A metaheuristic approach for mining gradual patterns

Dickson Odhiambo Owuor, Thomas Runkler, Anne Laurent

A Metaheuristic Approach for Mining Gradual Patterns

Dickson Odhiambo Owuor[a,*], Thomas Runkler[b], Anne Laurent[c]

[a]*SCES Strathmore University, Nairobi, Kenya*
[b]*Siemens AG, Munich, Germany*
[c]*LIRMM Univ Montpellier, CNRS, Montpellier, France*

Abstract

Swarm intelligence is a discipline that studies the collective behavior that is produced by local interactions of a group of individuals with each other and with their environment. In Computer Science domain, numerous swarm intelligence techniques are applied to optimization problems that seek to efficiently find best solutions within a search space. Gradual pattern mining is another Computer Science field that could benefit from the efficiency of swarm based optimization techniques in the task of finding gradual patterns from a huge search space. A gradual pattern is a rule-based correlation that describes the gradual relationship among the attributes of a data set. For example, given attributes $\{G, H\}$ of a data set a gradual pattern may take the form: *"the less G, the more H"* (simply denoted as $\{G^{\downarrow}, H^{\uparrow}\}$). In this paper, we propose a numeric encoding for gradual pattern candidates that we use to define an effective search space. In addition, we present a systematic study of several meta-heuristic optimization techniques as efficient solutions to the problem of finding gradual patterns using our search space.

Keywords: Genetic Algorithm, Local Search, Particle Swarm Optimization, Search Space, Swarm Intelligence, Random Search

---

*Corresponding author

Email addresses: `dowuor@strathmore.edu` (Dickson Odhiambo Owuor),
`thomas.runkler@siemens.com` (Thomas Runkler), `anne.laurent@umontpellier.fr` (Anne Laurent)

1. Introduction

Swarm-based optimization algorithms, most of which mimic the collective intelligence of groups of simple agents, have demonstrated excellent efficiency in solving many optimization problems [1]. According to [2], swarm-based optimization algorithms belong to the subcategory of metaphor-based algorithms in the class of metaheuristic algorithms.

Metaheuristic algorithms are computational intelligence models that are used to solve complex optimization problems. In this paper, we focus on the task of generating gradual pattern candidates as an optimization problem. Gradual pattern mining is a data mining field in Computer Science that deals with describing attributes/variables/features of data sets using rules such as: *"the higher the temperature, the less vehicle traffic"* . We provide a detailed description of gradual pattern mining in Section 2.

Gradual pattern (GP) mining has many applications especially in domains that seek to find correlational knowledge about variables of a data set. For instance, in [3], GP mining is applied in the discovery of the gradual relationship between numerical building variables in order to improve building efficiency.

One key challenge experienced in GP mining is in the task of identifying candidate rules. This involves combining different kinds of variations (increasing and decreasing) for each rule that correlates two or more variables [4, 5]. However, the task of identifying these candidates may be modelled into a search space problem onto which metaheuristic algorithms may be applied.

In this paper, we propose a search space such that each point is a numeric digit that encodes a GP candidate. Our experiment results show that our numeric encoding produces a lean search space metaheuristic algorithms to explore. The results also rank the computational efficiency of these algorithms in the task of finding *best* GP candidates from the search space.

The main contributions of this study are described as follows:

- We propose a numeric encoding that allows us to build a lean search space for GP candidates.

2

- We develop metaheuristic algorithms and compare their performances in exploring search spaces based on our numeric encoding and those based on other encodings.

2. Background

In Computer Science, a search space may be described as *"an expanse that is defined by the set of all feasible solutions through which an algorithm may iterate"* [6, 7]. In GP mining, a search space (known to be a lattice) holds all possible candidates that may be validated to be relevant GPs.

The subject of mining GPs (also referred to as gradual rules) has been studied by many researchers. To begin with, there exists different notions of gradual rules. Of particular interest, is the notion of applying Rescher-Gaines implication in order to measure gradualness. For instance, given two attributes A_1 and A_2, fuzzy sets M and N defined on A_1 and A_2 respectively. If $M(A_1)$ be the membership degree of A_1 in M, then the rule can be expresses as *"A_1 is in M implies A_2 is in N"*. In such expressions, we use Rescher-Gaines implication:

$$A_1 \rightarrow_{RG} A_2 = \begin{cases} 1 & \text{if } M(A_1) \leq N(A_2) \\ 0 & \text{else} \end{cases} \qquad (1)$$

Starting from the above notions, [8] formalizes 2 main kinds of gradual rules of the form *"the more/less A_1 is in M, the more/less A_2 is in N"*. The first kind (known as pure gradual rules), verifies a rule by evaluating properties between individual objects. The second kind (known as gradual dependence) verifies a rule by comparing the properties between objects. In this paper, we deal with pure gradual rules.

For the purpose of facilitating comprehension of this paper, we use a sample data set with 3 attributes and 4 objects or tuples (as shown in Table 1) to introduce GP mining. For instance, using these 3 attributes $\{age, sessions, marks\}$ we may formulate approximately 20 GP candidates like $\{age^\uparrow, sessions^\downarrow\}$, $\{sessions^\downarrow, marks^\uparrow\}$, $\{age^\uparrow, sessions^\downarrow, marks^\uparrow\}$ among others.

3

Table 1: A sample data set showing details of students who took part in a graded course.

age	sessions	marks
23	2	55
32	4	64
40	5	78
25	5	48

To clarify, only GP candidates whose computed *frequency support* exceed a specified *user-defined threshold* may be verified as relevant GPs [9, 10]. Generally, frequency support may be defined as: *"the proportion of ordered objects in a data set that verify a particular GP"*. For instance, the GP candidate $\{sessions^{\uparrow}, marks^{\uparrow}\}$ is verified by the first 3 objects (out of the 4 objects) of the data set (shown in Table 1) if they are taken in ascending order.

This is due to the fact that the object values of the attributes $\{sessions, marks\}$ subsequently increase in the first 3 objects with the exception of the fourth object values. Therefore, in this case the frequency support may be computed to be 3/4 or 0.75.

In view of this example, it is noticeable that the number of GP candidates is directly proportional to the number of a data set's attributes. According to [4, 10] (in GP mining), deterministic algorithms that have to find and validate each GP candidate are computationally overwhelmed when applied on data sets with large attribute sizes.

However, an effective search space $\mathcal{S}_{gp}$ may be defined such that every point in the search space represents a possible GP candidate. In addition, it is possible to iterate through the search space using a non-deterministic algorithm to find candidates which are validated by computing their frequency support.

Definitions and Notations

We give some definitions of gradual patterns as provided in [5, 9, 11, 12, 13]. For example, let D be a data set with attributes $\{age, sessions, marks\}$ and objects $\{obj1, obj2, obj3, obj4\}$ (as shown in Table 2).

Table 2: A sample data set, D, showing details of participants who took part in a training.

obj#	age	sessions	marks
obj1	23	2	55
obj2	32	4	64
obj3	40	5	78
obj4	25	5	48

Definition 1.1. (Gradual Item). *A gradual item is a pair at^v: where at is an attribute and v is a gradual variation such that $v \in \{\uparrow, \downarrow\}$. $\uparrow$ denotes an increasing variation and $\downarrow$ denotes a decreasing variation.*

For example, $age^\uparrow$ can be interpreted as *"the higher the age"*.

Definition 1.2. (Gradual Pattern). *A gradual pattern GP is a set of gradual items such that $GP = \{at_1^v, ..., at_n^v\}$.*

For example, $\{age^\uparrow, sessions^\downarrow, marks^\uparrow\}$ is a GP that can be interpreted as *"the higher the age, the less session numbers, the higher the marks obtained."*

The quality of a gradual item or gradual pattern is measured by its *frequency support*. The frequency support of a gradual pattern $sup(GP)$ may be defined as: *"the proportion of object couples that respect the gradual variations given collectively by all the items in the pattern"*. For instance, let A be an attribute of data set $\mathcal{D}$ which has a total of k objects and o be an object in $\mathcal{D}$ such that $A(o)$ denotes the value A takes for o, (o, o') denotes an object pair in $\mathcal{D}'$ (which is a transaction data set derived from $\mathcal{D}$ and it holds the set of all object pair combinations). The gradual item $A^\downarrow$ holds if $\forall (o, o') \in \mathcal{D}', \ A(o) > A(o')$ [9, 14]. Therefore, support of a GP can be defined by the formula that follows.

$$sup(GP) = \frac{1}{|\mathcal{D}'|} \cdot |\{(o, o') \in \mathcal{D}'/A(o) * A(o')\}|, \qquad (2)$$

where $* \in \{<, >\}$. Given a user-specified threshold σ, a GP is said to be *frequent* if:

$$sup(GP) \geq \sigma. \qquad (3)$$

5

For example, in Table 2, the gradual item $\{sessions^\uparrow\}$ and the GP $\{age^\uparrow, sessions^\uparrow\}$ are both respected by 3/6 object pairs (i.e. $\{(obj1, obj2), (obj1, obj3), (obj2, obj3)\}$). Therefore, the frequency support is 0.5 in both scenarios.

We adopt the following notation in the rest of the paper. Let $X = \{gp_1, gp_2, ..., gp_n\}$ denote a set of GPs extracted from a data set $\mathcal{D}$. Any GP in X is denoted as *relevant* if and only if $sup(gp_i) \geq \sigma$, where $i \leq n$. The GP in X that has the highest support is denoted as the *best GP*. An *invalid GP* is one that is composed of 2 or more conflicting gradual items (i.e., $\{age^\uparrow, marks^\downarrow, age^\downarrow\}$).

The remainder of the paper is organized as follows: we review related literature in Section 3; we define a lean search space for GP candidates built on top of our numeric encoding in Section 4; we describe a meta-heuristic approach for the problem of mining gradual patterns in Section 5; we present our experimental study in Section 6; finally, we conclude and give future directions concerning the study in Section 7.

3. Review of Literature

Many approaches for efficiently mining GPs have been proposed by numerous researchers. [8] describes an Apriori-based algorithm for extracting gradual dependency from data sets using different classes of fuzzy sets. However, according to [4], this approach is limited to extracting GPs of length 3. Instead of using object pairs to verify GPs, [4] proposes an ordered data set as an alternative.

In reality, none of the approaches proposed by [4] and [8] consider the problem of reducing the number of GP candidates that an algorithm has to verify. In view of Equation 2 and 3, it is easy to confirm that the most computationally intensive task in GP mining involves computing frequency support in order to verify/validate GP candidates by comparing their support values against a user-specified threshold.

In fact, the higher the number of GP candidates to be validated, the more computational resources required by a GP mining algorithm [12, 14, 15, 16]. Meta-heuristic based algorithms may be harnessed to accomplish the task of

efficiently searching for best GP candidates without having to evaluate all the candidates in the search space. In this study, we investigate LS, RS, GA, PSO algorithms for this task.

Random search (also known as pure random search) algorithms are stochastic algorithms that select and evaluate candidates from a search space independently, and the objective function has no effect on the strategy used to pick the next candidate (i.e. each candidate is selected out of pure randomness) [17, 18, 19]. RS is a simple and effective approach to frequent pattern mining problems. Therefore, it is surprising how scarcely it has been applied to the problem of searching for frequent pattern candidates.

Local search (also known as neighborhood search) is a meta-heuristic which tries to find optimal solutions by considering the neighbors of current solutions. In a typical combinatorial optimization problem, a search space is defined by a finite set of feasible candidates, each candidate has a cost, and the goal is to find a candidate with the minimum (or maximum) cost [20, 21, 22, 23]. In the realm of pattern mining, a research study conducted by [24] show that LS algorithms are computationally more efficient than systematic search methods in searching for good quality frequent patterns.

Genetic algorithm is an evolutionary algorithm inspired by natural selection systems. The main operators of GA are selection, crossover and mutation: where, selection operation allows GA to sample the *fittest* candidates from the search space (or population), crossover and mutation operations allow strong GA candidates to mate and produce stronger (or better) candidates [25, 26, 27]. In the realm of frequent pattern mining, research studies conducted by [28] and [29] prove that GA-based approaches surpus their classical counterparts in computational performance.

Particle swarm optimization is a swarm based optimization technique (originally proposed by [30]) that is inspired by the analogy of social interaction and communication (i.e. fish schooling or bird flocking). PSO simulates the movements of swarms in order to iteratively optimize a combinatorial optimization problem [31]. In the realm of frequent pattern mining, research studies con-

7

ducted by [32] and [33] demonstrate how PSO-based approaches improve the performance of the frequent pattern (FP)-growth technique.
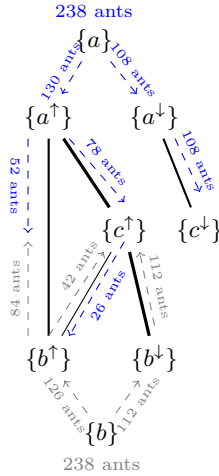

Figure 1: An example of artificial ants for generating GP candidates. Assume that at every iteration there are 2 groups of ants: 238 blue ants arriving at node $\{a\}$ and another 238 gray ants arriving at node $\{b\}$ and, each group of ants try to travel to the opposite node. Recall that every ant in each group is attracted to the path with the highest intensity of pheromone. Initially, the ants select paths haphazardly; but, after a few iterations ants that select shortest paths (illustrated as darker) reach their destinations faster while depositing pheromones. Therefore, ants coming in the opposite direction will find these paths more attractive than the longer paths.

In the realm of GP mining, a research study conducted by [5] proposes and describes an approach that applies an ant colony optimization (ACO) approach to efficiently search for GP candidates whose frequency support values exceed a user-specified threshold value. According to [34, 35, 36, 37], ACO-based approaches imitate the positive feedback reinforcement behavior of ants as they search for food to solve combinatorial optimization problems.

In [5], an ACO variant called *'max-min ant system'* (initially proposed by [37]) is applied to the case GP mining. For instance given attributes $\{a, b, c\}$ of a data set, a search space may be represented such that each point is a path that connects gradual items $\{a^\uparrow, a^\downarrow, b^\uparrow, b^\downarrow, c^\uparrow, c^\downarrow\}$ as shown in Figure 1. Each

8

path represents a GP candidate; therefore, the path that is most travelled by ants becomes the best GP candidate.

Additionally, the study conducted by [5] implement a GA-based approach and a PSO-based approach that efficiently search for best GP candidates whose frequency support exceed a user-specified threshold value. For case of GA and PSO, [5] proposes a bitmap encoding to represent each point of a search space as a combination of bits. For instance given attributes $\{a, b, c\}$ of a data set, a search space may be represented such that each point or candidate is encoded as a (3×2) bitmap. Figure 2 illustrates a bitmap encoding of GP candidate $\{a^\uparrow, b^\downarrow, c^\uparrow\}$.

	$\uparrow$	$\downarrow$
$\{c\}$	1	0
$\{b\}$	0	1
$\{a\}$	1	0

Figure 2: An example of a bitmap encoding for a GP candidate $\{a^\uparrow, b^\downarrow, c^\uparrow\}$. Where $\uparrow$ denotes *'increasing'* and $\downarrow$ denotes *'decreasing'*.

In the final analysis, [5] presents experimental results which show that ACO-based, GA-based and PSO-based techniques computationally out-perform traditional techniques in mining for GPs. Unlike the traditional techniques, these meta-heuristic based techniques do not need to evaluate all possible GP candidates in the search space.

4. Constructing a Numeric Search Space for GP Candidates

In this section, we propose and describe a numeric encoding for GP candidates in Section 4.1; we propose and describe how these encodings may be used to define a search space with lower and upper bounds in Section 4.2; and, we describe an objective function that computes the fitness (or cost) of any selected GP candidate in Section 4.3.

4.1. A Numeric Encoding for GP Candidates

Generally, the task of extracting GPs begins with generating GP candidates. This step is followed by that of computing the support of the candidates (using the formulas described in Section 2) in order to test if they are frequent or not [10]. Depending on the number of attributes in a data set, the possible number of GP candidates may grow exponentially. This growth may easily computationally overwhelm those algorithms that test every possible GP candidate [10, 5, 14].
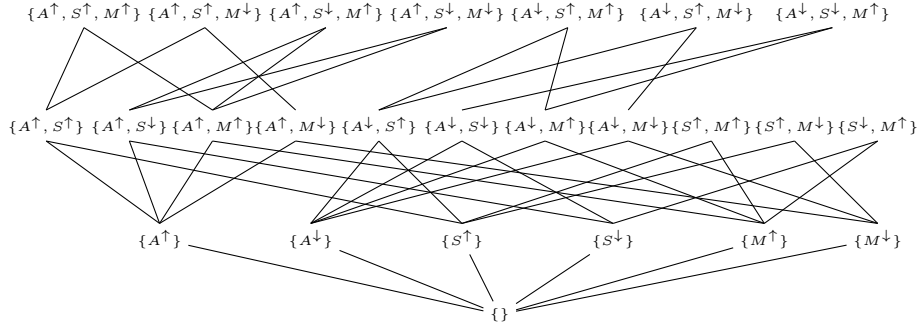
Figure 3: A lattice diagram showing the search space universe for GPs based on the attributes of Table 2. Where A denotes age, S denotes sessions and M denotes marks attributes of the data set in Table 2. Due to space constraints, omit GPs: $\{S^\downarrow, M^\downarrow\}$ and $A^\downarrow, S^\downarrow, M^\downarrow$.

In this paper, we propose and use meta-heuristic approaches (described in Section 5) to efficiently learn frequent GP candidates without testing all individuals in the search universe. In this section, we propose and demonstrate a numeric model that effectively encodes all possible GP candidates (of any given data set) in a search universe within an upper and a lower bound.

To begin with, we mention that a search space for GP candidates is made up of individuals that are formed by arranging a specific set of gradual items in different combinations; and, this set of gradual items is derived from the data set's attribute set. It is important to highlight that the *complementary notion* of GPs permits the existence of 2 gradual items for every attribute [14].

For instance, the data set in Table 2 with the attribute set {age, sessions, marks}, allows for 6 gradual items to exist: $\{age^\uparrow, age^\downarrow, sessions^\uparrow, sessions^\downarrow, marks^\uparrow, marks^\downarrow\}$. And, Figure 3 is a lattice diagram (or search space) of all possible candidates that may be formed from these 6 gradual items.

Different from the lattice diagram representation, we propose a numeric model that may be used to encode each GP candidate in the search universe as a *bit vector (or bit array/string or bit set)* as illustrated in Table 3.

Table 3: Gradual bit vectors of GP candidates in a search universe. Where A denotes age, S denotes sessions and M denotes marks attributes of the data set in Table 2.

Bit vector (b_{gp})	GP Candidate	Bit vector (b_{gp})	GP Candidate
101000	$\{A^\uparrow, S^\uparrow\}$	000110	$\{S^\downarrow, M^\uparrow\}$
100100	$\{A^\uparrow, S^\downarrow\}$	000101	$\{S^\downarrow, M^\downarrow\}$
100010	$\{A^\uparrow, M^\uparrow\}$	101010	$\{A^\uparrow, S^\uparrow, M^\uparrow\}$
100001	$\{A^\uparrow, M^\downarrow\}$	101001	$\{A^\uparrow, S^\uparrow, M^\downarrow\}$
011000	$\{A^\downarrow, S^\uparrow\}$	100110	$\{A^\uparrow, S^\downarrow, M^\uparrow\}$
010100	$\{A^\downarrow, S^\downarrow\}$	100101	$\{A^\uparrow, S^\downarrow, M^\downarrow\}$
010010	$\{A^\downarrow, M^\uparrow\}$	011010	$\{A^\downarrow, S^\uparrow, M^\uparrow\}$
010001	$\{A^\downarrow, M^\downarrow\}$	011001	$\{A^\downarrow, S^\uparrow, M^\downarrow\}$
001010	$\{S^\uparrow, M^\uparrow\}$	010110	$\{A^\downarrow, S^\downarrow, M^\uparrow\}$
001001	$\{S^\uparrow, M^\downarrow\}$	010101	$\{A^\downarrow, S^\downarrow, M^\downarrow\}$
(a)		(b)	

To put it another way, it is possible to compute the length of a bit vector whose individual bits may be toggled in order to provide different encodings for different GP candidates in the search universe, as illustrated in Definition 3.1.

Definition 3.1. (Gradual Bit Vector). *A bit array b_{gp} of size k that encodes a GP candidate; such that $k = (2 \cdot |AT|)$, where AT is the attribute set of a data set.*

For example, we derive a set of 6 gradual items from the attribute set $\{age, sessions, marks\}$ of the data set in Table 2 (i.e. $\{age^{\uparrow}, age^{\downarrow}, sessions^{\uparrow}, sessions^{\downarrow}, marks^{\uparrow}, marks^{\downarrow}\}$). From this, we construct a gradual bit vector of size 6 (i.e 111111) where each binary digit represents the *position* of each gradual item and if the gradual item is *present* or *absent*.

For instance, if items $\{age^{\uparrow}\}$ and $\{marks^{\uparrow}\}$ are the only gradual items present in a GP candidate, then the gradual bit vector may be updated as 100010. Using this numeric model, it is possible to represent all GP candidates in the search universe as gradual bit vectors (as shown in Table 3).

4.2. Defining a Search Space Based on Numeric Encoding

With attention to the numeric encoding (described in Definition 3.1), it is easy to observe that gradual bit vectors may also represent binary numbers (in the base-2 numeral/positional system) within a specific range [38, 39]. A lower bound and an upper bound for this range may easily be defined by the minimum and the maximum bit vectors respectively.

For example, the lower and upper bounds of gradual bit vectors of size 6 may be defined by the values 000000 and 111111 respectively. Notably, (in Table 3) all gradual bit vectors are located within the bounds of these two values.

A part from the binary numeral system, there exist numerous positional numeral systems that may be used in computational arithmetic. The conventional decimal numeral system (which uses 10 digits 0, 1, .., 9) is one of most popular number systems in use among various research domains [39]. We can convert each gradual bit vector into its decimal value and hexadecimal value equivalent.

For example, the bit vectors in Table 3 may be converted into their decimal value equivalents as shown in Table 4. We observe that all positions of the decimal values (that represent the gradual bit vector) lie within the bounds

12

Table 4: Decimal values of bit vector encodings obtained from Table 3.

Bit Vector	Decimal Value	Hexa Value	Bit Vector	Decimal Value	Hexa Value
101000	40	28	000110	06	06
100100	36	24	000101	05	05
100010	34	22	101010	42	2A
100001	33	21	101001	41	29
011000	24	18	100110	38	26
010100	20	14	100101	37	25
010010	18	12	011010	26	1A
010001	17	11	011001	25	19
001010	10	A	010110	22	16
001001	09	09	010101	21	15

(a)   (b)

of 5 and 42 (which are the lower and the upper bound values respectively).
Therefore, we may define a GP search space based on numeric encoding as:
"*an interval $\mathcal{S}_{gp}$ of integer numbers within the bounds of a and b such that, for any $x \in \mathcal{S}_{gp}$, $a \leq x \leq b$*".

Our numeric encoding simplifies the computation of determining the values of bounds a and b respectively. In the case of lower bound a: it is easy to verify that for any data set with at least 2 attributes, the smallest decimal value that corresponds to valid bit vector is 5. For this reason, it is true that $a = 5$ always.

In the case of upper bound b, we propose a different approach for computing its value based on the size of the attribute set of a data set. Using Table 4, we observe that the largest value of a bit vector representing a 2-attribute data set is 1010, and that of a 3-attribute data set is 101010, and that of a 4-attribute data set is 10101010, and so forth. Therefore, b can be computed from the sum of odd binary digits times their power of 2 (2^n) as shown in Equation (4).

$$b = \sum_{i=1}^{|AT|} 2^{2i-1}, \tag{4}$$

where $|AT|$ is the number of attributes. Therefore, we may define $\mathcal{S}_{gp}$ as:

$$\mathcal{S}_{gp} = \left\{ x \in \mathbb{Z} \;\middle|\; 5 \le x \le \sum_{i=1}^{|AT|} 2^{2i-1} \right\}. \tag{5}$$

It is interesting to note that the search space proposed in [5] is based on $[|AT| \times 2]$ binary encoding. Therefore, its dimensions are huge compared to our proposed search space. For example, given 3 attributes, the search space based on binary encoding allows algorithms to traverse almost ($2^6 = 64$) possible candidate combinations, while our proposed search space allows algorithms to traverse only ($42 - 5 = 37$) candidates. The numeric encoding reduces the search space by almost half of that which was proposed in [5].

However, not all gradual bit vectors correspond to valid GP candidates. For example, using the data set in Table 2, 001111 is an invalid GP candidate since it represents conflicting gradual items (i.e., $\{sessions^\uparrow, sessions^\uparrow, marks^\uparrow, marks^\uparrow\}$). In such scenarios, GP algorithms ignore such candidates.

Our proposed numeric search space (although to a lesser amount than the binary search space) still includes some invalid GPs. Using the lattice diagram in Figure 3, we learn that a data set with 3 attributes yields only 20 valid GP candidates. Therefore, our search space contains 17 invalid GP candidates that should be ignored by mining algorithms.

4.3. Computing Candidates' Fitness

Each gradual bit vector should be associated with a fitness/cost value. The fitness value should allow these algorithms to learn (using a probabilistic search) which gradual bit vector encodes the GP candidate with the highest frequency support (described in Section 2).

In this paper, we state that the gradual bit vector with the smallest fitness (or cost) value is the best candidate. Given this point, we observe that there exists a relationship between frequency support of a candidate and fitness. Therefore, a

slight modification of Equation 2 may yield an objective function for determining the fitness of GP candidates. Therefore, the objective function may be defined by the inverse of the concordant object pairs count as shown in Equation 6.

$$fitness(b_{gp}) = \frac{1}{|\{\forall (o, o') \in \mathcal{D}' : A(o) * A(o')\}|}. \tag{6}$$

For example, given the data set in Table 2, we construct some gradual bit vectors and compute their decimal value equivalents and their fitness values as shown in Table 5.

Table 5: A table illustrating the association between bit vectors, GP candidates, decimal values and fitness values respectively. Where A denotes age, S denotes sessions and M denotes marks attributes of the data set in Table 2.

Bit vector	GP Candidate	Decimal value	Fitness value
101000	$\{A^{\uparrow}, S^{\uparrow}\}$	40	0.25
100001	$\{A^{\uparrow}, M^{\downarrow}\}$	33	0.5
011010	$\{A^{\downarrow}, S^{\uparrow}, M^{\uparrow}\}$	26	1.0

Using Table 2, the pattern $\{A^{\uparrow}, S^{\uparrow}\}$ is respected by 4 out of 6 object pairs: $\{(obj1, obj2), (obj1, obj3), (obj1, obj4), (obj2, obj3)\}$. Therefore, the support is 0.67 (see Equation 2) and the fitness value is 0.25 (see Equation 6). As shown in Table 5, the GP $\{A^{\uparrow}, S^{\uparrow}\}$ has the lowest fitness value and highest support in comparison to GPs $\{A^{\uparrow}, M^{\downarrow}\}$ and $\{A^{\downarrow}, S^{\uparrow}, M^{\uparrow}\}$.

5. Metaheuristic Algorithms for GP Extraction

In this section, we propose 4 meta-heuristic algorithms (i.e., LS, RS, GA, PSO) to the problem of finding best GP from the search space proposed in Section 4.3.

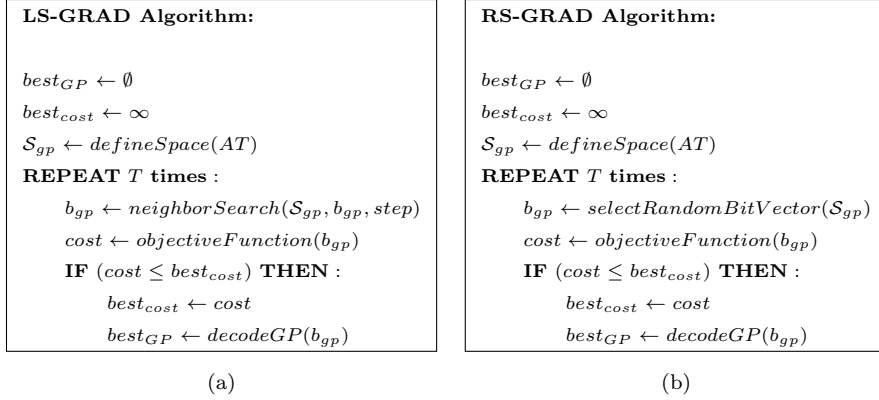### 5.1. LS-GRAD and RS-GRAD Algorithms

We propose RS-GRAD algorithm for GP extraction and it is based on a *random search algorithm* which is a technique whose optimization strategy is based

on a purely stochastic process. In every iteration, random search algorithms are designed to modify the current solution by a random factor [18].

Similarly, we propose LS-GRAD algorithm for GP extraction and it is based on a *stochastic hill climbing algorithm*. Hill climbing technique is an iterative technique that starts from an arbitrary solution and attempts to find a better neighboring solution. A hyper-parameter known as *'step-size'* controls how far the search for a neighboring solution is allowed to go [40].

The main steps of LS-GRAD algorithm and RS-GRAD algorithm are illustrated in Algs. 1(a) and 1(b) respectively.

LS-GRAD Algorithm:	**RS-GRAD Algorithm:**
$best_{GP} \leftarrow \emptyset$	$best_{GP} \leftarrow \emptyset$
$best_{cost} \leftarrow \infty$	$best_{cost} \leftarrow \infty$
$\mathcal{S}_{gp} \leftarrow defineSpace(AT)$	$\mathcal{S}_{gp} \leftarrow defineSpace(AT)$
REPEAT T **times** :	**REPEAT** T **times** :
$\quad b_{gp} \leftarrow neighborSearch(\mathcal{S}_{gp}, b_{gp}, step)$	$\quad b_{gp} \leftarrow selectRandomBitVector(\mathcal{S}_{gp})$
$\quad cost \leftarrow objectiveFunction(b_{gp})$	$\quad cost \leftarrow objectiveFunction(b_{gp})$
$\quad$ **IF** $(cost \leq best_{cost})$ **THEN** :	$\quad$ **IF** $(cost \leq best_{cost})$ **THEN** :
$\quad\quad best_{cost} \leftarrow cost$	$\quad\quad best_{cost} \leftarrow cost$
$\quad\quad best_{GP} \leftarrow decodeGP(b_{gp})$	$\quad\quad best_{GP} \leftarrow decodeGP(b_{gp})$
(a)	(b)

Alg. 1: (a) LS-GRAD algorithm, (b) RS-GRAD algorithm.

As shown in Alg. 1(a), LS-GRAD algorithm defines a search space $\mathcal{S}_{gp}$ such that each point is encoded as numeric digit that represents a GP candidate and it iterates T times. In each iteration, the algorithm:

- finds a neighbor of the current selected bit vector candidate b_{gp} in search space $\mathcal{S}_{gp}$ and this neighbor becomes the selected candidate (using *neighbor Search()* function);

- evaluates fitness of the selected candidate (using *objectiveFunction()*).

As shown in Alg. 1(b), RS-GRAD algorithm defines a search space $\mathcal{S}_{gp}$ such that each point is encoded as numeric digit that represents a GP candidate and

it iterates T times. In each iteration, the algorithm:

- randomly selects a bit vector candidate from search space S_{gp} (using $selectRandomBitVector()$ function);

- evaluates fitness of the selected candidate (using $objectiveFunction()$).

Computational Complexity of LS-GRAD and RS-GRAD

We use big-O notation [41, 42] to analyze the computational complexity of LS-GRAD and RS-GRAD algorithms. For every candidate that is selected, both LS-GRAD and RS-GRAD algorithms compute fitness using the *objective Function*() (or fitness function described in Section 4.3). We recall that the formula proposed for computing the fitness function (see Equation 6) is a variant of the formula used for computing frequency support (see Equation 2). Therefore, the computational complexities of implementing these two functions are fairly similar. According to [5], given a data set with a attributes and n objects, the task of computing the fitness of a GP candidate composed of k gradual items (where $k \leq a$) has a complexity of $O(k \cdot n^2)$.

Another key point is that in both LS-GRAD and RS-GRAD algorithms, the tasks of implementing functions like $defineSpace()$, $decodeGP()$, $neighbor$ $Search()$ and $selectRandomBitVector()$ have small and almost constant computational complexities relative to the task of the *objectiveFunction*(). Therefore, it is acceptable to simplify the big-O analysis to focusing on computationally intensive tasks (i.e. *objectiveFunction*()). Given these points, if both LS-GRAD and RS-GRAD algorithms iterate T times, their computational complexities are relatively of the same equivalence - that is $O(T \cdot k \cdot n^2)$.

5.2. GA-GRAD and PSO-GRAD Algorithms

We propose GA-GRAD algorithm for GP extraction and it is built on top of GA which is an optimization technique that is inspired by Darwin's theory about evolution [6]. GA techniques are initiated with a population of solutions (also called *chromosomes*). Existing solutions are selected according to their fitness and they mate to reproduce offsprings whose fitnesses are better.

Through this reproduction operation GA techniques efficiently find best solutions to combinatorial optimization problems. Specifically, GA uses the following operators in order to improve the fitness of the offsprings:
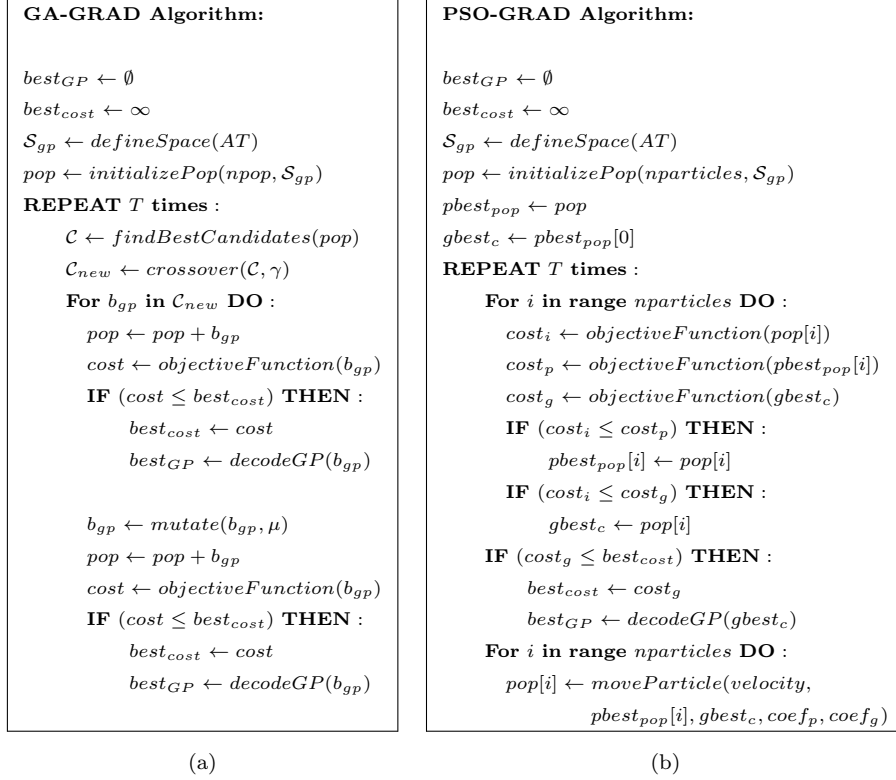
- Crossover: passing of good genes to next generation. γ denotes the parameter that controls crossover rate.

- Mutation: random changes in genes during crossover. μ denotes the parameter that controls mutation rate.

Similarly we propose PSO-GRAD algorithm for GP extraction and it is built on top of PSO which is an optimization technique that is inspired by collective behavior and movements of swarms (especially as they search for food). PSO techniques are initialized with a population of particles and in every iteration each particle moves to a new position whose cost/fitness is better than its current position [32].

A PSO algorithm performs a constant search for best position in a search space. Particles are generated and they are randomly assigned positions from a uniform distribution with the search space; the algorithm moves particles with a certain calculated velocity in every iteration. Personal best position (p_i and global best position ($gbest$) are the best position visited so-far by the i-th particle and the best position visited so far by any particle in the swarm [43]. PSO uses the following operators:

- Current direction/velocity: determines the movement of each particle.

- Personal best position (p_i) and cognitive component ($coef_p$): $coef_p$ co-efficient gives the importance of personal best value.

- Team/global best position ($gbest$) and social component ($coef_g$): $coef_g$ co-efficient gives the importance of global best value.

The main steps of GA-GRAD algorithm and PSO-GRAD algorithm are illustrated in Algs. 2(a) and 2(b) respectively.

<table>
<tr><td>

GA-GRAD Algorithm:

$best_{GP} \leftarrow \emptyset$
$best_{cost} \leftarrow \infty$
$\mathcal{S}_{gp} \leftarrow defineSpace(AT)$
$pop \leftarrow initializePop(npop, \mathcal{S}_{gp})$
REPEAT T **times** :
 $\mathcal{C} \leftarrow findBestCandidates(pop)$
 $\mathcal{C}_{new} \leftarrow crossover(\mathcal{C}, \gamma)$
 For b_{gp} **in** $\mathcal{C}_{new}$ **DO** :
  $pop \leftarrow pop + b_{gp}$
  $cost \leftarrow objectiveFunction(b_{gp})$
  IF $(cost \leq best_{cost})$ **THEN** :
   $best_{cost} \leftarrow cost$
   $best_{GP} \leftarrow decodeGP(b_{gp})$

  $b_{gp} \leftarrow mutate(b_{gp}, \mu)$
  $pop \leftarrow pop + b_{gp}$
  $cost \leftarrow objectiveFunction(b_{gp})$
  IF $(cost \leq best_{cost})$ **THEN** :
   $best_{cost} \leftarrow cost$
   $best_{GP} \leftarrow decodeGP(b_{gp})$

</td><td>

PSO-GRAD Algorithm:

$best_{GP} \leftarrow \emptyset$
$best_{cost} \leftarrow \infty$
$\mathcal{S}_{gp} \leftarrow defineSpace(AT)$
$pop \leftarrow initializePop(nparticles, \mathcal{S}_{gp})$
$pbest_{pop} \leftarrow pop$
$gbest_c \leftarrow pbest_{pop}[0]$
REPEAT T **times** :
 For i **in range** $nparticles$ **DO** :
  $cost_i \leftarrow objectiveFunction(pop[i])$
  $cost_p \leftarrow objectiveFunction(pbest_{pop}[i])$
  $cost_g \leftarrow objectiveFunction(gbest_c)$
  IF $(cost_i \leq cost_p)$ **THEN** :
   $pbest_{pop}[i] \leftarrow pop[i]$
  IF $(cost_i \leq cost_g)$ **THEN** :
   $gbest_c \leftarrow pop[i]$
  IF $(cost_g \leq best_{cost})$ **THEN** :
   $best_{cost} \leftarrow cost_g$
   $best_{GP} \leftarrow decodeGP(gbest_c)$
 For i **in range** $nparticles$ **DO** :
  $pop[i] \leftarrow moveParticle(velocity,$
   $pbest_{pop}[i], gbest_c, coef_p, coef_g)$

</td></tr>
<tr><td align="center">(a)</td><td align="center">(b)</td></tr>
</table>

Alg. 2: (a) GA-GRAD algorithm, (b) PSO-GRAD algorithm.

As shown in Alg. 2(a), GA-GRAD algorithm is initialized to a population pop having $npop$ candidates and each candidate is a gradual bit vector b_{gp} within search space $\mathcal{S}_{gp}$. The algorithm iterates T times and in each iteration it:

- selects 2 of the best candidates from population pop (using $findBest$ $Candidates()$ function);

- performs a crossover function using these 2 candidates in order to reproduce 2 offsprings (using $crossover()$ function at a rate of γ);

- evaluates the fitness of the offsprings (using $objectiveFunction()$);

- mutates each offspring's bit vector (using $mutate()$ function at a rate of μ) and evaluates their fitness;

- adds the offsprings to population *pop*.

As shown in Alg. 2(b), PSO-GRAD algorithm is initialized to a population *pop* having n particles and each particle's position represents a gradual bit vector within search space $\mathcal{S}_{gp}$. In addition, a global best particle $gbest_c$ is initialized. The algorithm iterates T times and in each iteration, it:

- evaluates the fitness of all particles' positions in the population and the fitness of the global best particle's position (using *objectiveFunction*());

- for each particle's position if the fitness value is lower than the fitness value of its personal best position in history, set the personal best position to this position;

- replace the global best particle position with any particle position whose fitness value is lower than the laters;

- moves each particle by a velocity *vel* towards its personal best at a rate of $coef_p$ and towards the global best particle at a rate of $coef_g$.

Computational Complexity of GA-GRAD and PSO-GRAD

We use big-O notation [41, 42] to analyze the computational complexity of GA-GRAD and PSO-GRAD algorithms. Similar to LS-GRAD and RS-GRAD algorithms, we take the computational complexity of a single *objectiveFunction*() (implemented by GA-GRAD and PSO-GRAD algorithms) as equivalent to $O(k \cdot n^2)$. Again, comparable to GA-GRAD and PSO-GRAD algorithms, we ignore functions whose computational complexities are small and almost constant relative to the *objectiveFunction*().

GA-GRAD algorithm is initialized to a population of *npop* candidates where only 2 candidates are selected in every iteration. Again, the algorithm evaluates the fitness of 2 crossed offsprings and 2 mutated offsprings in every iteration. Therefore, in 1 iteration the computational complexity of GA-GRAD algorithm is equivalent to $O(4 \cdot k \cdot n^2)$ and in T iterations it is equivalent to $O(4 \cdot T \cdot k \cdot n^2)$

20

PSO-GRAD algorithm is initialized to a population of n particles. In each iteration, the algorithm evaluates the fitness of all these particles together with the fitness of their historical personal best and global best particle. Therefore, in 1 iteration the computational complexity of PSO-GRAD algorithm is equivalent to $O\{(2n+1)\cdot k\cdot n^2\}$ and in T iterations it is equivalent to $O\{(2n+1)\cdot T\cdot k\cdot n^2\}$.

6. Experiments

In this section, we present an experimental study of the computational performance of our algorithms. Specifically, we analyze the performance and behavior of the algorithms when exploring search spaces based on either a bitmap encoding (initially proposed in [5]) or a numeric encoding (proposed in Section 4). The aim of this experiment is to establish which search space is simpler for meta-heuristic algorithms.

In addition to this, we compare the computational performance of the meta-heuristic algorithms in mining GPs. All experiments were conducted on a HPC (High Performance Computing) platform **Meso@LR**[1]. We used one node made up of 14 cores and 128GB of RAM.

6.1. Source Code

The Python source code of our algorithms is available at the GitHub repository `https://github.com/owuordickson/swarm_gp.git`. The Python package for installing these algorithms is available at the PyPi repository through this link: `https://pypi.org/project/so4gp/`.

6.2. Data Set Description

Table 6 briefly describes the features of the data sets used in our experiments for evaluating the performance of our algorithms. All these data sets are numeric data set. We performed a data cleaning task (on all the data sets) that involved omitting objects with one or more *missing values* and omitting attributes whose

---

[1]`https://meso-lr.umontpellier.fr`

21

objects are either *non-numeric values* or *time-stamp values*. All these data sets were retrieved from the UCI Machine Learning Repository[2][44].

Table 6: Experiment data sets.

Data set	#objects used	#attributes	Domain	Source
Breast Cancer (BCR)	116	10	Medical	[45]
Hepatitis C (HCV)	615	14	Medical	[46]
Chickenpox (CPX)	521	20	Zoonosis	[47]
Cargo 2000 (C2K)	3942	98	Transport	[48]
Air Quality (AQY)	9358	15	Environment	[49]
APS Data (APS)	2474	171	Vehicle	[50]
Omnidir (OMD)	2000	11	Coastline	[51]
Directio (DIR)	8075	21	Coastline	[51]
Power Consumption (HPC)	10001	9	Energy	[44]

The HCV data set is a data set that contains laboratory values of blood donors of healthy individuals and individuals infected with Hepatitis C or Fibrosis or Cirrhosis together with demographic values (i.e. age, sex) [46]. The CPX data set is a spatio-temporal data set containing county-level records of weekly chickenpox cases in Hungary between the years of 2005 and 2015 [47].

The C2K data set is a data set that holds records of Cargo 2000 airfreight tracking and events for a period of 5 months. The C2K data set has 98 attributes and 3942 objects (some with missing values) [48]. The AQY data set is a data set that contains records of a gas multi-sensor device (i.e. hourly averages of gas concentrations) deployed on a field in an Italian city [49].

The BCR data set is composed of 10 predictors and a binary variable indicating the presence or absence of breast cancer. The predictors are recordings of anthropometric data gathered from the routine blood analysis of 116 participants [45]. The HPC data set describes the electric power consumption in one household (located in Sceaux, France) between 2006 and 2010 [44].

The APS data set holds data of the *Air Pressure System* (APS) collected from heavy Scania trucks in their daily usage [50]. The DIR and OMD data sets

---

[2]https://archive.ics.uci.edu/ml/index.php

are obtained from OREMES's data portal[3] that recorded swell sensor signals of 4 buoys near the coast of the Languedoc-Roussillon region in France between 2012 and 2019 [51].

6.3. Parameter Tuning

In order to obtain the highest efficiency from GA-GRAD, PSO-GRAD and LS-GRAD algorithms, careful tuning of their respective parameters is required. We utilize a Bayesian optimization approach proposed by [52] to achieve this task. Firstly, we determine the appropriate maximum value of iterations T (see Section 5) for all the algorithms when applied to all the data sets described in Section 6.2. As can be seen in Figure 4, we observe that none of the algorithms exceeds 10 iterations when applied on any data set. Therefore, we set the maximum value of iterations to 20 in order to how smoothly each algorithm's plot settles in Section 6.4.
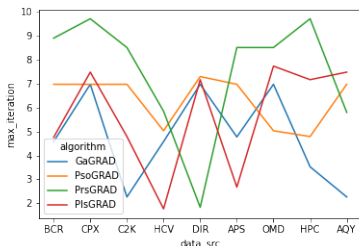
Figure 4: Determining the appropriate maximum value of iterations (T). Where GaGRAD denotes GA-GRAD, PsoGRAD denotes PSO-GRAD, PlsGRAD denotes LS-GRAD, PrsGRAD denotes RS-GRAD.

Secondly, for each algorithm we determine the optimum values of their respective parameters when applied on different data sets by using the automated Bayesian optimization approach [52]. Figures 5, 6, 7 show a plot of specific parameters against different data sets for GA-GRAD, PSO-GRAD and LS-GRAD algorithms respectively. From results displayed in these figures, it is can be de-

---

[3]https://data.oreme.org

23

duced that each algorithm needs a different combination of parameter values in order to perform optimally for different data sets.
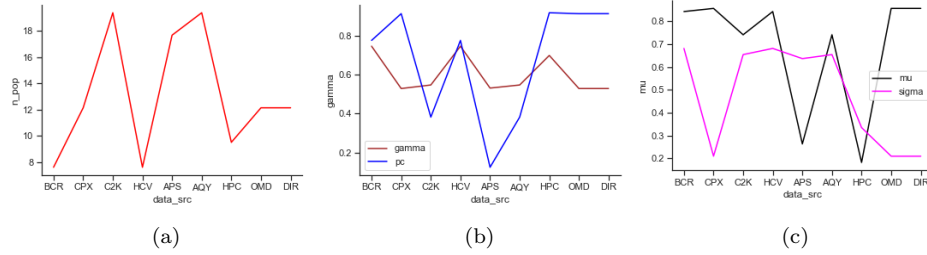
Figure 5: GA-GRAD parameters: (a) initial parent population, (b) offspring proportion (pc) and crossover γ parameter, (c) mutation μ and σ parameters.
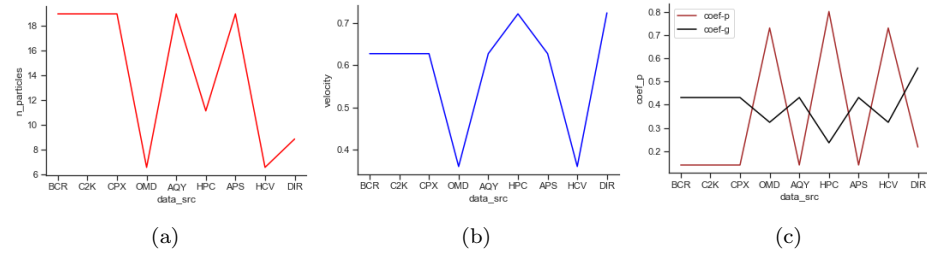

Figure 6: PSO-GRAD parameters: (a) initial particle population, (b) velocity parameter, (c) personal and global coefficients.
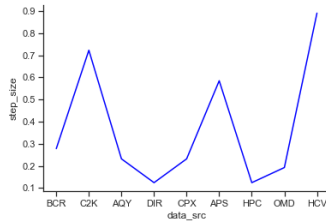

Figure 7: LS-GRAD's step-size parameter.

For this reason, in order to reach maximum the efficiency of each our algorithms in our experimental analysis presented in Section 6.4, we tune their

respective parameters to different values according to the data set involved.

6.4. Experiment Results

In this section, we present our experiment results which reveal the computational behavior of LS-GRAD, RS-GRAD, GA-GRAD, PSO-GRAD algorithms when applied to the data sets described in Section 6.2. All experiments involve performing at least 3 repeated test runs on each algorithm and the results are a mean of the repeated test runs of each algorithm on each data set. A single test run allows each algorithm to execute 20 iterations ($T = 20$) in each search space. In every iteration, all the algorithms try to find the best GP candidate among those provided by each search space.

In Section 6.4.1, we perform a *Wilcoxon test* to determine which search space better suites our algorithms. We conduct 2 experiments on the 4 algorithms: one using bitmap encoding to construct search spaces in and, another using numeric encoding to construct search spaces.

In Section 6.4.2, we compare the distribution of *valid* GPs and *invalid* GPs extracted from various data sets by our algorithms using the numeric-based search space against the bitmap-based search space. In Section 6.4.3, we conduct an experiment that compares the performance of our algorithms using the numeric-based search space against other existing GP mining algorithms

6.4.1. Experiment 1: Wilcoxon Test on Search Spaces

The Wilcoxon test is a non-parametric measure that compares the differences between at most 2 paired groups of data [53]. In this study, we evaluate the effect of different search spaces (i.e., numeric-based and bitmap-based) on our algorithms' run-time durations in different data sets. The dependent variable is the run-time duration measured on a continuous scale. The Wilcoxon test uses the following null and alternative hypotheses:

- Null hypothesis (H_0): the mean run-time for each group is equal.

- Alternative hypothesis: (H_a): one group's mean run-time is different from the other.

Table 7: Summary of run-times of algorithms on different data sets. Where Ga denotes GA-GRAD Pso denotes PSO-GRAD, Prs denotes RS-GRAD, Pls denotes LS-GRAD algorithms respectively and NU denotes the numeric-based search space, BM denotes the bitmap-based search space. For instance Ga-NU implies GA-GRAD using the numeric-based search space.

Data	Algorithms' mean run-times							
	Ga-NU	Ga-BM	Pso-NU	Pso-BM	Pls-NU	Pls-BM	Prs-NU	Prs-BM
APS	243.12	543.65	335.65	1711.50	133.88	253.88	53.17	179.40
AQY	346.30	653.78	439.62	487.62	60.69	98.26	59.20	51.09
C2K	165.80	422.08	136.93	416.10	23.18	60.14	22.67	43.79
DIR	1475.00	3516.75	828.90	1193.25	226.18	376.28	222.38	187.98
HPC	495.55	1632.75	489.70	628.88	114.78	96.68	92.06	77.50
OMD	293.98	767.12	123.00	135.65	38.08	52.42	35.28	29.86
HCV	0.77	1.05	0.47	0.48	0.22	0.21	0.19	0.15
CPX	1.73	2.34	1.61	1.58	0.33	0.31	0.28	0.22
BCR	0.36	0.39	0.52	0.48	0.08	0.09	0.08	0.07

Table 7 shows the mean run-times for each algorithm on the numeric-based and bitmap-based search space. Using these values, the results of the Wilcoxon test are shown in Table 8.

Table 8: Wilcoxon test on efficacy of search space

Algorithm	Results
GA-GRAD	test statistic = 0, p-value = 0.0039
PSO-GRAD	test statistic = 5, p-value = 0.0391
LS-GRAD	test statistic = 10, p-value = 0.1641
RS-GRAD	test statistic = 16, p-value = 0.4961

6.4.2. Experiment 2: Comparing Candidate GPs from Search Spaces

Here, we seek to discover which search space allows our algorithms to yield a better distribution in favor of *valid* GPs. Every candidate that each algorithm generates, is evaluated to check if it is valid or invalid. Figures 8, 9, 10, 11, 12, 13, 14, 15 are scatter plots of position against fitness/validity of candidates, where valid GPs are labelled as *'True'* and invalid GPs are labelled as *'False'*.

In Figures 8 and 9, we observe that the distribution of GP candidates is almost uniformly spread across the entire numeric-based search space; while,

it is sparsely spread across the bitmap-based search space. APS and C2K data sets have large number of attributes in comparison to the other data sets; this explains why the distribution of candidates diagonally cuts across the numeric search space. GA-GRAD algorithm does not find any valid GP using the bitmap-based search space from APS and C2K data sets.
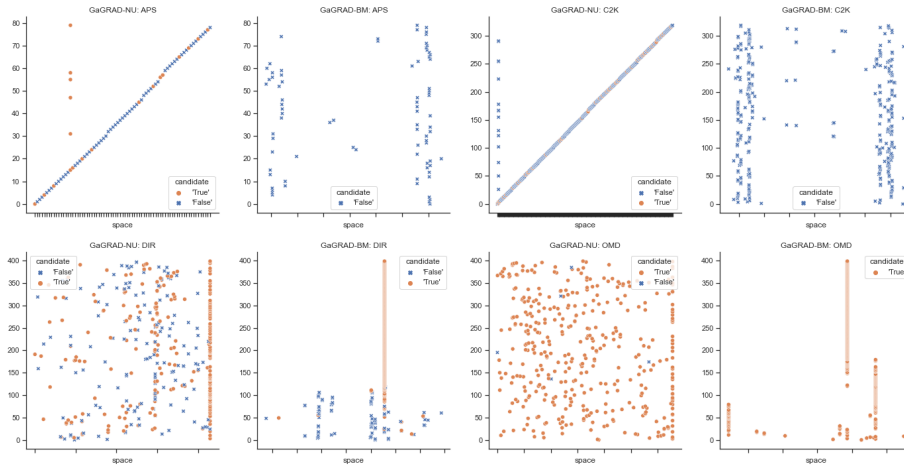

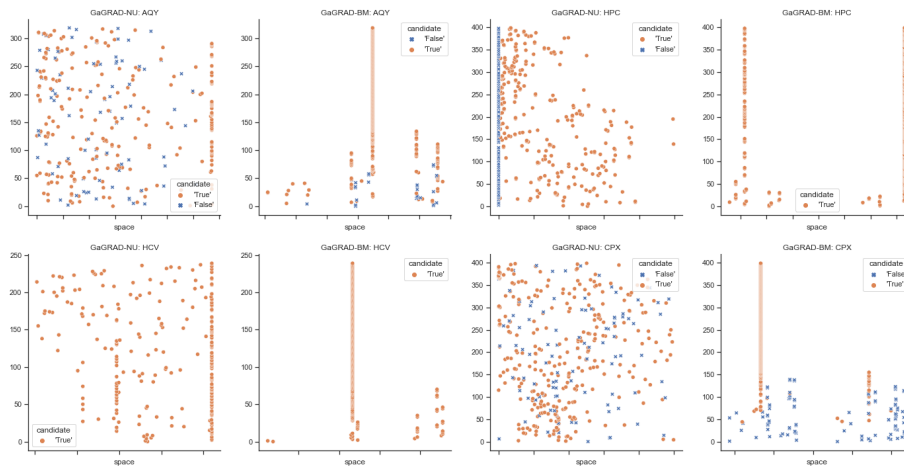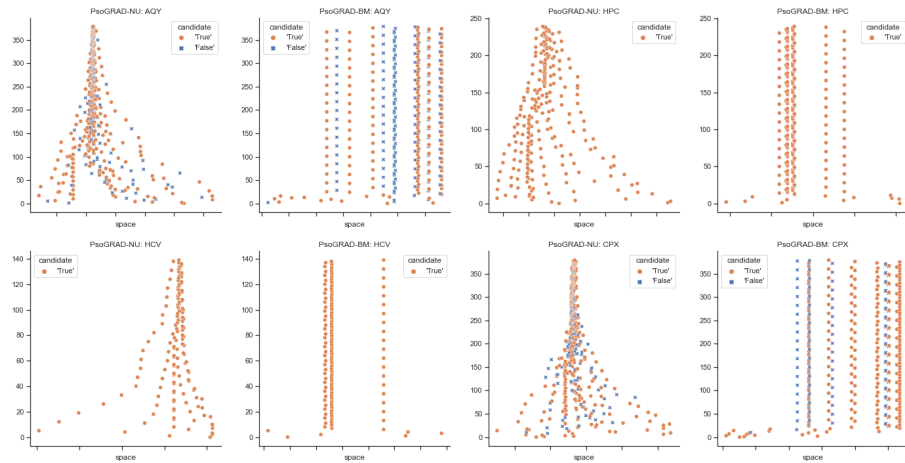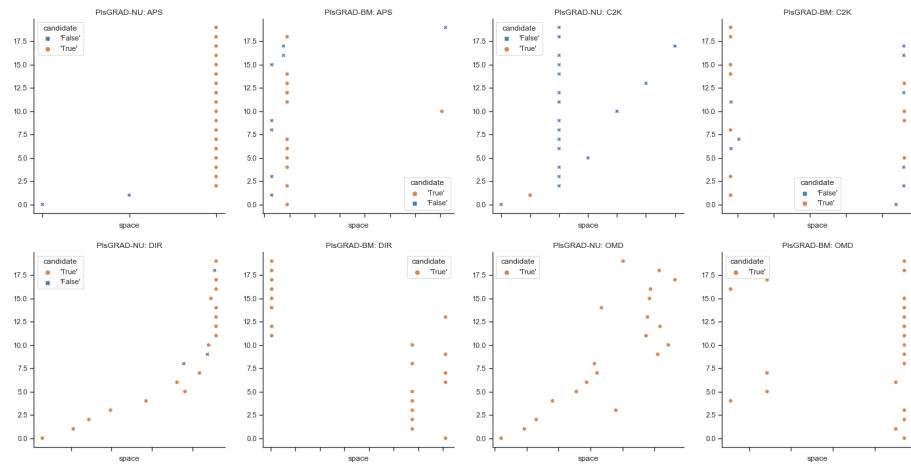Figure 8: GA-GRAD's candidates from APS, C2K, DIR and OMD data sets using numeric-based and bitmap-based search spaces.

Figure 9: GA-GRAD's candidates from AQY, HPC, HCV and CPX data sets.

Similarly in the case of PSO-GRAD algorithm, Figure 10 shows that PSO-GRAD algorithm does not find any valid GP using the bitmap-based search space from APS and C2K data sets.
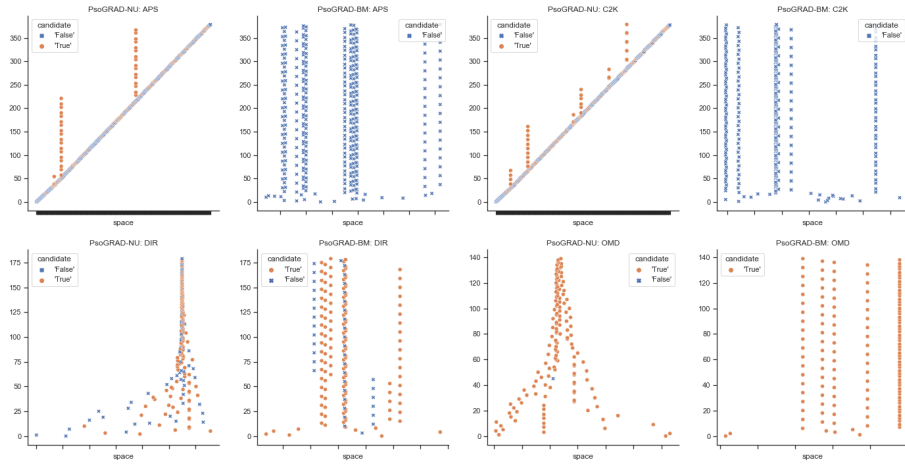
Figure 10: PSO-GRAD's candidates from APS, C2K, DIR and OMD data sets using numeric-based and bitmap-based search spaces.

Figure 11: PSO-GRAD's candidates from AQY, HPC, HCV and CPX data sets using numeric-based and bitmap-based search spaces.

Again in Figures 10 and 11, we observe that the distribution of candidates are directed towards a single point in the case of the numeric-based search space.

It is interesting to observe in Figures 12 and 13, that there is no significant difference between the distributions of GP candidates in both the numeric-based and the bitmap-based search spaces. This phenomenon confirms the results of the Wilcoxon test presented in Section 6.4.1 concerning the LS-GRAD algorithm.

Figure 12: LS-GRAD's candidates from APS, C2K, DIR and OMD data sets using numeric-based and bitmap-based search spaces.
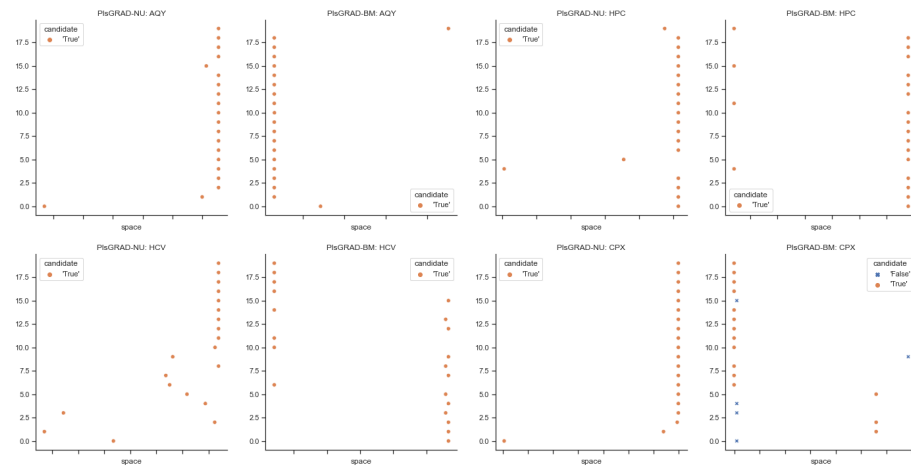

Figure 13: LS-GRAD's candidates from AQY, HPC, HCV and CPX data sets using numeric-based and bitmap-based search spaces.

29

In the case of RS-GRAD algorithm, we observe in Figures 14 and 15, that there is no significant difference between the distributions of GP candidates in both the numeric-based and the bitmap-based search spaces. Again this observation confirms the results of the Wilcoxon test presented in Section 6.4.1.
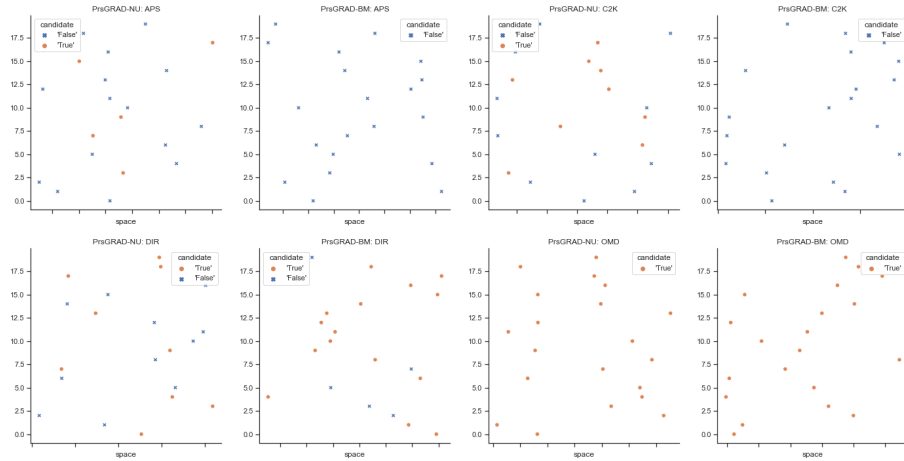

Figure 14: RS-GRAD's candidates from APS, C2K, DIR and OMD data sets using numeric-based and bitmap-based search spaces.
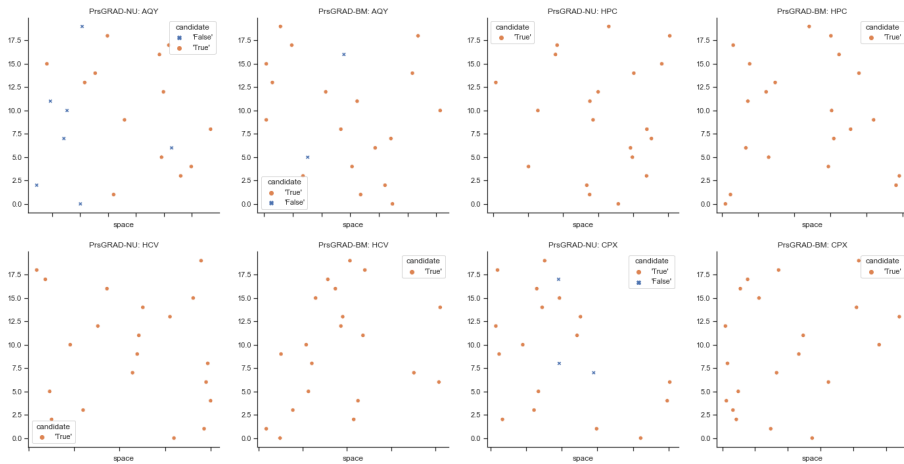

Figure 15: RS-GRAD's candidates from AQY, HPC, HCV and CPX data sets using numeric-based and bitmap-based search spaces.

6.4.3. Experiment 3: Comparative Analysis

In this experiment, we compare the performance of our algorithms in terms of run-time duration, memory consumption and number of extracted GPs against: GRAANK initially proposed by [9] and ACO-GRAANK proposed by [5].

We clarify the following: (1) There exists other GP mining algorithms that rely on a tree-based search for candidates; such as, GRITE proposed by [54] and ParaMiner proposed by [15]. The gradualness semantic of such algorithms (which is focused on an ordered precedence graph of objects) is different from those under investigation in this study (which extend GRAANK), in that they rely on a breath-first search and use a gradualness semantic focused on concordant object couples. The proposed search spaces are best suited for GRAANK-based algorithms; therefore, it is almost unfair to compare their computational performance to GRITE-based algorithms. (2) We use AcoGRAD to denote ACO-GRAANK. (3) GRAANK algorithm yields *'Memory Error'* when executed on APS and DIR data sets.
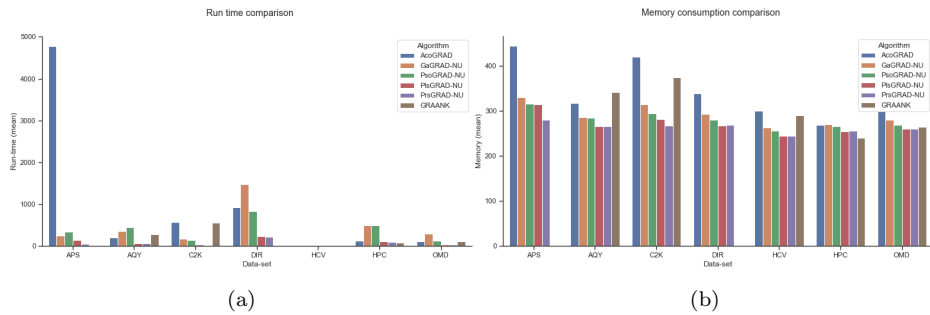

Figure 16: Comparison of algorithms' run-times and memory consumptions.

In Figure 16 (a), we observe that AcoGRAD has the longest run-time when applied on data sets with large number of attributes (i.e., APS and C2K). However, AcoGRAD and GRAANK algorithms have the shorter run-times than PSO-GRAD and GA-GRAD algorithms when applied on data sets with few attributes (i.e., HPC, OMD, AQY). LS-GRAD and RS-GRAD algorithms have the shortest run-times across all the data sets. In terms of memory consumption

31

(as shown in Figure 16 (b)), with the exception of AcoGRAD and GRAANK, all the other algorithms consume almost the same amount of memory. However, GRAANK's memory consumption grows exponentially when applied on APS and DIR data sets; thus, yielding a *'Memory Error'*.

In Figure 17 (a), we observe that GRAANK extracts the most number of valid GPs in C2K data set. However, in all the other data sets GA-GRAD and PSO-GRAD algorithms extract the most number of valid GPs. Recall that GRAANK algorithm yields *'Memory Error'* when applied on APS and DIR data set; for this reason, the number of valid and invalid GPs could not be recorded.
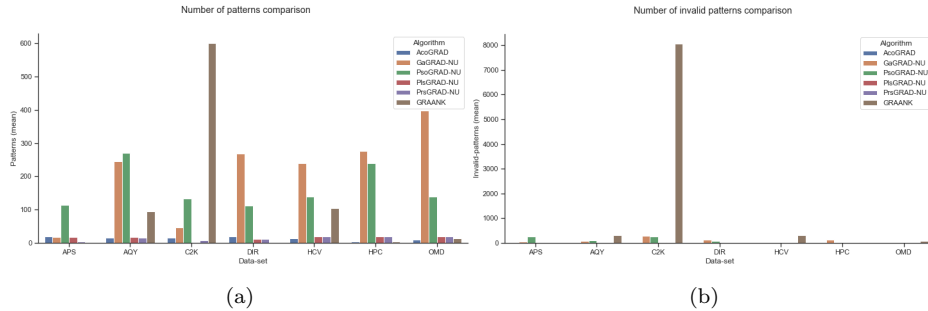

Figure 17: Comparison of number of GPs extracted by each algorithm.

In Figure 17 (b), we observe that GRAANK algorithm generates the most number of invalid GPs especially in C2K, AQY and HCV data sets. It is interesting to observe that our meta-heuristic algorithms generally generate very few invalid GPs. However, we also observe that GA-GRAD and PSO-GRAD algorithms generate slightly more number of invalid GPs than RS-GRAD and LS-GRAD algorithms.

6.5. Discussion of Results

6.5.1. Search Space

The Wilcoxon test results presented in Table 8 show low p-value (less than 0.05) for GA-GRAD and PSO-GRAD algorithms. This implies that there is sig-

nificant differences in the run-time durations when these 2 algorithms are using the bitmap-based search space versus when they are using the numeric-based search space. From Table 7, we observe that al 4 algorithms have faster run-times in most data sets when using the numeric-based search space. This makes the numeric-based search space more effective than its counterpart especially in the case of PSO-GRAD and GA-GRAD algorithms.

Again in Section 6.4.2, we observe that in the scatter plots of GA-GRAD and PSO-GRAD algorithms, GP candidates are more uniformly spread in the numeric-based search space than in the bitmap-based search space. This implies that the numeric-based search space allows these algorithms to explore more areas of the search space. Furthermore, these 2 algorithms do not find any valid GP from data sets with large numbers of attributes using the bitmap-based search space (i.e., APS and C2K). This may confirm the inefficacy of the bitmap-based search space especially when used on data sets with large numbers of data sets.

However, LS-GRAD and RS-GRAD algorithms seem to have almost similar distribution of GP candidates and little differences in run-time durations in both search spaces. In order to break the tie in favor of the numeric-based search spaces, we focus on the APS and C2K data sets. We observe in Figures 12 and 14 that both algorithms do not find any valid GP in the APS data set. Although LS-GRAD algorithm finds valid GPs from C2K data set using both search spaces, it does so at the expense of a longer run-time in the bitmap-based search space (see Table 7). In fact, the run-time is almost 3 times as long in comparison to the numeric-based search space.

6.5.2. Execution Run-time

We observe that generally PSO-GRAD and GA-GRAD algorithms have longer run times than LS-GRAD and RS-GRAD algorithms. Specifically, PSO-GRAD followed by GA-GRAD algorithms have the longest run times in all the search spaces. This may be due to the fact that for every search iteration, PSO-GRAD and GA-GRAD algorithms evaluate more than one candidate GP.

Recall (from Section 5.2 and Section 5.1) that the task of evaluating GP candidates is the most computationally intensive task. Therefore, the more GP candidates that are evaluated by the *objective function* in every search iteration, the higher the computational complexity of the algorithm.

6.5.3. Extracted Patterns

We observe that generally GA-GRAD, PSO-GRAD and RS-GRAD algorithms extract more GPs than LS-GRAD algorithm. This phenomenon may be attributed to the fact PSO-GRAD and GA-GRAD algorithms generally evaluate higher number of candidates per every iteration in comparison to LS-GRAD and RS-GRAD algorithm, which in turn improves their chances of finding more valid GPs. However, we mention that our proposed metaheuristic algorithms are not guaranteed to find all frequent GPs in each data set. In reality, their searches only focus and converge to candidates in the search space that extract the GPs with the highest support (also referred to as best GPs), which is a subset of the frequent GPs.

In contrast, the classical GRAANK (initially proposed by [9]) is a complete GP mining algorithm; in the sense that, it tests all possible candidates in the search space and extracts all the frequent GPs. In fact, our experiments reveal that GRAANK algorithm extracts the most number of valid GPs (especially in C2K data set); however, this comes with the downside of a higher run-time, an exponential growth of memory consumption (especially for APS and DIR data sets), and a generation of the huge numbers of invalid GPs.

Given these points, we recommend GRAANK algorithm as the suitable algorithm for extracting GPs from small data sets and, we recommend our proposed metaheuristic algorithms (i.e., GA-GRAD, PSO-GRAD, LS-GRAD and RS-GRAD) as the suitable algorithms for extracting GPs from large data sets (especially those with huge numbers of attributes).

7. Conclusion and Future Works

In this research study, we examine the problem of finding valid gradual patterns from a huge search space without having to test every candidate. We identify stochastic, evolutionary and swarm based optimization techniques as one of the efficient techniques for solving this problem. We define a lean search space based on numeric encoding that can easily be explored by these algorithms to find valid GPs.

In our experimental study, we compare the performances of LS-GRAD, RS-GRAD algorithms, GA-GRAD and PSO-GRAD algorithms in exploring gradual pattern search spaces based on bitmap encoding verses those based on numeric encoding. The experiment results reveal that all the algorithms find best gradual patterns at shorter run-times from search spaces built on numeric encoding. This may imply that the numeric encoding builds search spaces that are leaner and easier to explore by meta-heuristic algorithms than the bitmap encoding.

In future, an even more effective encoding for gradual pattern candidates may be envisioned that allows gradual patterns to be extracted using machine learning techniques. It may also be interesting to explore the possibility of further reducing the search space of candidates so that algorithms generate even fewer invalid gradual patterns.

References

[1] M. N. Ab Wahab, S. Nefti-Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms, PLOS ONE 10 (5) (2015) 1–36. `doi:10.1371/journal.pone.0122827`.

[2] M. Abdel-Basset, L. Abdel-Fatah, A. K. Sangaiah, Chapter 10 - metaheuristic algorithms: A comprehensive review, in: A. K. Sangaiah, M. Sheng, Z. Zhang (Eds.), Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Intelligent Data-Centric Systems, Academic Press, 2018, pp. 185–231. `doi:https://doi.org/10.1016/B978-0-12-813314-9.00010-4`.

[3] C. Fan, Y. Sun, K. Shan, F. Xiao, J. Wang, Discovering gradual patterns in building operations for improving building energy efficiency, Applied Energy 224 (2018) 116–123. `doi:https://doi.org/10.1016/j.apenergy.2018.04.118`.

[4] L. Di Jorio, A. Laurent, 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, CSTST '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 205–210. `doi:10.1145/1456223.1456268`.

[5] D. O. Owuor, T. Runkler, A. Laurent, J. O. Orero, E. O. Menya, Ant colony optimization for mining gradual patterns, International Journal of Machine Learning and Cybernetics (2021) 1–21`doi:10.1007/s13042-021-01390-w`.

[6] M. Obitko, P. Slavík, Visualization of genetic algorithms in a learning environment, in: Spring Conference on Computer Graphics, SCCG, Vol. 99, 1999, pp. 101–106.

[7] V. P. Patel, M. K. Rawat, A. S. Patel, Analysis of search space in the domain of swarm intelligence, in: M. Prateek, T. P. Singh, T. Choudhury,

H. M. Pandey, N. Gia Nhu (Eds.), Proceedings of International Conference on Machine Intelligence and Data Science Applications, Springer Singapore, Singapore, 2021, pp. 99–109.

[8] F. Berzal, J. C. Cubero, D. Sanchez, M. A. Vila, J. M. Serrano, An alternative approach to discover gradual dependencies, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15 (05) (2007) 559–570. `doi:10.1142/S021848850700487X`.

[9] A. Laurent, M.-J. Lesot, M. Rifqi, Graank: Exploiting rank correlations for extracting gradual itemsets, in: Proceedings of the 8th International Conference on Flexible Query Answering Systems, FQAS '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 382–393. `doi:10.1007/978-3-642-04957-6_33`.

[10] D. O. Owuor, A. Laurent, Efficiently mining large gradual patterns using chunked storage layout, in: L. Bellatreche, M. Dumas, P. Karras, R. Matulevičius (Eds.), Advances in Databases and Information Systems, Springer International Publishing, Cham, 2021, pp. 30–42. `doi:10.1007/978-3-030-82472-3_4`.

[11] D. Owuor, A. Laurent, 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. `doi:10.1109/FUZZ-IEEE.2019.8858883`.

[12] D. O. Owuor, A. Laurent, J. O. Orero, Mining fuzzy temporal gradual emerging patterns, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (2021). `doi:10.1142/S0218488521500288`.

[13] L. Di-Jorio, A. Laurent, M. Teisseire, Mining frequent gradual itemsets from large databases, in: Advances in Intelligent Data Analysis VIII, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 297–308. `doi:10.1007/978-3-642-03915-7_26`.

[14] T. D. Clémentin, T. F. L. Cabrel, K. E. Belise, A novel algorithm for extracting frequent gradual patterns, Machine Learning with Applications 5 (2021) 100068. `doi:10.1016/j.mlwa.2021.100068`.

[15] B. Negrevergne, A. Termier, M.-C. Rousset, J.-F. Méhaut, Paraminer: a generic pattern mining algorithm for multi-core architectures, Data Mining and Knowledge Discovery 28 (3) (2014) 593–633. `doi:10.1007/s10618-013-0313-2`.

[16] D. Owuor, A. Laurent, J. Orero, O. Lobry, Gradual pattern mining tool on cloud, Extraction et Gestion des Connaissances: Actes EGC'2021 (2021).

[17] F. J. Solis, R. J.-B. Wets, Minimization by random search techniques, Mathematics of Operations Research 6 (1) (1981) 19–30. `doi:10.1287/moor.6.1.19`.

[18] Z. B. Zabinsky, Pure Random Search and Pure Adaptive Search, Springer US, Boston, MA, 2003, pp. 25–54. `doi:10.1007/978-1-4419-9182-9_2`.

[19] Z. Zabinsky, Random Search Algorithms, American Cancer Society, 2011. `doi:10.1002/9780470400531.eorms0704`.

[20] P. Fränti, J. Kivijärvi, Randomised local search algorithm for the clustering problem, Pattern Analysis & Applications 3 (4) (2000) 358–369. `doi:10.1007/s100440070007`.

[21] H. H. Hoos, T. Stützle, Stochastic local search: Foundations and applications, Elsevier, 2004.

[22] H. Ishibuchi, T. Murata, Multi-objective genetic local search algorithm, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 119–124. `doi:10.1109/ICEC.1996.542345`.

[23] D. S. Johnson, C. H. Papadimitriou, M. Yannakakis, How easy is local search?, Journal of Computer and System Sciences 37 (1) (1988) 79–100. `doi:10.1016/0022-0000(88)90046-3`.

[24] M. Hossain, T. Tasnim, S. Shatabda, D. M. Farid, Stochastic local search for pattern set mining, in: The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), 2014, pp. 1–6. doi:10.1109/SKIMA.2014.7083547.

[25] J. Holland, Adaptation in natural and artificial systems, univ. of mich. press, Ann Arbor (1975).

[26] J. R. Koza, J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.

[27] S. Mirjalili, Genetic Algorithm, Springer International Publishing, Cham, 2019, pp. 43–55. doi:10.1007/978-3-319-93025-1_4.

[28] M. M. J. Kabir, S. Xu, B. H. Kang, Z. Zhao, Comparative analysis of genetic based approach and apriori algorithm for mining maximal frequent item sets, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 39–45. doi:10.1109/CEC.2015.7256872.

[29] M. Saravanan, V. L. Jyothi, A novel approach for sequential pattern mining by using genetic algorithm, in: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCI-CCT), 2014, pp. 284–288. doi:10.1109/ICCICCT.2014.6992971.

[30] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

[31] S. Rajamohana, K. Umamaheswari, Hybrid approach of improved binary particle swarm optimization and shuffled frog leaping for feature selection, Computers & Electrical Engineering 67 (2018) 497–508. doi:https://doi.org/10.1016/j.compeleceng.2018.02.015.

[32] S. Mishra, D. Mishra, S. K. Satapathy, Particle swarm optimization based fuzzy frequent pattern mining from gene expression data, in: 2011 2nd

International Conference on Computer and Communication Technology (ICCCT-2011), 2011, pp. 15–20. `doi:10.1109/ICCCT.2011.6075204`.

[33] M. Shruti, M. Debahuti, S. Sandeep, Ku., Fuzzy frequent pattern mining from gene expression data using dynamic multi-swarm particle swarm optimization, Procedia Technology 4 (2012) 797–801, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012). `doi:https://doi.org/10.1016/j.protcy.2012.05.130`.

[34] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1) (1996) 29–41. `doi:10.1109/3477.484436`.

[35] M. Dorigo, T. Stützle, Ant Colony Optimization: Overview and Recent Advances, Springer International Publishing, Cham, 2019, pp. 311–351. `doi:10.1007/978-3-319-91086-4_10`.

[36] T. A. Runkler, Ant colony optimization of clustering models, International Journal of Intelligent Systems 20 (12) (2005) 1233–1251. `doi:10.1002/int.20111`.

[37] T. Stützle, H. H. Hoos, Max–min ant system, Future generation computer systems 16 (8) (2000) 889–914. `doi:https://doi.org/10.1016/S0167-739X(00)00043-1`.

[38] D. Harris, S. Harris, Digital design and computer architecture, Morgan Kaufmann, 2010.

[39] G. O'Regan, Binary Number System, Springer International Publishing, Cham, 2018, pp. 53–56. `doi:10.1007/978-3-030-02619-6_12`.

[40] L. Nolle, On a hill-climbing algorithm with adaptive step size: Towards a control parameter-less black-box optimisation algorithm, in: B. Reusch (Ed.), Computational Intelligence, Theory and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 587–595. `doi:10.1007/3-540-34783-6_56`.

[41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2009.

[42] R. Vaz, V. Shah, A. Sawhney, R. Deolekar, Automated big-o analysis of algorithms, in: 2017 International Conference on Nascent Technologies in Engineering (ICNTE), 2017, pp. 1–6. `doi:10.1109/ICNTE.2017.7947882`.

[43] A. P. Piotrowski, J. J. Napiorkowski, A. E. Piotrowska, Population size in particle swarm optimization, Swarm and Evolutionary Computation 58 (2020) 100718. `doi:https://doi.org/10.1016/j.swevo.2020.100718`.

[44] D. Dua, C. Graff, UCI machine learning repository (2017).
URL `http://archive.ics.uci.edu/ml`

[45] M. Patrício, J. Pereira, J. Crisóstomo, P. Matafome, M. Gomes, R. Seiça, F. Caramelo, Using resistin, glucose, age and bmi to predict the presence of breast cancer, BMC Cancer 18 (1) (2018) 29. `doi:10.1186/s12885-017-3877-1`.

[46] G. Hoffmann, A. Bietenbeck, R. Lichtinghagen, F. Klawonn, Using machine learning techniques to generate laboratory diagnostic pathways—a case study, Journal of Laboratory and Precision Medicine 3 (6) (2018). `doi:10.21037/jlpm.2018.06.01`.

[47] B. Rozemberczki, P. Scherer, O. Kiss, R. Sarkar, T. Ferenci, Chickenpox cases in hungary: a benchmark dataset for spatiotemporal signal processing with graph neural networks (2021). `arXiv:2102.08100`.

[48] A. Metzger, P. Leitner, D. Ivanović, E. Schmieders, R. Franklin, M. Carro, S. Dustdar, K. Pohl, Comparing and combining predictive business process monitoring techniques, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (2) (2015) 276–290. `doi:10.1109/TSMC.2014.2347265`.

[49] S. De Vito, E. Massera, M. Piga, L. Martinotto, G. Di Francia, On field calibration of an electronic nose for benzene estimation in an urban pollution

monitoring scenario, Sensors and Actuators B: Chemical 129 (2) (2008) 750–757. `doi:10.1016/j.snb.2007.09.060`.

[50] C. Gondek, D. Hafner, O. R. Sampson, Prediction of failures in the air pressure system of scania trucks using a random forest and feature engineering, in: Advances in Intelligent Data Analysis XV, Springer International Publishing, Cham, 2016, pp. 398–402.

[51] F. Bouchette, OREME: the coastline observation system (2019).
URL `https://oreme.org/observation/ltc/`

[52] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012.

[53] J. Carrasco, S. García, M. Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review, Swarm and Evolutionary Computation 54 (2020) 100665. `doi:https://doi.org/10.1016/j.swevo.2020.100665`.

[54] A. Laurent, B. Negrevergne, N. Sicard, A. Termier, Pgp-mc: Towards a multicore parallel approach for mining gradual patterns, in: H. Kitagawa, Y. Ishikawa, Q. Li, C. Watanabe (Eds.), Database Systems for Advanced Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 78–84.