Definition and mining of quasi-cyclical patterns in agroclimatic data

Geise K. S. Santos, Tércio A. Santos Filho, Marcos A. Batista and Sérgio F. da Silva
geise.kss@gmail.com, tercioas@gmail.com, marcos.batista@pq.cnpq.br, sergio@ufg.br
IBiotec, Federal University of Goiás, Catalão, GO, Brasil
Av. Dr. Lamartine Pinto de Avelar, 1120, CEP 75704-020

Abstract— *Mining patterns in the association rules form are important to extract knowledge in multidimensional datasets. Several patterns occur in an approximately cyclical (quasi-cyclical) form in the nature and the variations of these cycles hinders the application of the traditional pattern mining techniques. The literature methods identify only common cycles. In this work, we presents the formal definition of quasi-cyclical pattern concept and we develop a method for cyclical and quasi-cyclical patterns mining. We conducted experiments using quantitative temporal databases to demonstrate that our temporal rules mining technique achieve satisfactory results.*

Keywords: Cyclical patterns, quasi-cyclical patterns, association rules

1. Introduction

Currently, there are several techniques that discover association rules among items set according to the past transactions. Association rules describe important relationships among items or variables in databases. For example, the purchase history mining from a supermarket can extract a rule that all people who buy meat also buy beer, and then we have the rule "meat $\rightarrow$ beer".

The extracted rules occur with a certain frequency, this is a cyclic variation over time. About the example cited before, the rule "meat $\rightarrow$ beer" can be observed, for example, on Saturdays during the interval 2pm-4pm. Then the people who buy meat on Saturday in the period of 2pm to 4pm hours also buy beer. There are also rules of associations that occur cyclically in nature, as the summer and winter seasons in a tropical region. These rules having cyclical occurrence are denominated cyclical patterns.

Nowadays, several methods of mining temporal data have been proposed [1], [2], [3], [4], [5], [6], [7], [8], [11], [12], [13], [14]. However, these methods are not applicable to mine generalized cyclical rules from quantitative temporal data. There are cyclic pattern definitions in the literature, as related in [9], but the proposed methods identify only exact cycles and they do not detect events that happen approximately cyclically (quasi-cyclical), like: the dry season and the rainy season, planting and harvest cycles of agricultural crops, and others. In these cases, the application of the traditional pattern mining techniques is more difficult because the variations (start, end, duration, and intensity) of the quasi-cycles.

In this work, we present the formal definition of the quasi-cyclical pattern concept and we develop an approach to detect these patterns from quantitative temporal databases. This mining process of quasi-cyclical patterns in the form of rules can be summarized as following: we use a genetic algorithm (GA) to extract rules from quantitative temporal databases; The rules encoded by GA chromosome are checked in the database to build a binary sequence according to their occurrence, and non-occurrence in each period; We calculate the ability of a chromosome based on how well these binary sequences fit in a cycle.

This work is divided into the following sections: in the Section 2, we present the definitions of cyclical and quasi-cyclical patterns. In the Section 3, we describe the used method and the designed algorithm to detect the quasi-cyclical and cyclical patterns. Then we present results obtained from synthetic binary sequences and real data in the Section 4. Finally, in the Section 5, we discuss about the work, its contributions and future work.

2. Definitions

Let $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m\}$ the set of observed variables. Episode is the value of a given variable $\mathbf{v}_i$, observed at a given instant of time. Let $E = \{e_1, e_2, \ldots, e_m\}$ a set of episodes recorded every time instant t, where e_i is an episode associated with the variable $\mathbf{v}_i$. The set E, at a given instant of time t, is called *super-episodesets* because they are recorded values of all m variables, at a given instant t.

A given interval condition is used to analyze if a variable value occurs at a given interval. An episodic condition is a condition for an interval variable at a given time interval. An association rule is an implication of the form $\mathbf{X} \Rightarrow \mathbf{Y}$ (if $\mathbf{X}$ then $\mathbf{Y}$), where $\mathbf{X}$ and $\mathbf{Y}$ are conjunctions of episodic conditions associated to the observation variables.

A primary cycle or period l_{cb} is a contiguous sequence (adjacent) of observed *super-episodesets*. The base cycle length (or duration) l_{cb} is given by the number of *super-episodesets* of the base cycle, which is predetermined. Mathematically, each base cycle or period corresponds to the time interval specified by $[t_{i \cdot l_{cb}}, t_{(i+1) \cdot l_{cb}})$. The base cycles number n_{cb} is calculated as $\lfloor \frac{|\mathcal{D}|}{l_{cb}} \rfloor$, where $|\mathcal{D}|$ is the *super-episodesets* number in the database.

According Ozden et al. [9], cyclical patterns are rules that occur in regular intervals from a given offset, for example, weekly, monthly, yearly, and others. Let c an occurrence cycle of the rule R, then:

$$c = (l, o),$$

where l is the cycle length, obtained by the number of base cycles or the period between an occurrence and another; and o is the offset, which corresponds to the first instant that the rule R occurs, such that $0 \leq o \leq l$.

The pattern of an association rule occurrence in the base cycles or periods can be represented by a binary sequence, which each value caractherize a period. Thus a value 1 at a given period means that the rule occurred at this period and a value 0 represents no occurrence of the rule. For example, given the rule cited in the section 1 "meat $\rightarrow$ beer" and the binary sequence 010110011010 to represent this rule occurrence, then the rule has the cycle $c = (3, 1)$. It means that this rule occurrence starts in the second period (p_1) and it occurs every 3 times.

A quasi-cyclic pattern is the occurrence of certain rule R in periods not totally accurate. A quasi-cyclic has a frequency component or occurrence probability, which determines rate repetition of the the rule R in the periods. Given a quasi-cycle qc, then:

$$qc = \{(l, o), freq\}$$

where l is the cycle length (calculated by the number of base cycles or periods), which the rule occurrence is checked and the o is the offset, i.e., the first period that R occurs, given $o \geq 0$. We can have $o > l$, when the quasi-cycle does not occur in the first period. If $o < l$, then the rule occurs in the first period. The probability of occurrence of the rule R, $0 \leq freq \leq 1$, is given by:

$$freq = \frac{nPeriods}{n_p},$$

where $nPeriods$ is the periods number that R occurs from o, and n_p is the cycles number from o.

Using again the binary sequence 010110011010 as example, it has, for instance, the quasi-cycle $qc = \{(4, 0), 0.667\}$ with a frequency of about 66.7%. That is, there are 3 periods of length 4 and the quasi-cycle appears in 66.7% of these periods. Remember that each bit of the binary sequence corresponds to the occurrence or not of the rule in a base cycle.

3. Methodology

To detect quasi-cyclical patterns of rules occurrence, first, we developed a mining technique of quantitative temporal association rules based on the proposition of a specific genetic algorithm to this task. The technique is described in terms of the steps and operators of the genetic algorithm and it is given by Algorithm 1.

Algorithm 1: Genetic Algorithm (GA)

Input: Coding of chromosome, fitness function, constraints (attributes of antecedent of rules and attributes of consequent of rules), maximum length of interval ($piv.(\max \mathbf{v}_i - \min \mathbf{v}_i)$), value minimum of time window ($janTime$).

Output: Quantitative temporal association rules.

1 Generate, randomly, a population of chromosomes ($\mathcal{C}$);
2 Evaluate each chromosome $\mathcal{C}_i$ of the population based in the fitness function;
3 Apply the method of *niching*;
4 Select the chromosomes by the roulette method until complete the set of parents (*matting pool*);
5 Apply the uniform *crossover* taking pairs of individuals in the set of parents;
6 Apply the uniform mutation in the newly generated chromosomes;
7 Select the best chromosomes among parents and children to the next generation;
8 While the maximum number of generations is not reached, return to step 2;
9 Return the set of association rules encoded by the population of chromosomes;

w	AC	v_0	v_1	t_0	t_1

Fig. 1: Gene representation.

3.1 Coding of chromosome

In the chromosome coding, each database variable is associated with a gene. If we consider m variables of observation, then we will have m genes: G_1, G_2, G_3, ..., G_m. Each chromosome gene G_i represents a episode associated to a variable $\mathbf{v}_i, i = 1 \ldots m$, and it is encoded as shown in the Figure 1.

In the Figure 1, w is a weight that is compared to a threshold to indicate if the episodic condition represented by the gene will be part or not of the rule. *AC* is a *flag* to indicate whether the episodic condition represented by gene will be part of the antecedent (*AC*=0) or consequent (*AC*=1) of the rule. v_0 and v_1 are the lower and upper limits of the variable range, respectively. t_0 and t_1 are the lower and upper limits of the time interval, respectively.

3.2 Fitness measure

The chromosome fitness measure is obtained by the rule occurrence frequency of this chromosome. If the rules occurrence corresponds to an exact cycle, then the frequency is 1 and consequently the chromosome fitness is also 1. But, if

the rule occurrence pattern is only quasi-cyclic, so the fitness is the quasi-cycle frequency, calculated as:

$$freq = \frac{nPeriods}{n_p},$$

where $nPeriods$ is the number of periods that R occurs from o, and n_p is the cycles number from o, given that $freq \in \mathbb{R}$ and $0 \leq freq \leq 1$. If we identified more than one quasi-cycle, the fitness value is given as the frequency of the most frequent quasi-cycle.

3.3 Genetic operators

We did the selection for reproduction using the roulette method. The pairs of individuals selected for reproduction are crossed by uniform crossover: select by random, a mask with the chromosome length, which indicates what parent chromosome will provide each gene to the first child; the second child is generated by the mask complement.

Each chromosome selected for mutation will have one mutated gene. The mutation can occurs in the weight w, in *AC* (when restrictions were not applied on the variables that composes the antecedent and consequent), or in the lower limits (v_0 and t_0) and higher (v_1 and T_1) of the variables intervals and of time.

3.4 Niching

We use a niching method , called clearing, described in Petrowski [10]. We propose a distance measure for quantitative temporal association rules in this method. The following, we describe the clearing niching method and the proposal distance measure.

The clearing method corresponds to the niching concept enunciated by JH Holland in 1975: the resources sharing by individuals population characterized by some similarity. However, instead of sharing the available resources, the clearing method provides a niche features only for the best individual of each subpopulation. This allows that the genetic algorithm performs a multimodal optimization (find the optimal and optimum at the same time). Furthermore, the clearing method of the enables that GA reduces the genetic drift problem, when it is used together with an appropriate selection operator.

The clearing is applied between the chromosomes fitness evaluation and the selection operator application for the crossing. The method uses a distance measure (dissimilarity) among the chromosomes (the phenotype, in our case, means association rules) to determine if they belong to the same subpopulation or not. Each subpopulation will have a dominant chromosome: witch has the highest fitness value in the subpopulation. If a chromosome belongs to a subset, then their dissimilarity related to the dominant chromosome is smaller than a given threshold σ, called clearing radius. The clearing method preserves the the dominant chromosome ability while decreases to zero the other chromosomes ability. Thus, the clearing assigns all the resources of a niche for a single chromosome: the winner. This method corresponds to remove of the population, in an imaginary way, all dominated individuals.

The clearing method also is generalizable to accept multiple winners chosen from the best individuals of the niche [10]. The niche ability is defined as the maximum number of chromosomes that a niche can have. If the capacity is greater than 1, then the population has more than one winner. If the niche capacity is equal to the population size, the clearing effect disappears and the search method becomes a standard GA. Thus, the choice of the niche capacity between 1 and the population size provides intermediate situations between the maximum clearing effect and a standard GA search.

The algorithm 2 presents a clearing method as in Petrowski [10]. Consider $\mathcal{C}$ (the chromosomes population) and $n_{\mathcal{C}}$ (chromosomes number in the population) as the global variables. Adding, σ (the clearing radius) and κ (capacity of each niche) are the input parameters. The variable *nbWinner* counts the number of population winners associated to the current niche. The chromosomes population $\mathcal{C}$ is represented by a vector of $n_{\mathcal{C}}$ chromosomes.

Algorithm 2: Clearing niching

Input: σ (clearing radius), κ (capacity of each niche).
Output: Clearing – allocation of the niche resources for the the fittest individual.

```
1  SortFitness(C);
2  for i ← 0 to n_C − 1 do
3      if Fitness(C[i]) > 0 then
4          nbWinners ← 1;
5          for j ← i + 1 to n_C − 1 do
6              if Fitness(C[j]) > 0 and
                Distance(f(C[i]), f(C[j])) < σ then
7                  if nbWinners < κ then
8                      nbWinners ← nbWinners + 1;
9                  else
10                     Fitness(C[j]) ← 0;
11                 end
12             end
13         end
14     end
15 end
```

The clearing algorithm (Algorithm 2) uses three functions:

- *SortFitness*($\mathcal{C}$): ordering the chromosomes population in the fitness order.
- *Fitness*($\mathcal{C}$[i]): returns the i-th chromosome ability of the population $\mathcal{C}$.
- *Distance*($f(\mathcal{C}$[i]), $f(\mathcal{C}$[j])): returns the distance between the phenotypes of population chromosomes;

- $f(\mathcal{C}[i])$: returns the chromosome $\mathcal{C}[i]$ phenotype. In our case the phenotype is a quantitative temporal association rule.

3.5 Calculating the distance between association rules

To take two association rules:

$$\begin{aligned} R &= (CA_R(v_{j_1}) \text{ AND } \ldots \text{ AND } CA_R(v_{j_n})) \\ &\Rightarrow (CC_R(v_{j_1}) \text{ AND } \ldots \text{ AND } CC_R(v_{j_m})) \end{aligned}$$

and

$$\begin{aligned} S &= (CA_S(v_{j_1}) \text{ AND } \ldots \text{ AND } CA_S(v_{j_o})) \\ &\Rightarrow (CC_S(v_{j_1}) \text{ AND } \ldots \text{ AND } CS_S(v_{j_p})) \end{aligned}$$

where $CA_R(v_{j_i})$, $CC_R(v_{j_i})$, $CA_S(v_{j_i})$, $CC_S(v_{j_i})$ are episodic conditions related to some variable v_{j_i} of the database. Each episodic condition has the form:

$$v_i \in [v0_i, v1_i] \text{ in the time interval } [t0_i, t1_i]$$

where v_i is any database variable. For a given rule R, the intersection of variables associated to the precedent (VA_R) episodic conditions with the variables associated to consequent (VC_R) episodic conditions must be empty, ie, $\text{VA}_R \cap \text{VC}_R = \emptyset$.

The distance between R and S, denoted by *Distance*(R, S) is given by the algorithm 3. For each episodic condition in the antecedent rule R, CA_R, we check if there is any episodic condition in the antecedent rule S, CA_S, which is comparable to CA_R. Two episodic conditions are comparable if they refer to the same variable. If we have two episodic conditions that are comparable, we calculate the distance between them. Otherwise, we increment the distance counter by one. Then the distance counter is divided by the number of episodic conditions n_{CA_R}. The same calculation is done to the subsequent episodic conditions. Finally, the distance among the rules is given by $(dist_A + dist_C)/2$, and the distance among episodic conditions, $DistanceEp(C_1, C_2)$, is given by algorithm 4.

The cyclical patterns mining and quasi-cyclical patterns mining is given by the Algorithm 5. The quasi-cycles are identified by the occurrence mapping or not of each detected rule for each cycle and period. Since each chromosome encode a rule, if the rule occurrence has a exactly cyclic pattern, then the chromosome fitness is one, corresponding to a rule frequency of 100% in the period. If the rule occurrence corresponds to a quasi-cyclic pattern, the chromosome fitness is the occurrence frequency of the rule.

4. Results

In order to evaluate the developed algorithm to detect cycles and quasi-cycles, we did an experiment using a synthetic binary sequence. We used in this experiment, the

Algorithm 3: *Distance*(R, S) – Distance between two quantitative temporal association rules.

Input: R and S, two quantitative temporal association rules.
Output: $dist$, distance between R and S.

$n_{CA_R} \leftarrow$ *numCond*(CA_R);
$dist_A \leftarrow 0$;
for *i*$\leftarrow$ *0* **to** $n_{CA_R} - 1$ **do**
  if $\exists CA_S(v_{j_k}) | v_{j_k} = v_{j_i}, v_{j_i} \in VA_R$ **then**
    $dist_A \leftarrow dist_A + DistanceEp(CA_R(v_{j_i}), CA_S(v_{j_k}))$;
  else
    $dist_A \leftarrow dist_A + 1$;
  end
end
$dist_A \leftarrow dist_A / n_{CA_R}$;
$n_{CC_R} \leftarrow$ *numCond*(CC_R);
$dist_C \leftarrow 0$;
for *i* $\leftarrow$ *0* **to** $n_{CC_R} - 1$ **do**
  if $\exists CC_S(v_{j_k}) | v_{j_k} = v_{j_i}, v_{j_i} \in VC_R$ **then**
    $dist_C \leftarrow dist_C + DistanceEp(CC_R(v_{j_i}), CC_S(v_{j_k}))$;
  else
    $dist_C \leftarrow dist_C + 1$;
  end
end
$dist_C \leftarrow dist_C / n_{CC_R}$;
$dist \leftarrow (dist_A + dist_C)/2$;

Algorithm 4: *DistanceEp*(C_1, C_2) – Distance between two temporal episodes associated to the same variable.

Input: C_1, C_2, v (two episodic conditions associated with the same variable v).
Output: $distEp$, distance between C_1 and C_2.

calculate the minimum value that v assume in the database and store into variable $\min v$;
calculate the maximun value that v assume in the database and store into variable $\max v$;
calculate the minimum value that t assume in the database and store into variable $\min t$;
calculate the maximun value that t assume in the database and store into variable $\max t$;
$dv \leftarrow \min\{(v_1^{(C_1)} - v_0^{(C_1)}), v_1^{(C_2)} - v_0^{(C_2)})\} - (\min\{v_1^{(C_1)}, v_1^{(C_2)}\} - \max\{v_0^{(C_1)}, v_0^{(C_2)}\})$;
$distV \leftarrow dv/(\max v - \min v)$;
$dt \leftarrow \min\{(t_1^{(C_1)} - t_0^{(C_1)}), t_1^{(C_2)} - t_0^{(C_2)})\} - (\min\{t_1^{(C_1)}, t_1^{(C_2)}\} - \max\{t_0^{(C_1)}, t_0^{(C_1)}\})$;
$distT \leftarrow dt/(\max t - \min t)$;
$distEp \leftarrow (distV + distT)/2$;

Algorithm 5: *Cycle_QuasiCycle_Detection*($BoolVet$, $nVars$) – Detecting cyclical patterns and quasi-cyclical patterns that express the rules detected occurrence.

Input: $boolVector$ (vector of boolean values that express the occurrence of each rule), $nVars$ (number of observed variables).
Output: list of detected cycles ($cyclesList$) and list of quasi-cycles detected ($quasiCyclesList$).

```
1  for j ← 0 to length(boolVector) do
2      if boolVector[j] = 1 then
3          positions.add(j) ;
4      end
5  end
6  for i ← 1 to (length(boolVector)/2) do
7      for k ← 0 to (length(positions) − 1) do
8          quasiCycle[0] ← cycle[0] ← currentQuasi ←
           currentCycle ← positions[k];
9          for j ← (k + 1) to (length(positions) − 1) do
10             if positions[j] = (currentCycle + i) then
11                 currentCycle ← cycle[length(cycle) −
                   1] ← positions[k];
12             end
13             if (positions[j] − currentQuasi)%i = 0
               then
14                 quasiCycle.add(positions[k]);
15             end
16         end
17         nPeriods ←
           length(boolVector) − positions[k];
18         if length(cycle) = (length(boolVector)/i)
           then
19             cyclesList.add((i, positions[k]));
20         end
21         if length(quasiCycle) > 1 then
22             if length(quasiCycle) = nPeriods then
23                 cyclesList.add((i, positions[k]));
24             else
25                 if length(quasiCiclo) <
                   lenght(boolVet)/i and quasiCiclo is
                   not subset of any cycle ∈ listaCiclos
                   then
26                     frequency ←
                       length(quasiCycle)/nPeriods;
27                     quasiCyclesList.add(
28                         ((i, positions[k]), frequency));
29                 end
30             end
31         end
32     end
33 end
```

```
Cycles:
        (1, 9)
        (1, 10)
        (3, 0)
        (4, 2)
        (4, 3)
        (5, 2)
        (5, 6)

Quasi-cycles:
        ((1, 0), 0.67)
        ((1, 2), 0.7)
        ((1, 3), 0.67)
        ((1, 6), 0.83)
        ((1, 7), 0.8)
        ((2, 0), 0.33)
        ((2, 2), 0.3)
        ((2, 3), 0.44)
        ((2, 6), 0.33)
        ((2, 7), 0.6)
        ((3, 2), 0.2)
        ((3, 7), 0.4)
        ((5, 0), 0.17)
```

Fig. 2: Cycles and quasi-cycles mined.

binary sequence “101100110111". The quasi-cycles and the cycles extracted from the sequence are shown in Figure 2.

We also performed two case studies in real quantitative temporal databases related to agrometeorological data, which are reported in the following. We used fixed values to the chromosomes population size (n_C=50), the generations number (*nGen*=250), the clearing radius (σ=0.5) and the maximum allowable variable range. The maximum allowable variable range is defined by $piv \cdot (\max v - \min v)$, where piv in these experiments has a value of 10%. We also kept the cross rate of 80% and the mutation rate of 3% for chromosome. In order to present the detected rules, we used a threshold of 0.4 for the fitness measure. Rules with lower fitness measure were not considered important because they have low occurrence frequency. The presentation of quasi-cycles were done in the same way.

4.1 Case study 1: Araraquara

In this case study, we used the database named Araraquara, collected by the Brazilian Agro-meteorological Monitoring System – Agritempo (http://www.agritempo.gov.br/). This database contains monthly agro-meteorological data of Araraquara, corresponding to the average minimum temperature values (*Tmin*), average maximun temperature values (*Tmax*), accumulated rainfall (*Prec*), average of Normalized Difference Vegetation Index (*NDVI*) and average of Water Requirement Satisfaction Index (*WRSI*). The data

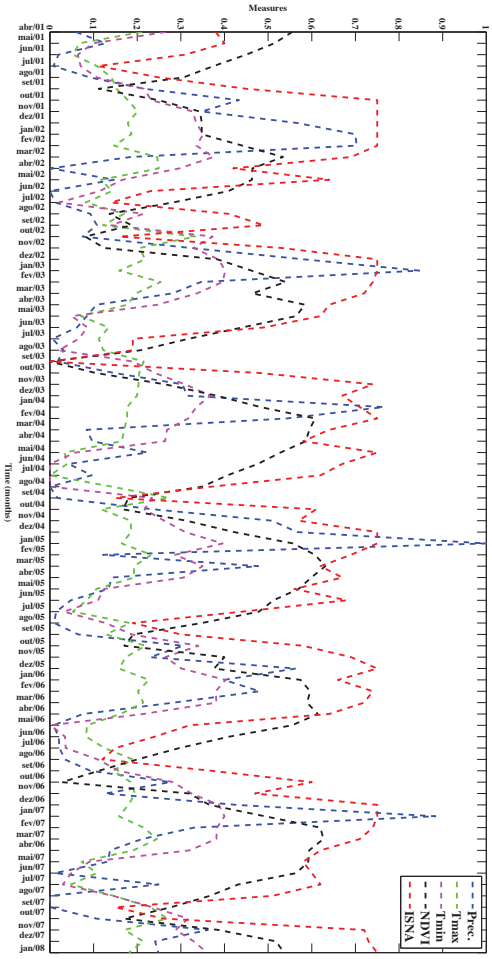

Fig. 3: Agrometeorological monthly measures of Araraquara's brazilian city.

corresponds to the period from April 2001 to January 2008. We plotted the variable values collected during these months in the Figure 3. Due to the large scale difference among the variables values, they were scaled in the range $[0, 1]$, so that we observe better the trends.

In this study, we had imposed the restriction that the it NDVI will be part of the rule consequent. The Figure 4 presents the mined rules together with the quasi-cycles and the cycles detected in the technical implementation using the parameters described in the section 4.

4.2 Case study 2: Piracicaba

In this case study, we use the cane sugar productivity database in Piracicaba, provided by Cepagri (Centre for Research in Agrometeorology and Climate Applied to Agriculture) maintained by State University of Campinas (UNICAMP). The database consists of four variables taken monthly in Piracicaba: the average minimum temperature (*Tmin*), the average maximum temperature (*Tmax*), average rainfall (*Prec*) and productivity of cane sugar (*Prod*) in

```
Rule:
     Tmax (celsius) in [27.23, 28.33] on dez-jan
     ==>
     NDVI (taxa) in [0.58, 0.62] on dez-jan

     Fitness = 1.000

     Cycles:
     (1, 0)
```

```
Rule:
     WRSI
 (taxa) in [0.98, 1.00] on jan-fev
     ==>
     NDVI (taxa) in [0.58, 0.62] on jan-fev

     Fitness = 1.000

     Cycles:
     (1, 0)
```

```
Rule:
     Prec (mm) in [10.16, 61.17] on abr-mai
     ==>
     NDVI (taxa) in [0.26, 0.30] on abr-mai

     Fitness = 0.667

     Quasi-cycles:
     ((1, 0), 0.666667)
     ((1, 2), 0.666667)
     ((1, 3), 0.500000)
```

Fig. 4: The rules mined in the Araraquara's database.

tonnes per hectare (ton/hec). The database corresponds to the period from January 2003 to December 2009. We plotted the variable values collected during these months in the Figure 5.

In this study, we had imposed the restriction that productivity ($Prod$) will be part of the consequent of the rule. We fixed the used parameters, as described before in the Section 4. The mined rules, cycles and quasi-cycles detected for an algorithm execution are shown in Figure 6.

4.3 Considerations

Overall, we can see that the technique can mine several association rules and can detect cyclical patterns and *quasi-cyclical* occurrences of these rules. The chromosome fitness value is calculated using the highest detected pattern frequency, which have a maximum value of one, when a cycle is found. We fixed the parameters for the two case studies, but they can be different and change some results. For example, using a larger population size (n_C) and a larger generations number (*nGen*), we expect to mine a larger rules number. Likewise, how smaller the clearing radius, theoretically, the environment capacity to accommodate niches and, therefore, the amount of mined rules increase.

5. Conclusion

Given the quasi-cyclical patterns definition and the rules mining with quasi-cyclical temporal occurrence, we showed that is possible to detect relevant rules for the analyzed databases. These rules, in this format used, not can mined by any literature algorithm for cycles mining. Additionally, the quasi-cycle definition allows the relevant patterns identification, which could not be found using the exact cycle detection.

Therefore, in this work, we presented a methodology for quasi-cyclic patterns detection, which can be used to detect patterns that occur approximately cyclically from quantitative temporal data, as in the case studies 4.1 and 4.2.

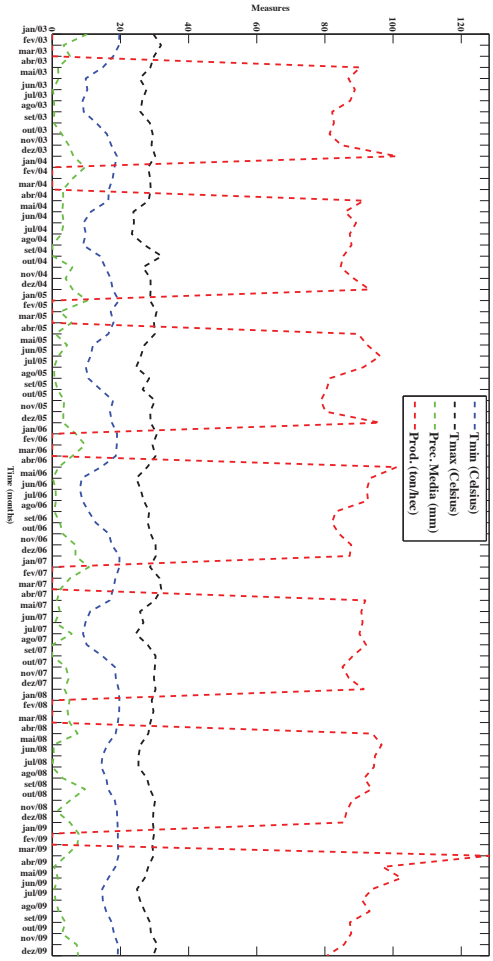

Fig. 5: Agrometeorological measures related to productivity of cane sugar in Piracicaba's brazilian city.

```
Rule:
     AvgPrec (mm/dia) in [2.50, 3.58] on out-nov
     ==>
     Prod
 (ton/hec) in [119.97, 128.20] on out-nov

     Fitness = 0.833

     Quasi-cycles:
     ((1, 0), 0.833333)
     ((1, 2), 0.666667)
     ((1, 3), 0.500000)
```

```
Rule:
     Tmin (celsius) in [17.93, 19.10] on abr-mai
     ==>
     Prod
 (ton/hec) in [119.97, 128.20] on abr-mai

     Fitness = 0.429

     Quasi-cycles:
     ((1, 5), 0.428571)
```

```
Rule:
     Tmax (celsius) in [28.35, 29.23] on nov-dez
     ==>
     Prod
 (ton/hec) in [119.97, 128.20] on nov-dez

     Fitness = 0.429

     Quasi-cycles:
     ((1, 0), 0.428571)
     ((1, 1), 0.428571)
```

Fig. 6: The rules mined in the Piracicaba's database.

This pattern type is extremely common in nature, according to we discussed in the introduction of this article.

As future work, we intend to set the genetic algorithm parameters, which were kept fixed in all experiments described in this work. We expect that this adjustment will allow to find a larger number of relevant rules. Other future work is to develop and to test other fitness measures, or improve the adopted fitness measure.

Acknowledgments

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq) and Research Support Foundation of Goiás State (FAPEG).

References

[1] S. Amo, N. A. Silva, R. P. Silva, and F. S. Pereira. Tree pattern mining with tree automata constraints. *Information Systems*, 35:570–591, 2010.

[2] B. Catania and A. Maddalena. A unified framework for heterogeneous patterns. *Information Systems*, 37:460–483, 2012.

[3] D.-A. Chiang, C.-T. Wang, S.-P. Chen, and C.-C. Chen. The Cyclic Model Analysis on Sequential Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1617–1628, 2009.

[4] J. K. Febrer-Hernández and J. Hernández-Palancar. Sequential pattern mining algorithms review. *Intelligent Data Analysis*, 16(3):451–466, 2012.

[5] T. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[6] Y. Hai and X. Li. A general temporal association rule frequent itemsets mining algorithm. *International Journal of Advancements in Computing Technology*, 3(11):63–71, 2011.

[7] Yong Joon Lee, Jun Wook Lee, Duck Jin Chai, Bu Hyun Hwang, and Keun Ho Ryu. Mining temporal interval relational rules from temporal data. *Journal of Systems and Software*, 82(1):155–167, 2009.

[8] N. R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1), 2010.

[9] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *Proceedings of the Fourteenth International Conference on Data Engineering*, pages 412–421, Washington, DC, USA, 1998.

[10] A. Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of 3rd IEEE International Conference on Evolutionary Computation*, pages 798–803, New York, USA, 1996.

[11] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, and Y. W. Choong. Mining multidimensional and multilevel sequential patterns. *ACM Transactions on Knowledge Discovery from Data*, 4(1):1–37, 2010.

[12] N. Tatti and B. Cule. Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1):34–66, 2011.

[13] X. Yan, C. Zhang, and S Zhang. Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems with Applications*, 36:3066–3076, 2009.

[14] M. Zhang and C. He. Survey on association rules mining algorithms. *Advancing Computing, Communication, Control and Management*, 56:111–118, 2010.