FLM-rules$ [5]. This information will be further used for calculating forecast windows. Furthermore, $FLM-rules$ can be extracted under a *maximum gap* constraint (maximum time

interval between each event type occurrence), which allows us to consider rules that make sense with respect to the context. More details about $FLM-rules$ are to be found in Section 2. Once extracted, $FLM-rules$ can be used as a model for forecasting problems. Section 3 details our forecast method as well as its evaluation principle. This evaluation is done in Section 4 using two synthesized datasets: a learning one and test one. Finally, Section 5 draws conclusions and perspectives.

2 FLM-rules

Let E be a set of event types. An event is then defined as a pair (e, t) where $e \in E$ is an event type and t is an integer, i.e. a timestamp that indicates the occurrence time of e. An *event sequence* S is a triple (s, T_s, T_e), where s is an ordered sequence of events $\langle (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) \rangle$ such that $\forall i \in \{1, \ldots, n\}$, $e_i \in E$ and such that $\forall i \in \{1, \ldots, n-1\}, T_s \leq t_i \leq t_{i+1} \leq T_e$. T_s and T_e are integers that respectively denote the starting time and the ending time of the event sequence. Let us consider the following toy example $X = (Seq, 0, 456)$ with $Seq = \langle (A, 45), (D, 79), (B, 82), (C, 101), (A, 120), (B, 150), (C, 176), (A, 200), (B, 420) \rangle$. X starts at time unit 0 and ends at time unit 456. Events types A and B occur three times, event type C occurs twice and event type D only occurs once. This toy example will be used all over this section. From such an event sequence, one can extract *episodes rules*. Those rules rely on *episodes*. We here consider a specific class of episodes, namely the *serial episodes*. A *serial episode* is a tuple α such that $\alpha = \langle e_1, e_2, \ldots, e_k \rangle$ with $e_i \in E$ for all $i \in \{1, \ldots, k\}$. In this paper, we will use the notation $e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_k$ to denote the serial episode $\langle e_1, e_2, ..., e_k \rangle$ where '$\rightarrow$' may be read as 'followed by'. For example, a serial episode is $A \rightarrow B \rightarrow C$. This can be read as event type A is observed, then, sometimes later, event type B occurs and, finally, event type C is observed later on. For simplicity reasons, we will now refer to *episodes* instead of referring to *serial episodes*. The $suffix$ of an episode is the last event type of the episode and the other events types are termed as the $prefix$. For example, $suffix(A \rightarrow B \rightarrow C) = C$ and $prefix(A \rightarrow B \rightarrow C) = A \rightarrow B$. When looking for occurrences of episodes, a maximum time gap constraint can be set between each symbol occurrence for reducing the search space and for selecting occurrences that make sense with respect to the application. This constraint is termed as $gapmax$. Let α and β be episodes such that $prefix(\beta) = \alpha$. An *episode rule* is the expression $\alpha \Rightarrow suffix(\beta)$. For example, if we consider episode $\alpha = A \rightarrow B$ and episode $\beta = A \rightarrow B \rightarrow C$, then we can generate episode rule $A \rightarrow B \Rightarrow C$. This can be read as "if event A occurs, and if, sometimes later, event B also occurs, then, event C might occur later on". Two measures can be linked to episode rules :

- the *support* measure, i.e. the number of occurrences of an episode rule throughout the whole event sequence,
- the *confidence* measure, i.e. the observed conditional probability of having the conclusion of an episode rule knowing that the premises already occurred.

Those measures rely on the definition of occurrences. We here use *minimal occurrences* as formally defined in [3]. Basically, a minimal occurrence of an episode α is a time interval $[t_{os}, t_{oe}]$ such that the first event type of α occurs at time t_{os} and such that $suffix(\alpha)$ occurs at time t_{oe}. The other event types of α have to occur in between. On the top of that, $[t_{os}, t_{oe}]$ is a minimal occurrence if, and only if, there is no other occurrence of α $[t'_{os}, t'_{oe}]$ such that $[t'_{os}, t'_{oe}] \subset [t_{os}, t_{oe}]$. That is, a minimal occurrence of a given episode can not spread over another occurrence of the same episode. Back to our example, the minimal occurrences of $A \rightarrow B$ in X are $mo(A \rightarrow B, X) = \{[45, 82], [120, 150], [200, 420]\}$. On the other hand, intervals $[45, 150], [45, 420]$, and $[120, 420]$ are not minimal occurrences of $A \rightarrow B$ as they spread over intervals that already contain $A \rightarrow B$.

Support and confidence measures are used for selecting episode rules according to a minimum support threshold σ and a minimum confidence threshold γ. That is, we can restrict our search to $frequent$ episodes rule, i.e. episode rules whose support is above σ, and/or to $confident$ episode rules, i.e episode rules whose confidence is above γ. Algorithms (e.g. [4], [5]) that aim at extracting episode rules all rely on the anti-monotonicity of the support measure for pruning the search space. Looking at our example, if we consider the episode rule $A \rightarrow B \Rightarrow C$, we can observe that, using minimal occurrences, its premise occurs three times in the event sequence while event type C only occurs twice after an occurrence of episode $A \rightarrow B$. Thus, the support of episode rule $A \rightarrow B \Rightarrow C$ is 2 and its confidence is $2/3 = 66,67\%$. Those measures can be refined by redefining them for all possible *widths*, i.e. the time span of all possible time windows that sart at the occurrence date of the first event type of episode occurrences. Therefore, for a given episode rule, it is possible to compute its support and its confidence with respect to various window widths and thus to its occurrences widths. $FLM - rules$ are frequent and confident episode rules having a confidence local maximum value lm at given width wm such that there exists a greater width for which confidence value is lesser that $lm - (dr \times lm/100)$, with dr a decrease rate set by the user. The higher is dr decrease rate, the higher is lm compared to confidence values defined for greater widths. wm is said to be the *optimal window size* of the episode rule. Thus, each $FLM - rule$ comes along with its optimal window size. For formal definitions, we refer the reader to [5]. For example, if $\sigma = 2$, $\gamma = 100\%$ and $dr = 10\%$, then rule $A \rightarrow B \Rightarrow C$ is selected and it comes along with an optimal window size of 56 minutes. For this optimal window size, its confidence is 100% and its support is 2. This can be interpreted as follows : "this rule is very confident for a width of 56 minutes. For this width, its confidence is 100% and it is observed twice in sequence X. Its confidence, for other widths, is below 100% or, if there are other windows for which confidence is greater or equal to 100% and for which the rule is frequent, then those window size are above 56 minutes. There exist a width above 56 minutes such that its confidence is lower than $100 - (10 \times 100/100) = 90\%$". Indeed, if we consider a width of 220 times units, then, in X, we can consider all occurrences of this episode rules. In this case, the support for the episode $A \rightarrow B$ rises up to 3. On the other hand, the confidence falls down to $2/3 = 66,67\%$. Due to space

limitations, we did not formally define all those concepts. Once again, we refer the reader to [5] for more formal definitions.

3 Forecasting problems

This section first details our forecast method in Subsection 3.1 before expounding its evaluation principle in Subsection 3.2.

3.1 Forecast method

We propose to describe the supply chain behavior by extracting $FLM-rules$ from a learning dataset. We will use this model for forecasting future problems. First, episodes rules are filtered on conclusions that relate to user-selected problem classes. Then, for each problem class, we aim at defining a *forecast window*. To do so, for each $FLM-rule\ r$ concluding on a user-specified problem class, we establish an *observation window* $]t^r, t_0]$ such that $t^r + k \times gapmax = t_0$ where t_0 is the date at which our forecasting method is triggered, $gapmax$ is the maximum gap constraint and k is the number of event types of the premise. Let Ts^r be the set of the occurrence dates of the first event of the occurrences of premisses of the rule r that occur in $]t^r, t_0]$. We select $ts^r_x \in Ts^r$ such that $\nexists\ ts^r_y \in Ts^r$ with $ts^r_x \neq ts^r_y$ and $ts^r_y > ts^r_x$ and such that no conclusion of rule r appear in $]ts^r_x, t_0]$. Then the conclusion of rule r should appear at $tc^r = ts^r_x + wm$ with $tc^r > t_0$ and wm the optimal window size of rule r. Let $Tc^C = \{tc^r \mid the\ conclusion\ of\ rule\ r\ belongs\ to\ class\ C\}$. Let $[tfs^C, tfe^C]$ be the forecast window associated to class C. Then, $tfs^C = min(T^C_c)$ and $tfe^C = max(T^C_c)$. In other words, for each problem class, and if there are problems that are about to occur, we generate a warning that comes along with a forecast window. Figure 1 summarizes this method for problems of class C.

3.2 Evaluation principle

We intend to evaluate our forecast method by using it at different dates t_0 on a test dataset. The main assumption is that the system, i.e. the supply chain in our case, exactly behaves the same in both the learning dataset and the test dataset. Two cases are to be considered:

- Case 1: our forecast method foresees a problem class C in the supply chain. It thus gives the forecast window $[tfs^C, tfe^C]$. In this case, we will check whether problems belonging to C occur in the forecast window that has been given.
- Case 2: no warning is provided by the system. In this case, as the system foresees nothing, we have no window to check. Nevertheless, it is necessary to make sure that there are no problem later on in the supply chain. To

do this, we check whether problems occur in $]t_0, t_0 + gapmax]$. Indeed, the last event type of the premise of rule can be detected at t_0. Therefore, the conclusion, as rules were extracted under a maximum gap constraint, can appear until $t_0 + gapmax$.

4 Experimental evaluation

In this section, we first give a brief overview of the supply chain simulation (cf. Subsection 4.1). Then, in Subsection 4.2, we discuss the various parameters that we set for extracting for $FLM - rules$ before describing extraction results in Subsection 4.3. Finally, Subsection 4.4 goes into evaluation details.

4.1 Supply chain simulation

The simulated supply chain is a distribution one (divergent tree). It is composed of a warehouse D_0 which delivers three distribution facilities denoted D_1, D_2, D_3. Finally, nine aggregated customers are connected to the distribution facilities. Each of them represents a consumers family that basically purchase the same things. The replenishment policies used at each facility are reorder point policies: when the inventory of one product drops below a certain level (the reorder point), a fixed quantity is ordered to the supplier. This supply chain distributes ten different products denoted p_i with $i \in [1 \dots 10]$. End customers demands can be seen, for each customer as: the interval between demands, which follows a Normal distribution, and the demand quantity for each product which follows a Poisson distribution. There are no production delays in the system, but transportation ones: one to supply the warehouse from an external source, and one from the warehouse to each of the distribution facilities. Several indicators are used along this supply chain. First there are inventories levels for each product at each facility. We note $inv_{i,j,k}$, the inventory level k of product i at facility j such that $j \in [0 \dots 3]$ and $k \in \{'low', 'medium', 'large'\}$. Inventories are low when the level is below the safety stock, $medium$ if below the reorder point and $large$ if over. Then, every placed customer order generates a satisfactory factor denoted $sat_{b,l}$ which reflects the satisfaction level l of the customer b according to the expected and effective delivery dates such that $l \in \{'ontime', 'late'\}$ and $b \in [1 \dots 9]$. If the facility owns enough inventory, or if the demand can be fulfilled within 2 days, then the customer demand is satisfied ($'ontime'$), while, if he must wait longer, he is unsatisfied ($'late'$). In any case, orders are delivered. In order to efficiently manage our supply chain, we will focus on the satisfaction of end customers. Indeed, it is interesting for decision-makers to know in advance the customers who might be unsatisfied in the future in order to limit the number of unsatisfied customers. Thus, each event type $sat_{b,l}$ is considered as a problem class that has to be forecasted along with its forecast window.

This supply chain was modeled using a specialized software for industrial flow simulation. We generated two datasets: the learning one and the test one. Each

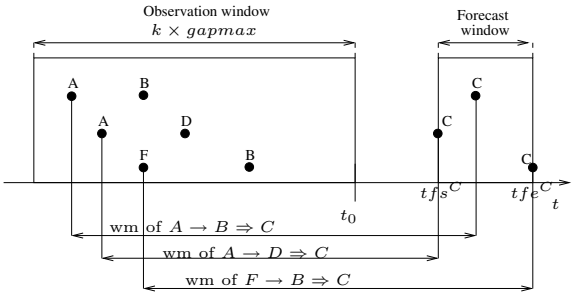

Fig. 1. Forecasting problem class C.

dataset is a one year simulation. Global demand is quite regular throughout the year though three customers have seasonal demand, i.e. they ask for twice as much product during the summer. This seasonality generates interesting fluctuations in the supply chain. The basic time unit of the system is the hour. This means that demands may occur every hour. This time unit is important as it points out the dynamics of the system. The simulation generates about 300,000 events a year. Event types are defined over a 250 symbols alphabet. The three aggregated customers who realize many orders in our simulation are customers 1 to 3 and parameters are set to be quite often unsatisfied. Aggregated customers 4 to 9 realize less orders and are always satisfied in the simulation.

4.2 Extraction parameters

The choice of the various parameters is done according to several experiments. First, we set the minimum confidence threshold to 0.9 as we want quite strong rules. Next, according to the our functional expert, we set $gapmax$ to $10,000$ minutes. Then, in order to have enough rules for all dissatisfied customers, we set the minimum support threshold to 100 which is a very low support. Furthermore, so as to obtain quite generic rules and to avoid over-fitting, we set the maximum number of symbols per rule to 3. It is also useful for making extraction tasks tractable as the minimum support is very low. Last parameter is the decrease rate. The higher is the decrease rate, the lower is the value of the confidence for at least one width that is larger than the optimal window size. In our case, we had to set this value to 0 because we were not able to extract $FLM-rules$ with decrease rates higher than 2%. And if we set the decrease rate to 2%, then we have only 12 rules. This means that confidence generally increases up to a given level of confidence that corresponds to a width such that the rule is frequent and confident enough and such that the confidence for the following width is the same. Besides, we observed that, at most, the confidence increases no more than 4% (average for all customers related rules) and that the support goes up no more than 8% (average for all customers related rules) for widths that are larger than the optimal window size. In the end, this also justifies the use of

the optimal window sizes with a decrease rate set to 0% for establishing forecast windows.

4.3 Extraction results

We extracted rules from our dataset using the Winminer prototype [5] on a Suse Linux platform (2.6.22.5-31-default i686 kernel, 512 MB RAM, Athlon 64 3000+ 1.8 GHz). Within nine hours, around $3,000,000$ $FLM-rules$ are extracted. $FLM-rules$ are numerous in this case because (1) support is very low and (2), by setting the decrease rate to $O\%$, we extract as many FLM-rules as standard episode rules. Once filtered with respect to dissatisfied customers, the number of $FLM-rules$ falls down to $37,282$. The distribution of these rules matches the simulation frequency of dissatisfaction of the three customers that can suffer delivery problems. Thus we have $36,877$ rules for customer 1, 226 rules for customer 2 and 179 rules for customer 3. This demonstrates that we had to set a quite low support, otherwise, it would not have been possible to get rules for customer 1 and customer 2. About the optimal windows size, it is also necessary to separately consider the various dissatisfied customers: for customer 1, the average optimal window size lm is 1070 minutes, for the customer 2 average lm is 6391 minutes and for customer 3 average lm is 2434 minutes. This also matches our simulation as customers 1 and 3 generally place order more frequently than customer 2.

4.4 Evaluation results

We carried out a series of measurements for different dates t_0 using our test data set. We made one forecast per week, i.e. 52 forecasts. We then evaluated our forecasts as explained in Subsection 3.1. Only three customers present dissatisfactions in our sequence. It is thus a question of foreseeing correctly at which moment they are going to be unsatisfied. Results of the evaluation are given by Figure 2. For each customer b, we denote by $\widehat{late}$, the number of forecasts stating that customer b is about to be unsatisfied, i.e. rules ending on $sat_{b,late}$ are likely to occur. $\widehat{ontime}$ denotes the number of forecasts that foresee no problem. As you can see, for customer 1, we are able to predict 90% of the problems without having false alarms though this type of customer is always dissatisfied. For customer 2, we predicted 88% of the problems, but this time, we have 4 false alarms. And finally, for customer 3, we did predict 96% of the problems and we just have 2 false alarms. About the forecast windows, average starting and ending dates are: $t_o + 6$ hours and $t_0 + 54$ hours for customer 1, $t_0 + 16$ hours and $t_0 + 208$ hours for customer 2, $t_0 + 15$ hours and $t_0 + 87$ hours for customer 3. Thus, end-users are given enough time to trigger corrective actions in order to deal with those future problems. Finally, if we average all the data for all customers, we are able to forecast 93% of the problems.

customer 1	$\widehat{late}$	$\widehat{ontime}$
late	47	5
ontime	0	0

customer 2	$\widehat{late}$	$\widehat{ontime}$
late	32	2
ontime	4	14

customer 3	$\widehat{late}$	$\widehat{ontime}$
late	16	0
ontime	2	34

Fig. 2. Evaluation results.

5 Conclusions and perspectives

In this paper, we propose to rely on $FLM-rules$ for building a model of a system that can be further used for forecasting events and their corresponding forecast windows. We tested our forecast method by simulating a supply chain. Results are encouraging. Indeed, we are able to predict 93% of the problems in the supply chain without having more than 4 false alarms (out of 52 predictions). This also shows that local patterns can be used for both modelling and forecasting, without asking the user to browse huge collections of patterns. Future work directions include the generation of a mathematical model of the system from episode rules.

References

1. A. L. Symeonidis, D. D. Kehagias, and P. A. Mitkas. Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques. *Expert Systems with Applications*, 25:589–602, 2003.
2. M.-C. Chen, C.-L. Huang, K.-Y. Chen, and H.-P. Wu. Aggregation of orders in distribution centers using data mining. *Expert Systems with Applications*, pages 453–460, 2005.
3. H. Mannila, H. Toivonen, and A. Verkamo. Discovery of frequent episodes in event sequences. *Data mining and Knowledge Discovery*, pages 259–298, 1997.
4. M. Klemettinen, H. Mannila, and H. Toivonen. Interactive exploration of interesting findings in the telecommunication network alarm sequence analyser tasa. *Information and Software Technology*, 1999.
5. N. Meger and C. Rigotti. Constraint-based mining of episodes rules and optimal window sizes. *Principle and practice of Knowledge Discovery in Databases*, pages 313–324, 2004.