Mining Fuzzy Temporal Gradual Emerging Patterns

Dickson Odhiambo Owuor, Anne Laurent, Joseph Onderi Orero

Mining Fuzzy Temporal Gradual Emerging Patterns

Dickson Odhiambo Owuor
School of Computing & Engineering Sciences
Strathmore University, Nairobi, Kenya
dowuor@strathmore.edu

Anne Laurent
LIRMM
Univ Montpellier, CNRS
Montpellier, France
anne.laurent@umontpellier.fr

Joseph Onderi Orero
School of Computing & Engineering Sciences
Strathmore University, Nairobi, Kenya
jorero@strathmore.edu

Received (received date)
Revised (revised date)

Gradual emerging patterns (GEPs) are gradual item sets that occur less frequently in one data set and more frequently in another. For instance, let *'fan speed'* and *'temperature'* be attributes of two numerical data sets. A gradual item set *"the higher the speed, the lower the temperature"* (which correlates a data set's attributes) becomes a GEP if it is less frequent (in terms of support as in frequent pattern mining) in one data set and more frequent in another. However, such patterns do not indicate how time gap impacts the emergence. Many correlations appear over time, for instance when phenomena appear after some meteorological situation due to latency. Previous works have not taken this temporal aspect into account. In this paper, we introduce temporal gradual emerging patterns (TGEPs) which are temporal gradual patterns (TGPs) whose frequency supports increase significantly between transformed data sets. For instance, a TGP *"the higher the speed, the lower the temperature, almost 3 minutes later"* becomes a TGEP if it occurs more frequently in one transformed data set than in another. Furthermore, we extend border manipulation to the case of mining TGEPs. In addition, we propose a more efficient ant colony optimization technique that exploits a heuristic approach to construct TGEPs.

Keywords: frequent pattern mining; gradual patterns; emergent patterns; temporal data; ant colony optimization

1. Introduction

Emerging patterns (EPs) are item sets whose frequency increases significantly from one data set to another. EPs are described using *growth rate*, which is the ratio of an EP's frequency support in one data set to its frequency support in another data set. For instance, suppose a car shop in 2017 had 200 purchases of $\{FOG\_LAMPS, BATTERY, TYRES\}$ out of 1000 transactions, and in 2018 it had 500 such purchases out of 1000 transactions. This purchase is an EP with a *growth rate* of 2.5 from the year 2017 to 2018.

More specifically, an EP is present if its *growth rate* across data sets is larger than a given specified minimal numerical threshold. EPs can be applied to discover distinctions that exist amongst a collection of data sets with classes such as *"hot vs cold"*, *"poisonous vs edible"*. In other words, EPs are a powerful tool for capturing discriminating characteristics between the classes of different data sets.[1–3]

Gradual patterns identify relationships among data sets' attributes in order to discover correlation knowledge that take linguistic representations such as: *"the more A, the less B"*.[4–6] Further, as an extension to gradual patterns, temporal gradual patterns (TGPs) additionally identify the temporal tendencies of correlations among these attributes.[7] For instance a TGP may take the form: *"the more A, the less B,* ***2 days later.****"* This is achieved by a technique that allows for transformation of a single timestamped data set into numerous data sets based on *date-time* attribute then, mining them for TGPs.

In this paper, we extend TGPs described in order to introduce temporal gradual emerging patterns (TGEPs). TGEPs may be defined as temporal gradual item sets whose frequency supports increase significantly between transformed data sets. TGEP mining unearths a new possibility for discovering gradual trends with respect to time in timestamped numerical data sets. For example, given a timestamped weather data set with attributes *'rain'* and *'wind'*, a TGEP may take the form: *"the more rain, the more wind, almost 2 hours later"* with a **frequency support of 0.06** in one transformed data set and, *"the more rain, the more wind, almost 2 hours later"* with a **frequency support of 0.84** in another transformed data set.

Most often, apart from the significant increase in frequency support, it may be possible for a drift of time gap to additionally occur. For instance the second transformed data set may produce the pattern: *"the more rain, the more wind, almost 5 hours later"* with a **frequency support of 0.84**. The latter extracts more meaningful knowledge from the data set, since it also shows the amount of time that elapses before/after a temporal gradual item emerges. This pattern has a *growth rate* of 14 after approximately *3 hours.* This example can be extended to real-life timestamped data sets to extract more interesting TGEPs.

In spite of this, it should be remembered that TGEPs are extracted from transformed data sets. For this reason, the complexity of dealing with more than 2 transformed data sets when extracting TGEPs arises. The process of mining all TGPs from each of these data sets and comparing the patterns against each other to identify emerging ones proves to be computationally time-consuming. Consequently, we introduce 3 contributions in the section that follows.

Main Contributions

- We introduce and describe temporal gradual emerging patterns (TGEPs).
- We propose a novel approach based on an ant colony optimization (ACO) technique for mining TGEPs. ACO, originally described by,[8] is a heuristic approach for optimizing combinatory problems. In this study, we extend ant colony optimization to the problem of mining TGEPs.
- We extend border manipulation technique to the case of mining TGEPs and, compare the computational efficiency of this border-based technique to that of ant-based technique. A border is a concept that is used to represent interval closed item sets[a] as a pair.[9,10]

The remainder of this paper is organized as follows: we provide preliminary definitions and notations of gradual patterns and emerging patterns in Section 2. We describe temporal gradual patterns (TGPs) and introduce TGEPs in Section 3. We propose a border-based approach and an ant-based approach for extracting TGEPs in Section 4 and Section 5 respectively. We compare the performance of these two proposed approaches in Section 6. Finally, we conclude in Section 7.

2. Preliminary Concepts and Notations

In order to describe TGEPs, we provide some preliminary definitions of gradual patterns (as given by[5,7,11,12]) and emerging patterns (as given by[3,9,13]).

2.1. *Gradual Patterns*

In the case of gradual patterns, assume a data set $\mathcal{D}_g$ is defined by attributes $\{A_1, A_2, ..., A_n\}$ and it consists of tuples $\{r_1, r_2, ..., r_k\}$ (as shown in Table 1).

Table 1: Sample data set containing rainfall amount and wind speed recordings.

id	**rain** (mm)	**wind** (km/h)
r1	10	42
r2	13	21
r3	18	35
r4	10	21
r5	18	35

[a]collection $\mathcal{S}$ of sets are said to be interval closed if X and Z are in $\mathcal{S}$ and Y is a set such that $X \subseteq Y \subseteq Z$.

Example 2.1. Let us consider a data set ($\mathcal{D}_g$) in Table 1

Definition 2.1. `Gradual Item.` *A gradual item is a pair* (i, v) *where* i *is an attribute and* v *is a variation* $v \in \{\uparrow, \downarrow\}$. $\uparrow$ *stands for an increasing variation while* $\downarrow$ *stands for a decreasing variation.*

For example, $(rain, \uparrow)$ may be interpreted as *"the more rain"*.

Definition 2.2. `Gradual Pattern.` *A gradual pattern (GP) is a set of gradual items, denoted by* $GP = \{(i_1, v_1), ..., (i_n, v_n)\}$.

For example, $\{(rain, \uparrow), (wind, \downarrow)\}$ is a gradual item set that may be interpreted as *"the more rain, the less wind"*.

The quality of a gradual pattern is measured by support. Support (sup) *of a gradual pattern is the ratio of the proportion of tuples that respect the pattern to the total number of tuples.* Therefore, given a minimum numerical threshold σ, a gradual pattern GP is said to be frequent only if:

$$sup(GP) \geq \sigma \tag{1}$$

It should be underlined that the process of summing up tuples that respect a particular gradual pattern is achieved by pairing the affected tuples in the order of their values. Therefore, it is impossible to determine if a gradual pattern is valid by counting each tuple individually. For this reason, one efficient technique for calculating the support of a gradual pattern involves exploiting a representation of these tuple pairings as concordant or discordant pairs.[6,7]

For example, given the data set in Table 1, the total number of pairs is given by the formula: $k(k-1)/2$ where k is the number of tuples in the data set. In this example the total number of pairs is 10 and the pairs of tuples that respect the gradual pattern (concordant pairs) $\{(rain, \uparrow), (wind, \downarrow)\}$ are: $\{[r1, r2], [r1, r3]\}$. Therefore, the support for the pattern is 2/10.

It should be emphasized that it is trivial to extract 1-item set gradual patterns (i.e. $\{(rain, \uparrow)\}$) since they do not uncover any meaningful correlation knowledge. Hence, the least length for any elementary gradual pattern is 2.[12]

2.2. *Emerging Patterns*

In the case of emerging patterns, assume a data set $\mathcal{D}$ is defined by item set $I = \{i_1, i_2, ..., i_n\}$ and it consists of transactions $\{t_1, t_2, ..., t_n\}$. Every transaction is a subset of item set I (as shown in Table 2 (a)).

Definition 2.3. `Emerging Pattern.` *An emerging pattern* (EP) *is a set of items that appear less frequently in transactions of one data set* $\mathcal{D}_1$ *and more frequently in transactions of another data set* $\mathcal{D}_2$.

Example 2.2. Let us consider two data sets: ($\mathcal{D}_1$) in Table 2 (a), and ($\mathcal{D}_2$) in Table 2 (b).

Table 2: Two sample data sets containing transactions

transaction	items
t1	bread, milk, sugar
t2	eggs, milk
t3	cheese, bread
t4	butter, sugar

(a)

transaction	items
t1	bread, milk, sugar
t2	eggs, bread, milk
t3	bread, milk, sugar, cheese
t4	bread, milk, sugar, eggs

(b)

For example, (as illustrated in Table 2) $\{bread, milk, sugar\}$ is an emerging item set since its frequency occurrence count in transactions (or frequency support) is significantly greater in data set $\mathcal{D}_2$ than in data set $\mathcal{D}_1$.

The quality of an emerging pattern is measured by growth rate. Growth rate (gr) *of an emerging pattern* (EP) *is the ratio of frequency support of the pattern in data set* $\mathcal{D}_2$ *to another data set* $\mathcal{D}_1$. For example, given $\mathcal{D}_1$ and $\mathcal{D}_2$, the *growth rate* of item set EP in favour of data set $\mathcal{D}_2$ is given as follows:

$$gr(EP) = \begin{cases} 0 & if\ sup(EP)_{\mathcal{D}_2} = 0 \\ \infty & if\ sup(EP)_{\mathcal{D}_1} \neq 0\ ,\ sup(EP)_{\mathcal{D}_2} = 0 \\ \dfrac{sup(EP)_{\mathcal{D}_2}}{sup(EP)_{\mathcal{D}_1}} & otherwise \end{cases} \tag{2}$$

Therefore, given a numerical threshold ρ, item set EP is emerging only if:

$$gr(EP) \geq \rho \tag{3}$$

It is important to emphasize that growth rate is derived from frequency support of the involved patterns.[9,10] Similar to transactional frequent pattern mining, frequency support as a quality measure for extracted patterns also applies to gradual pattern mining. It is for this reason that the concept of emerging patterns can be extended to the case of gradual pattern mining.[12] Therefore, a gradual emerging pattern may be defined as follows:

Definition 2.4. `Gradual Emerging Pattern.` *A gradual emerging pattern* (GEP) *is a set of gradual item sets whose support* $sup(GEP)_{\mathcal{D}_{g2}} > sup(GEP)_{\mathcal{D}_{g1}}$.

For example, $GP = \{(rain, \uparrow), (wind, \uparrow)\}$ is a gradual emerging pattern if support $sup(GP)$ is significantly greater in data set $\mathcal{D}_{g2}$ than in data set $\mathcal{D}_{g1}$.

3. Temporal Gradual Emerging Patterns

In this section, we seek to describe temporal gradual emerging patterns (TGEPs). We begin by describing temporal gradual patterns (TGPs) because an emerging TGP makes up a TGEP. TGP mining extends gradual pattern mining in order to additionally estimate the time lag that may exist between gradual item sets.[7]

For instance, a TGP may take the form *"the higher X, the higher Y* ***almost 3 months later****"*. For the purpose of clarity, we provide definitions of TGPs (as given by[7]) as follows.

Definition 3.1. `Time Lag.` *"a time lag (tl) is the amount of time that passes before or after changes in one gradual item affects the changes in another gradual item denoted by $tl = \alpha\beta t$"*. Where α is an operator $\alpha \in \{+, -\}$ and '+' implies after and '$-$' implies before; β is an operator $\beta \in \{=, \simeq\}$ and '=' implies equal to and '$\simeq$' implies almost; t is the value of time lag.

For example, '$- \simeq 1week$' is a time lag that may be interpreted as *'almost 1 week earlier'*.

Definition 3.2. `Fuzzy-Temporal Gradual Item.` *A fuzzy-temporal gradual item is a temporal gradual item with a fuzzy time lag $\alpha\beta t$* where $\alpha \in \{+, -\}$ and $\beta \in \{\simeq\}$ so that '$+ \simeq t$' implies a time lag of almost t later and, '$- \simeq t$' implies a time lag of almost t earlier.

For example, $(wind, \uparrow)_{-\simeq 1week}$ is a fuzzy-temporal gradual item that may be interpreted as the *"the more wind, almost 1 week earlier"*.

Definition 3.3. `Fuzzy-Temporal Gradual Pattern.` *A fuzzy-temporal gradual pattern (TGP_f) consists of one reference gradual item and a set of fuzzy-temporal gradual items, denoted by $TGP_f = \{(i_1, v_1), (i_2, v_2)_{\alpha\beta t_2}, ..., (i_n, v_n)_{\alpha\beta t_n}\}$.*

It should be realized that for Definition 3.3 to be relevant, there must be one *reference gradual item.* A *reference gradual item* is determined by a user-specified reference attribute. From the reference gradual item other temporal gradual items are added with an associated time lag.

For example, $\{(rain, \uparrow), (temperature, \downarrow)_{+\simeq 2weeks}\}$ is a fuzzy-temporal gradual pattern: where $(rain, \uparrow)$ is a reference gradual item and the pattern may be interpreted as *"the more rain, the lower the temperature, almost 2 weeks later"*.

Example 3.1. Let us consider a data set ($\mathcal{D}_g$) shown in Table 3.

Table 3: Sample timestamped data set containing rainfall amount and wind speed

id	**date** (day/month)	**rain** (mm)	**wind** (km/h)
r1	01/06	10	42
r2	04/06	13	21
r3	05/06	18	35
r4	10/06	10	21
r5	12/06	18	35

The first process in the extraction of TGPs is to *transform* a timestamped numerical data set step-wisely into a temporal format. This process requires that one attribute be selected as a *reference attribute* and *transformation step* be set. For example, (using data set in Table 3) if *'rain'* is selected as the reference attribute and a transformation step be set at 1; then the data set ($\mathcal{D}_g$) in this table will be transformed into data set ($\mathcal{D}'_g$) in Table 4 such that:

- every tuple (r_n) of *'rain'* attribute is mapped to tuple (r_{n+1}) of *'wind'* attribute, and
- date difference is calculated as ($r_n - r_{n+1}$).

Table 4: Data set ($\mathcal{D}'_g$) transformed from data set $\mathcal{D}_g$ by *step:* $s = 1$.

id	**date *diff*** ($r_n - r_{n+1}$)	**rain** (r_n)	**wind** (r_{n+1})
t1	3	10	21
t2	1	13	35
t3	5	18	21
t4	2	10	35
t5	-	-	-

It is observable that a transformation step of 1, leads to generation of a transformed data set ($\mathcal{D}'_g$) that represents 4 out of the 5 tuples of the original data set ($\mathcal{D}_g$). Therefore, $\mathcal{D}'_g$ has a *representativity* of 0.8. Representativity (*rep*) *of a TGP is the ratio of tuple size in a transformed data set ($\mathcal{D}'_g$) to tuple size in the original data set ($\mathcal{D}_g$).* Therefore, given a minimum numerical threshold δ, a *TGP* is relevant only if:

$$rep(TGP) \geq \delta \tag{4}$$

The second process in the mining of TGPs is to extract gradual patterns from the *transformed data set* and approximate a *time lag* associated with the extracted gradual patterns.

As shown above, TGPs are extracted from *transformed data sets*. The value of the specified minimum representativity (δ) determines the number of transformation steps; consequently, the number of transformed data sets. It may be the case that a particular TGP occurs frequently in more than one transformed data set; as a result, it becomes an *emerging TGP*. Under those circumstance, we may define a TGEP as follows.

Definition 3.4. `Temporal Gradual Emerging Pattern.` *A temporal gradual emerging pattern ($TGEP$) is a set of temporal gradual patterns that appear more frequently in one transformed data set $\mathcal{D}'_g$ and less frequently in another transformed data set $\mathcal{D}''_g$.*

4. Border-based Discovery of TGEPs

In this section, we propose an approach that exploits border manipulation for extraction of TGEPs. First, we describe how border manipulation technique may be applied to the case of frequent item sets and gradual item sets. Finally, we propose to extend it to the case of temporal gradual item sets.

4.1. *Border Representation of Frequent Item Sets*

The border-based approach was introduced by[9] and it offers condensed representation and efficient manipulation of large interval closed item sets. In considerations of clearly describing the border-based approach for discovery of TGEPs, we formalize the notion of interval-closed sets and provide the definition of a border as given by.[1, 2, 9, 12]

Property 4.1. `Interval Closed Sets`. Collections of sets $\mathcal{C}$ are said to be interval closed if: $\forall X, Z \in \mathcal{C}$; $\forall Y$ such that $X \subseteq Y \subseteq Z$, it also holds that $Y \in \mathcal{C}$. Such sets are also referred to as *convex* sets.

Definition 4.1. `Border`. *A border is an ordered pair* $< \mathcal{L}, \mathcal{R} >$*, (where* $\mathcal{L}$ *-* *`left-hand bound of the border`* *and* $\mathcal{R}$ *-* *`right-hand bound of the border`)* *if: (a) each of* $\mathcal{L}$ *and* $\mathcal{R}$ *is an antichain*[b] *collection of sets, and (b) each element of* $\mathcal{L}$ *is a subset of some element in* $\mathcal{R}$ *and each element of* $\mathcal{R}$ *is a superset of some element in* $\mathcal{L}$*.*

For example, a collection of sets $[\mathcal{L}, \mathcal{R}]$ may be represented by (or is said to have) a border $< \mathcal{L}, \mathcal{R} >$, where $[\mathcal{L}, \mathcal{R}] = \{Y \mid \exists X \in \mathcal{L}, \exists Z \in \mathcal{R} \text{ such that } X \subseteq Y \subseteq Z\}$.

For the purpose of relating Property 4.1 to Definition 4.1,[9] presents three main propositions: (a) the collection of all large item sets with respect to a minimum threshold (σ) is interval closed, (b) each interval-closed collection $\mathcal{C}$ of sets has a unique border $< \mathcal{L}, \mathcal{R} >$, where $\mathcal{L}$ is the collection of minimal item sets in $\mathcal{C}$ and $\mathcal{R}$ is the collection of maximal item sets, and (c) the collection of large item sets with respect to a minimum threshold (σ) in a data set has a *left-rooted* border.

To clarify, a border $< \mathcal{L}, \mathcal{R} >$ is called *left-rooted* if $\mathcal{L}$ is a singleton set (i.e. the left-hand bound is $\{\emptyset\}$ and its right-hand bound is the collection of maximal item sets) and it is *right-rooted* if $\mathcal{R}$ is a singleton set. As a result of these propositions,[9] illustrates that an efficient maximal set pattern mining algorithm (i.e. Max-Miner) may be used to extract collections of large item sets from data sets which in-turn provide *left-rooted* borders for each data set with respect to a minimum support threshold.

All in all,[9] demonstrates how an efficient discovery of left-rooted borders from two data sets (i.e. $\mathcal{D}_1$ and $\mathcal{D}_2$) through Max-Miner algorithm, followed by a repetitive border differential procedure (implemented as BORDER-DIFF algorithm) on the two borders allows for extraction of emerging patterns. The overall algorithm is known as MBD-LLBORDER algorithm.

[b]A collection of sets $\mathcal{C}$ is an *antichain* if $\forall X, Y \in \mathcal{C}, X \nsubseteq Y$ and $Y \nsubseteq X$

4.2. *Border Representation of Gradual Item Sets*

It should be noted that the MBD-LLBORDER algorithm cannot be applied directly to the case of gradual item sets, since the nature of gradual item sets is slightly different from that of classical frequent item sets. Unlike frequent item sets, (in terms of tuple transactions and gradual items) gradual item sets in both cases deal with pairs and not singletons. We demonstrate this in Section 2.1.

As demonstrated in Section 2.1: (a) deriving the frequency support of a gradual pattern involves ordering tuples in concordant pairs; and (b) discovering any meaningful correlation knowledge among attributes of a data set involves extracting non-trivial gradual patterns composed of at least 2 gradual items. It should be remembered that the border representation of large item sets presented by[9] (see Section 4.1) is most suitable for classic frequent item sets that are each composed of singleton items.

For example, in the classic item set case a 4-*length* pattern $\{A, B, C, D\}$ may fully be decomposed into its 4 items: $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$. In the gradual item set case a 4-*length* pattern $\{(A, \uparrow), (B, \downarrow), (C, \uparrow), (D, \downarrow)\}$ at best may be decomposed into its 6 gradual items: $\{(A, \uparrow), (B, \downarrow)\}, \{(A, \uparrow), (C, \uparrow)\}, \{(A, \uparrow), (D, \downarrow)\}, \{(B, \downarrow), (C, \uparrow)\}, \{(B, \downarrow), (D, \downarrow)\}, \{(C, \uparrow), (D, \downarrow)\}$.

Nevertheless,[12] identifies two properties of gradual patterns that allow for border representation of gradual patterns. They are: (a) a collection of frequent gradual patterns is interval-closed, and (b) a collection of frequent gradual patterns may be represented as a left-rooted border $< \{\emptyset\}, \mathcal{R} >$, where $\mathcal{R}$ is the set of maximal gradual item sets in the collection broken down into gradual items of length 2.

With reference to the first identified property of gradual patterns, it derives explicitly from *anti-monotonicity* feature of gradual patterns. The anti-monotonicity property states that: *"no frequent gradual pattern containing n items can be built over an infrequent gradual pattern containing a subset of these n items."* For instance, if a maximal gradual pattern $\{(A, \uparrow), (B, \downarrow), (C, \uparrow)\}$ is frequent, then all subsets of this pattern are also frequent.[7,11]

Regarding the second identified property of gradual patterns, it derives from *pairings* that come with mining non-trivial gradual patterns. Consequently,[12] proposes a subsequent representation of maximal gradual item sets into its smaller gradual items of length 2. That is to say, a maximal gradual pattern of length k may be re-represented by a set of $k(k-1)/2$ gradual items of length 2.

4.3. *Border Representation of Temporal Gradual Item Sets*

Temporal gradual patterns (TGPs) (as described in Section 3) are simply gradual patterns that have been improved to indicate an estimated *time lag* among the gradual item sets. In fact[7] presents T-GRAANK approach which extends the GRAANK approach (proposed by[6]) for mining TGPs.

It should be clarified that the distinctive feature of approximated time lag of TGPs is majorly introduced by the *data set transformation* process. As an advantage, the transformation process allows for generation of multiple transformed data sets from the original timestamped data set using a specified *representativity* threshold.[7] The novel idea of harnessing this process so as to mine a single data set for *emerging TGPs* (or TGEPs) is quite interesting. In fact, we propose Algorithm 1, BT-GRAANK stands for MBD-LLBORDER Temporal GRAdual rANKing.

Algorithm 1: *BT-GRAANK* algorithm

```
Input  : $D-$ data set, $refCol-$ reference column, $\sigma-$ minimum support, $\delta-$
         minimum representativity
Output: $TGEPs-$ TGEPs represented as borders
1 $\mathcal{D}^*, T_d^* \leftarrow$ transform $(D, \delta, refCol)$ ; /* $\mathcal{D}-$ transformed data set, $T_d-$ time
    differences, $*$ denotes multiple */
2 $tgps^* \leftarrow$ extract-tgps $(\mathcal{D}^*, T_d^*, \sigma)$;
3 $leftBdr^* \leftarrow$ maximal $(tgps^*)$;    /* maximal TGPs as left-rooted borders */
4 $TGEPs \leftarrow$ MBD-LLBORDER $(leftBdr^*)$;
5 return $TGEPs$;
```

(1) Build multiple transformed data sets $\mathcal{D}'_g, \mathcal{D}''_g, ..., \mathcal{D}^*_g$ from a timestamped data set $\mathcal{D}_g$ from a specified representativity threshold δ (see Section 3).
(2) Extract all TGPs from each transformed data set w.r.t a minimum support threshold σ.
(3) Construct border representations of all maximal TGPs from the transformed data sets as described in Section 4.2.
(4) Apply a modified MBD-LLBORDER algorithm (using modified a union operator) to the borders obtained in Step 3 (two borders at a time).

It is important to note that this border-based strategy is an efficient technique for mining TGEPs; however, the search space grows exponentially with respect to the number of attributes.[7,14] Therefore, this strategy is not suitable for discovering patterns in huge data sets with a large number of attributes.

5. Ant-based Discovery of TGEPs

In this section, first we introduce an alternative approach that is based on ant colony optimization (ACO) for mining gradual patterns (GPs) and temporal gradual patterns (TPGs), and second we extend this approach in order to mine for temporal gradual emerging patterns (TGEPs).

5.1. *Ant Colony Optimization*

We propose an ACO strategy that uses a probabilistic approach to efficiently generate gradual item set candidates from a data set's attributes. ACO, as originally described by,[8] is a general-purpose heuristic approach for optimizing various combinatorial problems. It exploits the behavior of a colony of artificial ants in order to search for approximate solutions to discrete optimization problems.

ACO imitates the positive feedback reinforcement behavior of biological ants as they search for food: where the more ants following a path, the more chemical pheromones are deposited on that path and, the more appealing that path becomes for being followed by other ants.[8, 15–19]

According to,[20] ACO utilizes a set of artificial ants to probabilistically contrive solutions $\mathcal{S}$ through a collective memory, *pheromones* stored in matrix $\mathcal{T}$, together with a problem specific heuristic η. In this section, we will consider one variant of ACO called '`MAX-MIN` ant system' presented by[21] for probabilistic generation of gradual item set candidates.

5.2. *ACO for Gradual Pattern Mining*

The principal aim of gradual pattern mining approaches is to extract (if possible maximal) gradual item sets whose support surpass a specified threshold. It can be shown that a heuristic approach can generate, with extremely high efficiency, maximal gradual item set candidates whose probability of being valid is high.[22, 23] In this paper, we present an *ant*-based approach that guides artificial ants to find highly probable maximal candidates.

In order to apply ACO to the problem of GP mining, we need to define: (1) a suitable representation of the gradual item set candidate generation problem, (2) a probabilistic rule $\mathcal{P}$ for generating solutions $\mathcal{S}_n$, (3) a technique for updating the pheromone matrix $\mathcal{T}_{a,j}$, (4) a convergence proof for confirming that this approach finds an optimal gradual pattern from a data set.

Representation of the problem. In order to represent GP mining problem as a combinatorial problem, we take the position that a gradual item set may also be known as a `pattern solution`. In that case all possible gradual item set solutions ($\mathcal{S}_n$) are admissible and can be generated based on the pheromone matrix ($\mathcal{T}_{a,j}$); therefore, a *tabu* list is not necessary. To clarify, a *tabu* list consists of solution candidates that violate the admissible conditions.[17, 24, 25] Notwithstanding, all the gradual item sets in a generated solution will be evaluated and the solution updated with only valid item sets.

For instance, let {*Rain, Wind, Temperature*} be the attributes of a numeric data set. Given minimum specified support threshold (σ), for every attribute there exists only 3 gradual item set possibilities: increase ($\uparrow$), decrease ($\downarrow$), or irrelevant ($\times$). Therefore, examples of maximal *pattern solutions* may be: $\{Rain\uparrow, Wind\uparrow, Temperature\downarrow\}$, $\{Rain\uparrow, Wind\downarrow, Temperature\downarrow\}$.

Probabilistic rule. First we mention that initially there exists an equal chance for any attribute A (of data set $\mathcal{D}$) to either increase ($\uparrow$) or decrease ($\downarrow$) or be irrelevant ($\times$). As the algorithm acquires more knowledge about valid patterns, the possibilities of the 3 options are adjusted accordingly. For this reason, the probability rule $\mathcal{P}(A^*)$ (where $* \in \{\uparrow, \downarrow, \times\}$) is given by calculating the proportions of its artificial pheromones p_{a*} as shown in Equations (5), (6), (7).

$$\mathcal{P}(A^{\uparrow}) = \begin{cases} 1 & if\ 0 < k \leqslant \frac{p_{a\uparrow}}{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))} \\ 0 & otherwise \end{cases} \tag{5}$$

$$\mathcal{P}(A^{\downarrow}) = \begin{cases} 1 & if\ \frac{p_{a\uparrow}}{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))} < k \leqslant \frac{((p_{a\uparrow})\ +\ (p_{a\downarrow}))}{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))} \\ 0 & otherwise \end{cases} \tag{6}$$

$$\mathcal{P}(A^{\times}) = \begin{cases} 1 & if\ \frac{((p_{a\uparrow})\ +\ (p_{a\downarrow}))}{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))} < k \leqslant \frac{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))}{((p_{a\uparrow})\ +\ (p_{a\downarrow})\ +\ (p_{a\times}))} \\ 0 & otherwise \end{cases} \tag{7}$$

Pheromone update. We define an artificial pheromone matrix as shown in Equation (8). The matrix contains knowledge about pheromone proportions of the 3 gradual options for each attribute. In fact, it is through the pheromone matrix that the algorithm learns how to generate highly probably valid gradual item set candidates.

$$\mathcal{T}_{a,j} = q \times 3 \tag{8}$$

where q: number of attributes, $a = 1, ..., q$ and, $j \in \{+, -, \times\}$

At the beginning all the artificial pheromones p_{aj} in the pheromone matrix $\mathcal{T}_{a,j}$ are initialized to 1, then they are updated as follows:

- Every generated gradual item set solution is evaluated and only valid solutions are used to update the artificial pheromone matrix. Invalid solutions are stored with aim of using them to reject their supersets.
- In a given valid solution, each gradual item set is used to update the corresponding artificial pheromone p_{aj} (where j is either $\uparrow$ or $\downarrow$) using Equation (9). It is essential to point out that since the attribute possibilities do not cancel each other out, we ignore the evaporation factor.
- Again using the same valid solution, for every attribute that does not have a gradual item set appearing in the solution - we update the corresponding irrelevant pheromone p_{aj} (where j is $\times$) using Equation (9).

$$p_{aj} = p_{aj} + sup(Sol) \tag{9}$$

where $sup(Sol)$ is the frequency support of the generated pattern solution

Convergence proof. In the case of GP mining we wish to find an *optimal solution* (which is a valid maximal gradual item set) that updates the matrix such that the preceding generated solutions are either subsets of or similar to the optimal solution. Such a characteristic may also be referred to as a *Convergence Property.*

The study of[26] illustrates a *convergence proof* that applied directly to `MAX--MIN` *Ant System.* The proof holds that:

"for any small constant $\epsilon > 0$ and for a sufficiently large number of algorithm iterations t, the probability of finding an optimal solution at least once is $\mathcal{P}*(t) \geq 1 - \epsilon$ and that the probability tends to 1 for $t \to \infty$."

Further,[26] establishes that after an *optimal solution* has been found, it takes a limited number of algorithm iterations for the pheromone trails that belong to the found optimal solution to grow higher than any other pheromone trail. With regard to GP mining, this implies that the values of the pheromone matrix will no longer change significantly after such iterations. Therefore, we propose that this *convergence property* can be harnessed to determine the limit of algorithm iterations for generating gradual item set candidates.

Not only may the ACO strategy be applied to the case of GP mining but also to the case of TGP mining. As described in Section 3, TGP mining involves two main processes: (1) transforming a timestamped data set into multiple data sets using a specified representativity threshold, and (2) applying a modified GRAANK algorithm that extracts gradual patterns together with an approximated time lag. Therefore, we substitute the modified GRAANK algorithm for a modified ACO-based algorithm.

5.3. *Growth-rate Manipulation for Mining TGEPs*

It should be noted that applying the proposed ACO-based approach on a data set extracts numerous GPs or TGPs, but only one pheromone matrix for every data set or transformed data set respectively (see Equation (8)). As illustrated in the Section 5.2, the values of the matrix depend on the patterns extracted since each valid pattern increments it by its support (see Equation (9)). In that case, a single matrix cumulatively stores support values of all extracted patterns. The matrix can be normalized using the number of algorithm iterations determined by the *convergence property.*

The definitions given in Section 2.2 about emerging patterns (EPs) and their growth-rate validates the idea that: if two data sets each provide its *support-based* pheromone matrix, then dividing the two matrices element-wisely generates a *growth-rate matrix.* Through division, the growth-rate matrix reduces any irrelevant EPs to zero but allows for construction of relevant EPs. To this end, there exists no reason to keep the gradual patterns previously extracted.

Example 5.1. Let $\{Rain,\ Wind,\ Temperature\}$ be attributes of data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. $\mathcal{P}_1$ and $\mathcal{P}_2$ (shown in Figure 1) be the pheromone matrices of data sets $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. A growth-rate matrix in favor of $\mathcal{P}_1$ is shown in Figure 2. As can be deduced from the growth-rate matrix in Figure 2, we may construct a GEP $\{(Rain, \uparrow), (Wind, \downarrow)\}$ with a growth-rate of at least 1.5 from data set $\mathcal{D}_1$ to $\mathcal{D}_2$.

$$\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & .8 & 0 & 0 \\
\text{Wind} & 0 & .9 & 0 \\
\text{Temp} & 0 & 0 & .85 \\ \hline
\end{array}
\qquad
\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & .4 & 0 & 0 \\
\text{Wind} & 0 & .6 & 0 \\
\text{Temp} & 0 & .75 & 0 \\ \hline
\end{array}$$

(a) (b)

Figure 1: (a) normalized support values of $\mathcal{P}_1$, and (b) normalized support values of $\mathcal{P}_2$

$$\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & 2 & 0 & 0 \\
\text{Wind} & 0 & 1.5 & 0 \\
\text{Temp} & 0 & 0 & \infty \\ \hline
\end{array}$$

Figure 2: Growth-rate matrix from pheromone matrix $\mathcal{P}_1$ to $\mathcal{P}_2$

In this paper, our main aim is to extend the ACO strategy to the case of TGEP mining and compare its performance to the border-based strategy. Although mining TGPs using an ACO-based approach is easily achievable (see Section 5.2), constructing TGEPs from growth-rate matrices is difficult since the matrices do not provide information about associated time-lags.

For this reason, we propose an additional *time-lag matrix* which is updated with approximated time-lags of validated patterns every time the *pheromone matrix* is updated with support values of these patterns. Finally, the combined content of the growth-rate matrix and the time-lag matrix allow for the construction of TGEPs.

$$\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & .8 & 0 & 0 \\
\text{Wind} & 0 & .8 & 0 \\
\text{Temp} & 0 & 0 & .8 \\ \hline
\end{array}
\qquad
\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & +2mins & 0 & 0 \\
\text{Wind} & 0 & +2mins & 0 \\
\text{Temp} & 0 & 0 & 0 \\ \hline
\end{array}$$

(a) (b)

$$\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & .4 & 0 & 0 \\
\text{Wind} & 0 & .4 & 0 \\
\text{Temp} & 0 & 0 & .4 \\ \hline
\end{array}
\qquad
\begin{array}{l|ccc|}
 & \uparrow & \downarrow & \times \\ \hline
\text{Rain} & +6mins & 0 & 0 \\
\text{Wind} & 0 & +6mins & 0 \\
\text{Temp} & 0 & 0 & 0 \\ \hline
\end{array}$$

(c) (d)

Figure 3: (a) pheromone matrix for $\mathcal{D}'_g$, (b) time-lag matrix for $\mathcal{D}'_g$, (c) pheromone matrix $\mathcal{D}''_g$, and (d) time-lag matrix for $\mathcal{D}''_g$,

Example 5.2. Let $TGP_1 = \{(Rain^{\uparrow}, Wind^{\downarrow})_{\approx +2mins}, sup = 0.8\}$ be extracted from a transformed data set $\mathcal{D}'_g$, and $TGP_2 = \{(Rain^{\uparrow}, Wind^{\downarrow})_{\approx +6mins}, sup = 0.4\}$ be extracted from a transformed data set $\mathcal{D}''_g$. Figure 3 shows the support pheromone matrices and the corresponding time-lag matrices for the transformed data sets.

A growth-rate matrix in favor of the support-based pheromone matrix of transformed data set $\mathcal{D}'_g$ is shown in Figure 4. This growth-rate matrix is mapped element-wisely onto time-lag matrices of $\mathcal{D}'_g$ and $\mathcal{D}''_g$ in order to eliminate irrelevant time-lag elements; in this case none of the elements are irrelevant.

$$\begin{array}{r|ccc|}
 & \uparrow & \downarrow & \times \\
\hline
\text{Rain} & 2 & 0 & 0 \\
\text{Wind} & 0 & 2 & 0 \\
\text{Temp} & 0 & 0 & 2 \\
\hline
\end{array}$$

Figure 4: Growth-rate matrix from pheromone matrix of data set $\mathcal{D}'_g$ to $\mathcal{D}''_g$

As can be deduced by combining growth-rate matrix in Figure 4 and time-lag matrices in Figure 3 (b) and (d), we may construct a TGEP $\{(Rain, \uparrow), (Wind, \downarrow)\}$ with a growth-rate of 2 after approximately 4 minutes. All things considered, we propose Algorithm 2, TRENC stands for Temporal gRadual Emerging aNt Colony optimization.

Algorithm 2: TRENC algorithm

```
Input  : $D-$ data set, $refCol-$ reference column, $\sigma-$ minimum support, $\delta-$
         minimum repsentatitvity
Output: $TGEPs-$ TGEPs in JSON format
1 $\mathcal{D}^*, T_d^* \leftarrow$ transform $(D, \delta, refCol)$ ; /* $\mathcal{D}-$ transformed data set, $T_d-$ time
    differences, $*$ denotes multiple */
2 $\mathcal{P}^*, \mathcal{T}^* \leftarrow$ aco-matrices $(D^*, T_d^*, \sigma)$ ; /* $\mathcal{P}-$ pheromone matrix, $\mathcal{T}-$ time-lag
    matrix */
3 $\mathcal{G}^* \leftarrow$ gen-growthrate ($\mathcal{P}[x]$, $\mathcal{P}^*$);        /* $\mathcal{G}-$ growth-rate matrix, $x-$
    user-specified w.r.t preferred transformed data set */
4 $TGEPs \leftarrow$ construct $(\mathcal{G}^*, \mathcal{T}^*)$;
5 return $TGEPs$;
```

(1) Build multiple transformed data sets $\mathcal{D}'_g, \mathcal{D}''_g, ..., \mathcal{D}^*_g$ from a timestamped data set $\mathcal{D}_g$ using a specified representativity threshold δ (see Section 3).
(2) From each transformed data set, build a normalized support pheromone matrix along with corresponding time-lag matrices.
(3) Generate growth-rate matrices from the pheromone matrices obtained in Step 2 (two pheromone matrices at a time).
(4) Combine each growth-rate matrix with the two corresponding time-lag matrices to construct TGEPs.

6. Experiments

In this section, we implement the border-based BT-GRAANK algorithm (described in Section 4) and the ant-based TRENC algorithm (described in Section 5) for mining TGEPs and analyze their computational performances. All experiments were conducted on a (High Performance Computing) HPC platform **Meso@LR**[c]. We used one node made up of 112 cores and 128GB of RAM.

6.1. *Source Code*

The `Python` source code of our proposed algorithms are available at our GitHub repository: `https://github.com/owuordickson/trenc.git`.

6.2. *Data Set Description*

Table 5 shows the features of the data sets used in the experiments for evaluating the computational performance of our proposed algorithms.

Table 5: Experiment data sets

Data set	#tuples	#attributes	Timestamped	Domain	Origin
Buoys (Directio)	6121	21	Yes	Coastline	[27]
Power Consumption	10001	9	Yes	Electrical	[28]

The 'Power Consumption' data set, obtained from `UCI Machine Learning Repository`,[28] describes the electric power consumption in one household (located in Sceaux, France) in terms of active power, voltage and global intensity with a one-minute sampling rate between December 2006 and November 2010.

The 'Directio' data set is one of 4 data sets obtained from OREMES's data portal[d] that recorded swell sensor signals of 4 buoys near the coast of the Languedoc-Roussillon region in France between 2012 and 2019.[27] These data sets can be retrieved from: `https://github.com/owuordickson/trenc/tree/master/data`.

6.3. *Experiment Results*

In this section, we present the results of our experimental study on the two data sets using our proposed algorithms. These results reveal that the two algorithms behave differently when applied on different data sets (especially if they vary in number of attributes). We use these results to analyze and compare the computational efficiency and parallel efficiency of the algorithms as presented in Sub-section 6.3.1 and Sub-section 6.3.2 respectively. We discuss these results in Section 6.4. All the

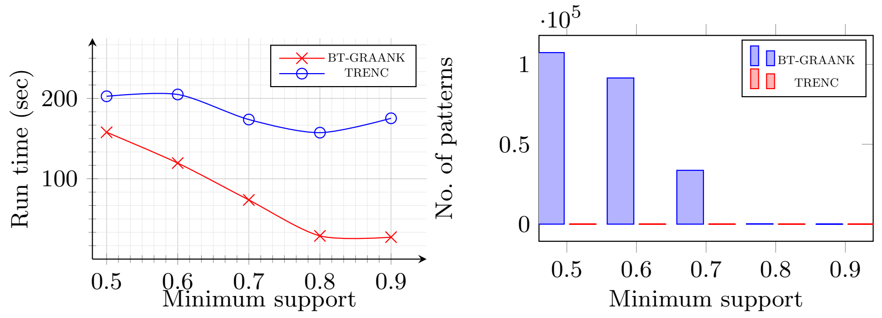
[c]`https://meso-lr.umontpellier.fr`
[d]`https://data.oreme.org`

experiment results can be obtained from: `https://github.com/owuordickson/meso-hpc-lr/tree/master/results/tgeps/112cores`.

6.3.1. *Comparative Experiments: computational efficiency*

This experiment compares the run-time and number of temporal gradual emerging patterns (TGEPs) extracted by TRENC and BT-GRAANK from data sets *UCI* and *Directio*. We mention that the minimum representativity threshold is set at `0.99` so that very few transformations are applied on the original data sets, which improves the quality of TGEPs (see Section 3).

`UCI data set: #attributes=9/#tuples=10K/cores=56/min-rep=0.99`



(a) (b)

Figure 5: UCI data set: (a) plot of run time against minimum support threshold and, (b) bar graph of number of patterns against minimum support threshold.

Figure 5 (a) shows run-time and, Figure 5 (b) shows the number of TGEPs extracted by TRENC and BT-GRAANK algorithms when applied on the *UCI* data set. We observe that the run-time of BT-GRAANK (blue curve) is lower than that of TRENC (red curve) and it reduce significantly (as well as the number of TGEPs) as support threshold is increased. For the case of TRENC, the run-time and number of TGEPs are almost constant.

Figure 6 (a) shows run-time performance and, Figure 6 (b) shows the number of TGEPs extracted by TRENC and Border-TGRAANK algorithms when applied on the *Directio* data set. In this instance, BT-GRAANK (in comparison to TRENC) has the highest run-time (which reduces) and fewest TGEPs as the support threshold is increased. Again, the run-time and number of extracted of TGEPs are almost constant for the case of TRENC.

It should be remembered that support threshold plays an important role in determining the quality and quantity of extracted frequent patterns (see Section 2.1). The higher the threshold, the higher the quality of the patterns; consequently, triv-

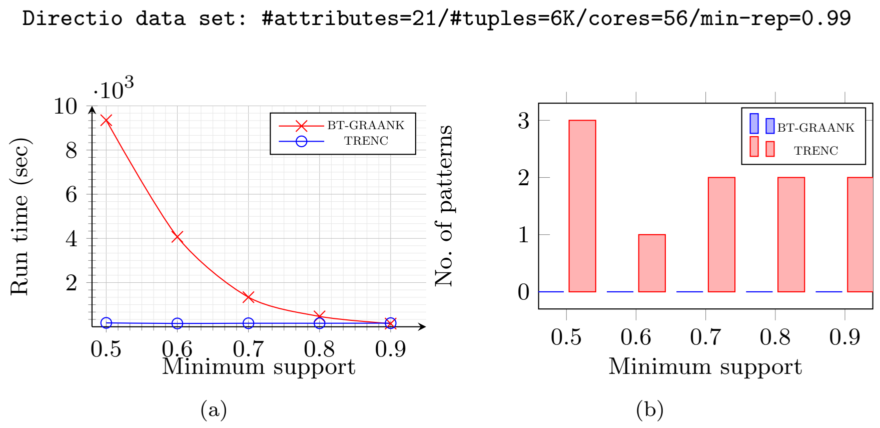


Figure 6: Directio data set: (a) plot of run time against minimum support threshold and, (b) bar graph of number of patterns against minimum support threshold.

ial patterns are ignored. This explains why the run-time and number of TGEPs reduce significantly for the case of BT-GRAANK. Concerning TRENC, it seems this threshold has little effect on the quantity of TGEPs.

6.3.2. *Comparative Experiments: parallel efficiency*

This experiment compares the run-time of BT-GRAANK and TRENC algorithms against different number of CPU cores on data sets *UCI* and *Directio*. In Figure 7, the run-time of both algorithms reduce as the number of cores increase.

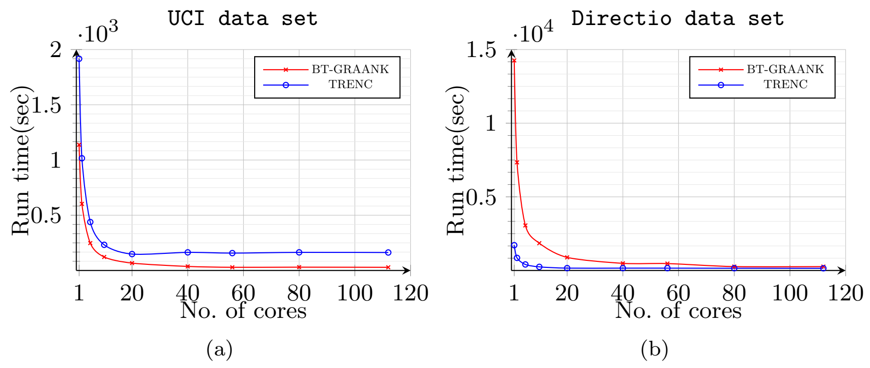


Figure 7: Plot of run time versus no. of cores on data sets (a) UCI and, (b) Directio

We use these results to analyze their multiprocessing behavior using the `speedup` and `parallel efficiency` performance measures. (1) `Speedup S(n)` may be defined as: *"the ratio of the execution time of a single processor to the execution time of n processors"* ($S(n) = T_1/T_n$). (2) `Parallel efficiency E(n)` may be defined as: *"the average utilization of n processors"* ($E(n) = S(n)/n$).[29]

`UCI data set: #attributes=9/#tuples=10K/min-rep=0.99/min-sup=0.8`

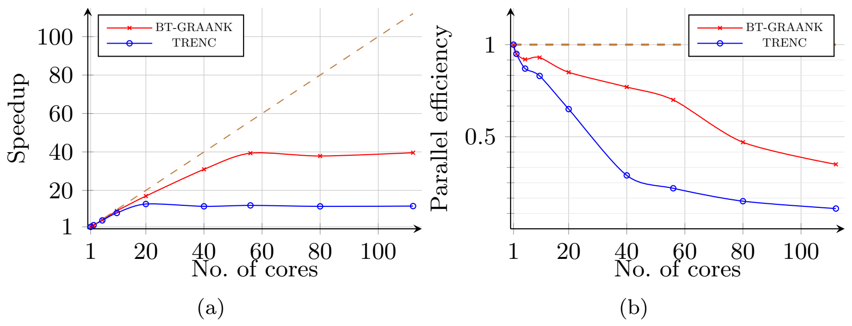


Figure 8: UCI data set: (a) plot of speed up versus number of cores (b) plot of parallel efficiency versus number of cores

In Figure 8 (a), we observe that BT-GRAANK (in comparison to TRENC) has a highest Speedup and, in Figure 8 (b) has a highest parallel efficiency when both are applied on the UCI data set. Again, in Figure 9 (a), we observe that BT-GRAANK (in comparison to TRENC) has a highest Speedup and, in Figure 9 (b) has a highest parallel efficiency when both are applied on the Directio data set. However, it should be observed, from Figure 7 (b), that the run-time of BT-GRAANK is higher than that of TRENC on data set Directio.

`Directio data set: #attributes=21/#tuples=6K/min-rep=0.99/min-sup=0.8`

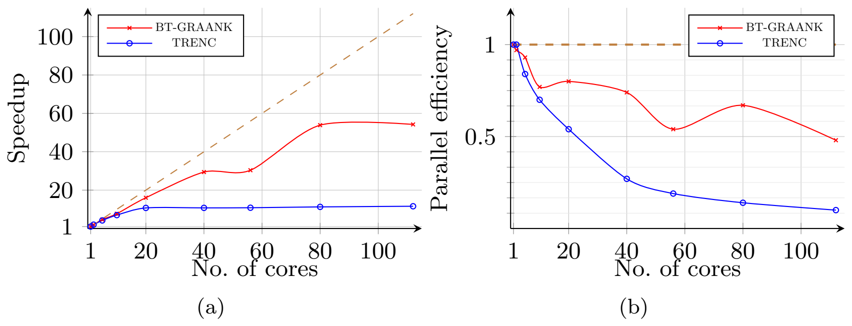


Figure 9: Directio data set: (a) plot of speed up versus number of cores (b) plot of parallel efficiency versus number of cores

6.3.3. *Consistent Temporal Gradual Emerging Patterns (TGEPs)*

In this section, we present the consistent TGEPs (see Section 3 for Definition) extracted from the two data sets described above.

Table 6: Consistent TGEPs extracted from data sets UCI and Directio

Data set	Consistent TGEPs
Buoys (Directio)	$\{(Tz, \uparrow), (Hm0, \uparrow)\}$: growth-rate 1.0 after every 2.5 hours
Power Consumption (UCI)	$\{(activepower, \uparrow), (voltage, \downarrow)\}$: growth-rate 1.04 after every 24 hours

6.4. *Discussion of Results*

6.4.1. *Computation Run-time Complexity*

Unlike BT-GRAANK, we observe that the run-time for TRENC on both data sets is almost constant. As described in Section 5.2 is based on a heuristic technique for efficiently generating maximal gradual item sets. Therefore, the run-time required for extracting patterns through this technique is determined by how long the pheromone matrix takes to converge. Again, this property enables TRENC to have a lower run-time than BT-GRAANK (see Section 4.3) when executed on a data set with a large number of attributes. For example since data set Directio has more attributes than data set UCI, there is a significant increase in run-time for the case BT-GRAANK and a relatively small change for the case of TRENC (see Figure 5 (a) and Figure 6 (a).

Finally, we observe that BT-GRAANK has a higher Speedup and parallel efficiency than TRENC for both data sets as shown in Figure 8 and Figure 9. Majorly, this is due the fact that TRENC's run-time is almost constant despite the variations in support threshold and number of cores. This implies the advantage that TRENC can extract high quality TGEPs using few processors and at any support threshold.

6.4.2. *Extracted TGEPs*

We observe that BT-GRAANK extracts more TGEPs than TRENC from the UCI data set. It should be emphasized that BT-GRAANK identifies borders from two maximal items. For this reason, numerous borders are used to construct few TGEPs (see Section 4.3). In fact, we discover that both algorithms identify similar consistent TGEPs from the UCI data set as shown in Table 6.

7. Conclusion

In this work, first we have introduced the concept of temporal gradual emerging patterns; second, we have proposed two strategies for mining temporal gradual emerging patterns; third, we have proposed an experimental computational performance comparison including a parallel implementation of these two approaches on a HPC supercomputer. Finally, we recommend ant-based strategy as the most suitable strategy for mining temporal gradual emerging patterns especially when dealing with huge data sets having large numbers of attributes.

Acknowledgments

This work is part of a Ph.D. thesis and the authors would like to thank the French Government through the office of Co-operation and Cultural Service (Kenya) and the office of Campus France (Montpellier) for their involvement in creating the opportunity for this work to be produced. This work has been realized with the support of the High Performance Computing Platform `MESO@LR`, financed by the Occitanie / Pyrénées-Méditerranée Region, Montpellier Mediterranean Metropole and the University of Montpellier.

References

1. G. Dong, X. Zhang, L. Wong and J. Li, "Caep: Classification by aggregating emerging patterns", in *Discovery Science*, eds. S. Arikawa and K. Furukawa (Springer Berlin Heidelberg, Berlin, Heidelberg, 1999), pp. 30–42.
2. J. Li, G. Dong and K. Ramamohanarao, "Making use of the most expressive jumping emerging patterns for classification", *Knowledge and Information Systems* **3** (2001) 131–145.
3. R. Kotagiri and J. Bailey, "Discovery of emerging patterns and their use in classification", in *AI 2003: Advances in Artificial Intelligence*, eds. T. T. D. Gedeon and L. C. C. Fung (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003), pp. 1–11.
4. F. Berzal, J. C. Cubero, D. Sanchez, M. A. Vila and J. M. Serrano, "An alternative approach to discover gradual dependencies", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **15** (2007) 559–570.
5. L. Di-Jorio, A. Laurent and M. Teisseire, "Mining frequent gradual itemsets from large databases", (Springer-Verlag, Berlin, Heidelberg, 2009), pp. 297–308.
6. A. Laurent, M.-J. Lesot and M. Rifqi, "Graank: Exploiting rank correlations for extracting gradual itemsets", in *Proceedings of the 8th International Conference on Flexible Query Answering Systems* (Springer-Verlag, Berlin, Heidelberg, 2009), FQAS '09, pp. 382–393.
7. D. Owuor, A. Laurent and J. Orero, "Mining fuzzy-temporal gradual patterns", in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (IEEE, New York, NY, USA, 2019), pp. 1–6.
8. M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **26** (1996) 29–41.
9. G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences", in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, NY, USA, 1999), KDD '99, pp. 43–52.
10. G. Dong and J. Li, "Mining border descriptions of emerging patterns from dataset pairs", *Knowledge and Information Systems* **8** (2005) 178–202.

11. L. Di Jorio, A. Laurent and M. Teisseire, "Fast extraction of gradual association rules: A heuristic based method", in *Proceedings of the 5th International Conference on Soft Computing As Transdisciplinary Science and Technology* (ACM, New York, NY, USA, 2008), CSTST '08, pp. 205–210.
12. A. Laurent, M.-J. Lesot and M. Rifqi, "Mining emerging gradual patterns", in *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT-15)* (Atlantis Press, 2015).
13. J. Li and L. Wong, "Emerging patterns and gene expression data", in *Proceedings of the 12th International Conference on Genome Informatics* (Universal Academy Press, Tokyo, Japan, 2001), pp. 3–13.
14. A. García-Vico, C. Carmona, D. Martín, M. García-Borroto and M. del Jesus, "An overview of emerging pattern mining in supervised descriptive rule discovery: taxonomy, empirical study, trends, and prospects", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **8** (2018) 1–22.
15. C. A. Silva, T. A. Runkler, J. M. Sousa and R. Palm, "Ant colonies as logistic processes optimizers", in *International Workshop on Ant Algorithms*, eds. M. Dorigo, G. Di Caro and M. Sampels (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002), pp. 76–87.
16. C. Blum, "Ant colony optimization: Introduction and recent trends", *Physics of Life Reviews* **2** (2005) 353 – 373.
17. T. A. Runkler, "Ant colony optimization of clustering models", *International Journal of Intelligent Systems* **20** (2005) 1233–1251.
18. M. Dorigo and M. Birattari, *Ant Colony Optimization* (Springer US, Boston, MA, 2010), pp. 36–39.
19. M. Dorigo and T. Stützle, *Ant Colony Optimization: Overview and Recent Advances* (Springer International Publishing, Cham, 2019), pp. 311–351.
20. S. A. Hartmann and T. A. Runkler, "Online optimization of a color sorting assembly buffer using ant colony optimization", in *Operations Research Proceedings 2007*, eds. J. Kalcsics and S. Nickel (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), pp. 415–420.
21. T. Stützle and H. H. Hoos, "Max–min ant system", *Future generation computer systems* **16** (2000) 889–914.
22. B. Kalpana and R. Nadarajan, "Incorporating heuristics for efficient search space pruning in frequent itemset mining strategies", *Current Science* **94** (2008) 97–101.
23. H. Li, Y. Zhang, N. Zhang and H. Jia, "A heuristic rule based approximate frequent itemset mining algorithm", *Procedia Computer Science* **91** (2016) 324 – 333.
24. P. C. Pinto, A. Nagele, M. Dejori, T. A. Runkler and J. M. C. Sousa, "Learning of bayesian networks by a local discovery ant colony algorithm", in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (2008), pp. 2741–2748.
25. H. Chen, G. Tan, G. Qian and R. Chen, "Ant colony optimization with tabu table to solve tsp problem", in *2018 37th Chinese Control Conference (CCC)* (2018), pp. 2523–2527.
26. T. Stutzle and M. Dorigo, "A short convergence proof for a class of ant colony optimization algorithms", *IEEE Transactions on Evolutionary Computation* **6** (2002) 358–365.
27. F. Bouchette, "OREME: the coastline observation system", 2019.
28. D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository", 2017.
29. D. L. Eager, J. Zahorjan and E. D. Lazowska, "Speedup versus efficiency in parallel systems", *IEEE Transactions on Computers* **38** (1989) 408–423.