Universidad Autónoma del Estado de México

Facultad de Ingeniería

Diseño de una metaheurística para el minado de reglas de asociación en bases de datos transaccionales

T E S I S

PARA OBTENER EL GRADO DE:

Maestro en Ciencias de la Ingeniería

PRESENTA:

Ing. Gretel Bernal Baró

TUTOR ACADÉMICO:
Dra. Rosa María Valdovinos Rosas

TUTORES ADJUNTOS:
Dr. Marcelo Romero Huertas
Dr. José Raymundo Marcial Romero

Toluca, México, 2018

Agradecimientos

Durante esta investigación fueron muchas las personas que contribuyeron al desarrollo y conclusión de la misma. En primer lugar, quiero agradecer a mi familia, en especial a mi abuela y mi madre por contar con su apoyo incondicional en todo momento, por los valores que me han inculcado, por ser mi fortaleza y por brindarme una vida llena de aprendizajes experiencias y sobre todo felicidad.

A mis familiares residentes en México, mi tía More, José Emilio, Mariza, Natalia y Aitana, gracias a ustedes fue menor la nostalgia de estar lejos de Cuba.

A Darwin, por ser una persona importante en mi vida, por haberme apoyado en las buenas y en las malas, sobre todo por su paciencia y amor incondicional.

Deseo expresar mi agradecimiento a la directora de esta tesis, la Dra. Rosa María Valdovinos Rosas, por la dedicación y apoyo que ha brindado a este trabajo, por el respeto a mis sugerencias e ideas y por su rigor y exigencia durante la dirección de esta Tesis. Gracias por la confianza ofrecida desde que llegué a esta universidad.

A los Doctores José Francisco Martínez Trinidad y al Dr. Jesús Ariel Carrasco Ochoa, por su oportuna guía en el desarrollo de esta investigación.

A mis amigos, pues han estado en todo momento disponibles para brindarme su ayuda, en especial a Diana Alejandra Mendoza, Liliana Lara y Jose Ángel Álvarez.

Por último y no menos importante, a la UAEMEX por haber permitido formar parte de su estudiantado, a CONACyT por el apoyo económico sin el cual no hubiera sido posible llegar a este punto y a México por haberme abierto sus puertas y hacerme sentir como una ciudadana más.

Resumen

La minería de reglas de asociación se ha convertido en un área de interés en los últimos años. La misma ha sido aplicada para reconocer hábitos de compra, análisis de crímenes, en la bioinformática, medicina, seguridad de redes, entre otros. El objetivo de la Minería de Reglas de Asociación (RAs) es encontrar asociaciones interesantes de la forma "si antecedente entonces consecuente", entre combinaciones de los valores de los atributos o ítems que describen a los objetos de un conjunto de datos.

En el enfoque tradicional de minado de reglas de asociación, se generan un gran número de reglas, seleccionar entre ellas un subconjunto de reglas de utilidad es una tarea costosa. Por esta razón, en la literatura se ha propuesto el uso de algoritmos de optimización como alternativa de solución. Al respecto, en la literatura se presenta el algoritmo Enjambre de partículas (PSO) como uno de los que tiene un mejor desempeño en cuanto al corto tiempo de ejecución que emplea a la hora de minar RAs. Sin embargo, este algoritmo tiene como deficiencia que tiende a converger prematuramente en reglas con bajos valores de aptitud (soporte y confianza) en conjuntos de datos con gran cantidad de ítems.

Por ello, en esta Tesis se propone un algoritmo metaheurístico basado en el algoritmo PSO, que mina reglas de asociación sin la necesidad de obtener los patrones frecuentes e introduce un método de búsqueda guiada que estimula el minado de reglas de asociación de calidad. Además, de proponer la integración de una estructura de datos resumen que reduce el número de filas con respecto al conjunto de datos original, con la intención de reducir el tiempo de ejecución empleado por el algoritmo.

De acuerdo con los experimentos realizados, el algoritmo propuesto obtiene un conjunto de reglas de asociación de mayor calidad que las obtenidas por otros algoritmos basados en PSO para el minado de reglas de asociación. Además, se puede apreciar que el tiempo de ejecución alcanzado por el algoritmo propuesto es competitivo con respecto a las propuestas con que fue comparado. Por último, el algoritmo propuesto

mantiene un desempeño estable al variar el número de ítems y transacciones existentes en el conjunto de datos. Es decir que independientemente de la naturaleza y volumen de los datos tiende a generar un subconjunto de reglas de interés para el problema de estudio.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La minería de datos (MD) constituye un campo de las ciencias de la computación estudiada por su creciente número de aplicaciones en la sociedad actual, con el principal objetivo de extraer de forma automática información relevante, valiosa y no evidente en conjuntos de datos extensos que ayude a responder preguntas que en general consumen o demandan tiempo de los analistas, permitiendo tomar decisiones confiables basadas en la información descubierta [1].

Entre las principales tareas que explotan las bondades de las técnicas de MD se pueden mencionar [2]: identificación con antelación de potenciales terroristas a través del uso de los datos almacenados en los conjuntos de datos comerciales; detección de fraudes en las tarjetas de crédito; reconocer hábitos de compra y predicción del mercado con el objetivo de llevar a cabo diferentes estrategias tales como: realizar ventas cruzadas, planificar las promociones de determinados artículos, para el diseño de catálogos y permitir identificar patrones de compra beneficiosos para situar estratégicamente productos en los grandes almacenes [3].

Dependiendo del problema de estudio el proceso de MD se puede abordar desde dos perspectivas [1]: desde el punto de vista predictivo o desde el punto de vista descriptivo. La investigación presentada en esta tesis tiene un enfoque descriptivo ya que explota el uso de técnicas de MD como las reglas de asociación (RAs) [4] para la identificación y extracción de las asociaciones existentes entre los *ítems* (atributos) de conjuntos de datos transaccionales.

Dado un conjunto de datos de transacciones donde cada transacción está compuesta por un conjunto *ítems*, una RA tiene la forma $X \Rightarrow Y$, con antecedente X y consecuente Y, donde X y Y son ambos un conjunto frecuente de *ítems* o *itemsets* frecuentes y la intersección de X y Y es vacía, $X \cap Y = \emptyset$. Una RA es validada con el uso de medidas como el soporte y la confianza que seleccionan reglas interesantes del conjunto de todas

las reglas posibles [5].

Muchos estudios se concentran en el desarrollo de algoritmos para la extracción de *itemsets* frecuentes debido a que es la tarea más compleja en el proceso de extracción de RAs. Agrawal et al. [6] fueron los pioneros en esta área y los primeros en proponer un algoritmo para extraer las RAs entre los *ítems* de un conjunto de datos transaccional, al cual denominaron "Algoritmo Apriori". El algoritmo Apriori tiene como debilidad que realiza varios accesos al conjunto de datos para contabilizar la frecuencia de los candidatos a *itemsets* frecuentes directamente en ella.

Seguido de Apriori, varios algoritmos fueron desarrollados para el minado de *itemsets* frecuentes tales como *FP-Growth* propuesto por Han et al. [7]. *FP-Growth* es más rápido que Apriori, la ventaja de este algoritmo es que construye una estructura compacta llamada *FP-tree*, debido al uso de esta estructura el algoritmo solo requiere 2 accesos al conjunto de datos. Sin embargo, la construcción del árbol *FP-Tree* necesita más memoria en conjuntos de datos transaccionales que involucran gran cantidad de *ítems* y transacciones.

Después de extraer los *itemsets* frecuentes, cuando el número de *ítems* en el conjunto de datos es grande, comúnmente se generan un gran número de reglas, seleccionar, entre ellas, un subconjunto de reglas de calidad es una tarea costosa. Por esta razón, recientemente, metaheurísticas de inspiración biológica como Colonia de Hormigas [8, 9], Enjambre de Partículas [10, 11, 12, 13] y Algoritmos Genéticos (GA) [12, 14, 15] se han propuesto para minar subconjuntos de RAs de alta calidad. A partir de estos trabajos, los mejores resultados, en cuanto al corto tiempo de ejecución empleado a la hora de realizar esta tarea [16, 17], han sido reportados por aquellos basados en Enjambre de Partículas (PSO). Sin embargo, el algoritmo PSO tiene como deficiencia que tiende a converger prematuramente en soluciones de baja calidad en conjuntos de datos que involucran un mayor número de *ítems* [18]. Por esta razón, en esta Tésis, se propone desarrollar una nueva metaheurística, basada en PSO, para minar RAs, sin generar *itemsets* frecuentes, pero con una estrategia de evolución guiada. Experimentos sobre conjuntos de datos reales muestran que la metaheurística propuesta encuentra RAs de mejor calidad que otros algoritmos del estado del arte.

1.1. Objetivos

Diseñar y desarrollar una metaheurística para la identificación automática de reglas de asociación que permita predecir las posibles relaciones entre dos o más *ítems* en conjuntos de datos transaccionales. En específico la investigación persigue:

- Integrar y analizar la mina de datos para su tratamiento como conjunto de datos

transaccional.

- Desarrollar una metaheurística, para minar las RAs relevantes en grandes conjuntos de datos transaccionales, sin generar patrones frecuentes, el mismo adaptará varios algoritmos heurísticos para la integración de una metaheurística.
- Detección de las mejoras de la metaheurística propuesta con respecto a otras propuestas existentes en el estado del arte.

1.2. Hipótesis

Con el diseño y desarrollo de una metaheurística para la extracción automática de reglas de asociación en conjuntos de datos transaccionales, será posible obtener un conjunto de reglas que determinen conocimiento de interés para al problema de estudio.

1.3. Estructura de la tesis

La Tesis está estructurada de la siguiente manera, en el primer Capítulo se exponen las razones y objetivos del desarrollo de la investigación. El segundo y tercer capítulo está conformado por la fundamentación teórica y el estado del arte respectivamente, exhibiendo los conceptos de utilidad para el desarrollo de la investigación y los trabajos relacionados que sirven de base para el desarrollo de la misma. El Capítulo 4 y 5 incluye la descripción de la estrategia de solución y la experimentación desarrollada. Por último, las conclusiones de los resultados alcanzados y trabajos futuros se encuentran en el Capítulo 6.

Capítulo 2

Marco Teórico

Este capítulo incluye la teoría que sustenta la investigación. Por ende, la exposición de conceptos comienza con la explicación de qué es la MD, metodologías de MD, qué son las RAs, medidas y clasificación de las RAs. Por último, algoritmos exactos y metaheurísticas basadas en PSO utilizadas para el minado de RAs.

2.1. Minería de datos

Larose en [1], define la MD como "el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde cantidades de datos almacenados en distintos formatos". Es decir, la tarea fundamental de la MD es encontrar modelos inteligibles a partir de los datos. Para que este proceso sea efectivo deberá ser automático o semi-automático (asistido) y el uso de los patrones descubiertos deberá ayudar a tomar decisiones que reporten beneficios a la organización.

Llevar a cabo el proceso de MD requiere la aplicación de una metodología estructurada para la utilización eficiente de las técnicas y herramientas disponibles. En la literatura predomina el uso de tres metodologías [19]: KDD (*Knowledge Discovery in Databases*), SEMMA (el acrónimo corresponde a las fases básicas del proceso: *Sample, Explore, Modify, Model, Assess*) y CRISP-DM (*Cross Industry Standard Process for Data Mining*). No obstante, por la adaptación al problema de estudio en esta investigación se utilizará la metodología CRISP-DM (Figura 2.1).

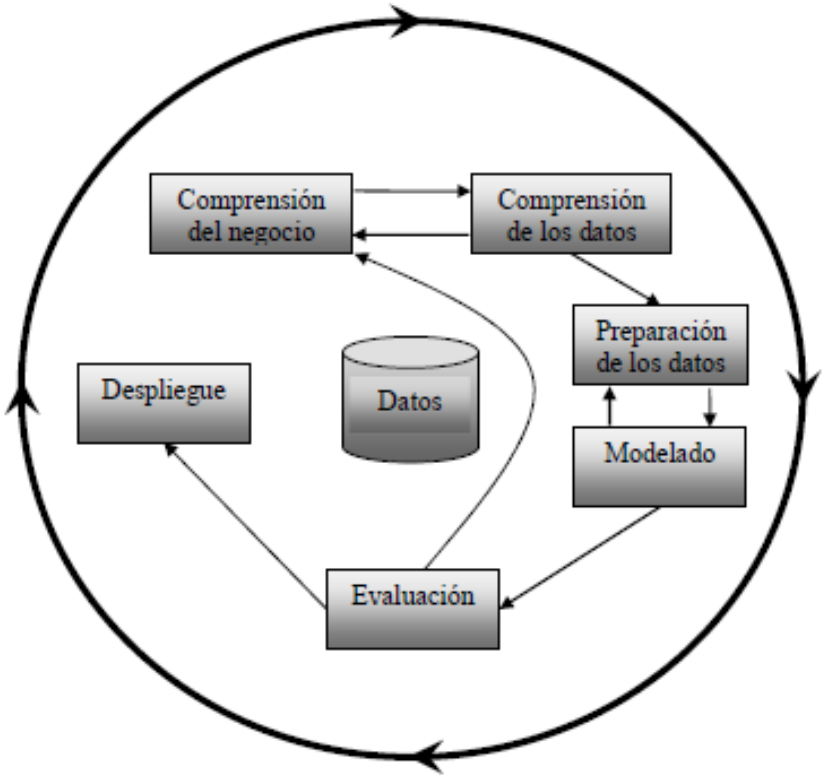

Figura 2.1: Diagrama de la metodología CRISP-DM [20]

2.1.1. Metodología CRISP-DM

La metodología CRISP-DM según Moro [21] es considerada la guía de referencia más utilizada en el desarrollo de proyectos de MD, por el nivel de detalle con que describe las tareas en cada fase del proceso. Esta metodología consta de 6 fases:

a) Comprensión del negocio: Esta fase consiste en comprender o definir el problema, permite entender los objetivos y requisitos que tendrá el proyecto.

b) Comprensión de los datos: La fase comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse con los mismos. Este proceso involucra la descripción de los datos, estableciendo los volúmenes de información con que se trabajará.

c) Preparación de los datos: Esta fase cubre todas las actividades para construir el conjunto de datos final desde los datos iniciales. Incluye tareas como la limpieza de los datos aplicando técnicas de normalización, discretización de campos numéricos, tratamiento de valores nulos, entre otros.

d) Modelado: La fase de modelado es considerada una de las más importantes en el proceso de MD. Es en esta fase donde se aplican la o las técnicas de minería para la deducción del conocimiento.

e) Evaluación: En esta fase se evalúa el modelo, teniendo en cuenta el cumplimiento de los criterios de éxito definidos. Si el modelo generado es válido en función de los criterios de éxito establecidos se procede a la explotación del modelo. También se lleva a cabo una revisión del proceso de MD, con el objetivo de identificar elementos que pudieran ser mejorados.

f) Despliegue: Una vez que el modelo ha sido construido y validado sigue la fase de despliegue, donde el conocimiento generado necesitará ser organizado y presentado de manera que el cliente lo pueda usar.

En la siguiente sección se describe la técnica de minería utilizada para la deducción de las asociaciones existentes entre los *ítems* de conjuntos de datos.

2.2. Reglas de Asociación

En la MD las RA representan una técnica para descubrir asociaciones o correlaciones interesantes a partir de un conjunto de datos. Los algoritmos de RA comúnmente son aplicados a bases de datos transaccionales. Estas bases de datos se refieren a una colección de registros de transacciones en el que cada transacción está compuesta por un conjunto de *ítems* [5] (Tabla 2.1).

La definición de RA plantea que: sea $I = \{i_1, i_2, ..., i_n\}$ un conjunto de n atributos binarios llamados *ítems* y $D = \{t_1, t_2, ..., t_n\}$ un conjunto de transacciones almacenadas en un conjunto de datos. Cada transacción en D tiene un Id (identificador) único y contiene un subconjunto de *ítems* de I (un conjunto de *ítems* es denominado *itemset*). De este modo, una RA se define como una implicación de la forma: X entonces Y ($X \Rightarrow Y$), con antecedente X y consecuente Y, en la cual X e Y son subconjuntos de I ($X, Y \subseteq I$) e *itemset* de intersección vacía ($X \cap Y$), es decir sin *ítems* en común [22].

Por ejemplo, con las transacciones de la Tabla 2.1 y tomando en cuenta el formato de las RA se tienen las posibles asociaciones hipotéticas entre productos:

Pañales $\Rightarrow$ Cervezas

Tabla 2.1: Ejemplo de conjunto de datos transaccional

Transacciones	
TID	Artículos
1	Pan, Queso, Jamón
2	Pan, Pañales, Cervezas
3	Leche, Pañales, Cerveza
4	Pan, Queso, Pañales, Cerveza
5	Pan, Queso, Jamón, Cerveza

$Queso \wedge Pan \Rightarrow Jamón$.
$Cerveza \wedge Pan \Rightarrow Jamón \wedge Queso$.

2.2.1. Métricas más utilizadas para seleccionar las RAs de interés

El problema de minería de RAs consiste en encontrar todas las reglas interesantes a partir de un conjunto de transacciones. Las medidas o métricas más utilizadas para seleccionar las RAs de interés del conjunto de todas las reglas posibles son [1]: el soporte y la confianza. Ellas describen la calidad, probabilidad, generalidad y fiabilidad de la regla.

El soporte de una regla $X \Rightarrow Y$ en un conjunto de transacciones T, es la fracción de transacciones de T que contienen los *ítems* de $X \cup Y$. En otras palabras, se entiende por soporte a la probabilidad con que aparecen un conjunto de *ítems* involucrados en una serie de transacciones [23].

$$Soporte(X \Rightarrow Y) = P(X \cup Y) \tag{2.1}$$

La Confianza (también llamada precisión) de una regla se define como el número de instancias que la regla predice correctamente y es calculada como la fracción de transacciones en el conjunto de datos que contiene ambos, antecedente y consecuente de la cantidad de transacciones que contienen su antecedente [23]. Si el resultado está más cerca de 1, la relación de X implica Y es más fuerte.

$$Confianza(X \Rightarrow Y) = \frac{Soporte(X \cup Y)}{Soporte(X)} \tag{2.2}$$

2.2.2. Clasificación de las Reglas de Asociación

Las RAs pueden ser clasificadas en varios grupos, de acuerdo con los siguientes criterios [24]:

a) Según los tipos de valores que manejan las reglas:

- Si una regla se refiere a asociaciones entre la presencia o ausencia de un *ítem*, se trata de una regla de asociación booleana. La Regla 1 representa este tipo de clasificación.

  Regla 1: *"Si compra computadora ⇒ compra software contable"*
- Si una regla describe asociaciones entre *ítems* o atributos cuantitativos, entonces se dice que es una regla de asociación cuantitativa. En ella los *ítems* o atributos son divididos en intervalos. La Regla 2 es un ejemplo de asociación cuantitativa.

  Regla 2: *"edad (X, «20...30») ∧ ingreso (X, «*500.000*...1.000.000») ⇒ compra (X, computador portátil)"*.

  Lo que indica que existe una relación entre la edad del cliente X, los ingresos y el tipo de computadora que este cliente compra.

b) Con base en las dimensiones de datos que involucra una regla:

- Si en los *ítems* o atributos de una regla de asociación se referencia solo en una dimensión, se trata de una regla de asociación de dimensión simple. La regla de asociación 1 y la 3 hace referencia a este tipo de reglas ya que solo corresponde al hecho de aparecer o no en la cesta de la compra.

  Regla 3: *"Si compra vino ⇒ compra gaseosa"*
- Si se referencia dos o más dimensiones se dice que es una regla de asociación multidimensional. Se puede incrementar la dimensión de una regla incluyendo, por ejemplo, el tiempo en que fue comprado y el cliente. Al reescribir la Regla 3 se tiene:

  Regla 4: *"Si Cliente(X) compra vino en Tiempo(Marzo) ⇒ compra gaseosa"*

c) Con base en los niveles de abstracción que involucra la regla:

Algunos métodos para encontrar RA permiten incorporar a las reglas diferentes niveles de abstracción representados por conceptos que aglutinan otros conceptos o *ítems*. Por ejemplo, supóngase que en un conjunto de RA se tiene las reglas 5 y 6. Nótese que en las reglas de ejemplo los *ítems* son referenciados a diferentes niveles de

abstracción («*computadora*» es un nivel más alto de abstracción que «*computadora portátil*»).

Regla 5: *"edad (X, «20...30») ⇒ compra (X, «computadoras portátiles»)"*

Regla 6: *"edad (X, «20...30») ⇒ compra (X, «computadoras»)"*.

d) Instantáneas o secuenciales:

Estas reglas expresan patrones de comportamientos secuenciales que se dan en instantes distintos pero cercanos de tiempo. Por ejemplo, si un cliente compra una cuchilla y una loción para después del afeitado, probablemente comprará crema para afeitar la próxima vez que vaya a comprar. Existen algoritmo de RA, que utilizan métodos específicos para encontrar las secuencias.

2.3. Minado de Reglas de Asociación en conjuntos de datos transaccionales

En esta sección se presentan: las formas de representar un conjunto de datos transaccional, así como los algoritmos exactos más utilizados en el minado de patrones frecuentes y RAs en colecciones de datos transaccionales.

2.3.1. Representación de un conjunto de datos transaccional

Según Adamo [25] existen 3 formas de representar un conjunto de datos transaccional.

- Representación binaria (Figura 2.2 a): almacena por cada transacción un vector binario donde un 1 en la posición i-ésima denota la presencia del i-ésimo *ítem* en la transacción y un 0 denota la ausencia del mismo.
- Representación horizontal (Figura 2.2 b): consiste en un conjunto de pares (r, cas), donde r es el identificador de la transacción y cas es una secuencia canónica de *itemsets*. Esta representación almacena por cada transacción una lista con los *ítems* presentes en ella.
- Representación vertical (Figura 2.2 c): consiste en un conjunto de pares (a, lista) donde a es un *ítem* y la lista es una secuencia ordenada de los identificadores de las transacciones donde el *ítem* está presente.

El proceso de minado de reglas de asociación consta de dos pasos fundamentales:

	a	b	c	d	e
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

(a) Booleana

r		cas
1	→	abde
2	→	bce
3	→	abde
4	→	abce
5	→	abcde
6	→	bcd

(b) Horizontal

a	b	c	d	e
↓	↓	↓	↓	↓
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

(c) Vertical

Figura 2.2: Representaciones para A={a,b,c,d,e} [25]

- Primero se buscan en el conjunto de transacciones, los *itemsets* frecuentes (conjuntos de *ítems* frecuentes). Nótese que para que una regla $X \Rightarrow Y$ sea interesante $X \cup Y$ debe ser un *itemsets* frecuente.
- A partir de los conjuntos frecuentes de *ítems* son generadas las reglas de asociación interesantes.

A continuación, se describen los algoritmos exactos más utilizados en el minado de *itemsets* frecuentes.

2.3.2. Algoritmos para el minado de *itemsets* frecuentes

La generación de *itemsets* frecuentes es el paso más costoso del proceso de minado de RAs y en este paso se enfocan la mayoría de los trabajos existentes en la literatura. Esto es debido a que el tamaño del espacio de búsqueda de los *itemsets* frecuentes depende exponencialmente del tamaño del conjunto de ítems. Sin embargo, existe una propiedad llamada Clausura Descendente del soporte, mediante la cual no es necesario recorrer todo el espacio de búsqueda para encontrar los conjuntos frecuentes de ítems [6].

Esta propiedad plantea que todo subconjunto de un conjunto frecuente de ítems es frecuente, mientras que todo superconjunto de un conjunto no frecuente de ítems tampoco es frecuente [26]. Como consecuencia de esta propiedad, el espacio de búsqueda asociado queda dividido por una frontera en dos subespacios: el subespacio que sólo contiene *itemsets* frecuentes y el subespacio que sólo contiene *itemsets* no frecuentes.

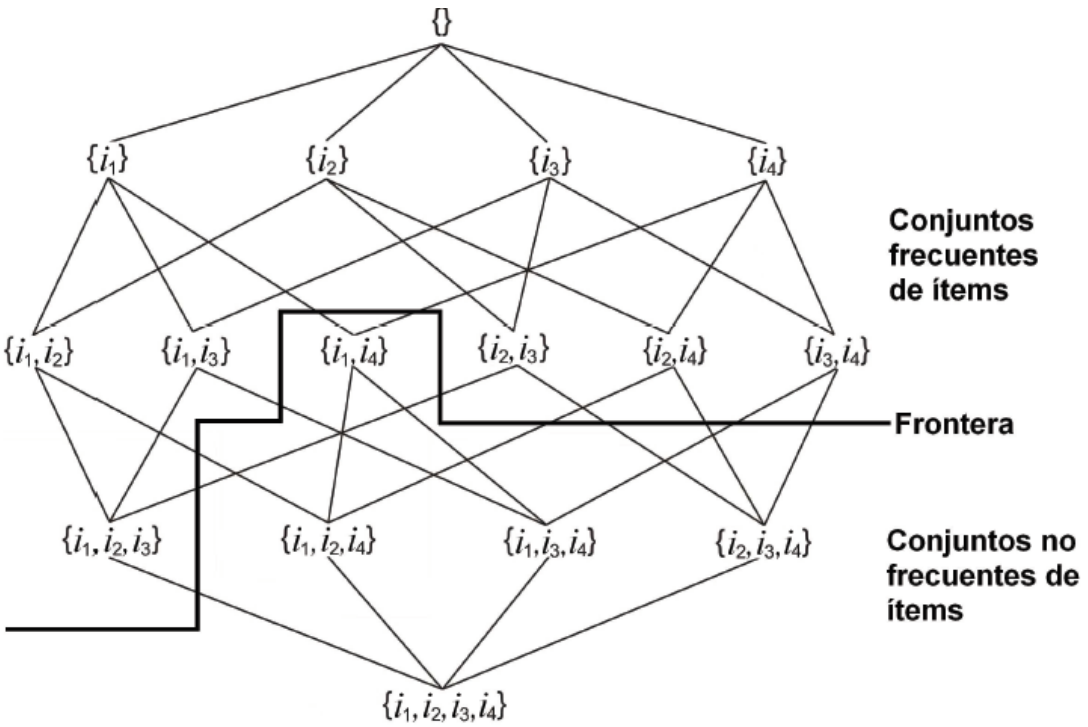

Figura 2.3: Retículo formado por el *itemset* I = $\{i_1, i_2, i_3, i_4\}$ [26]

La figura 2.3 muestra la frontera entre los subespacios para el conjunto de transacciones T = $\{\{i_1, i_2, i_3\}, \{i_2, i_4\}, \{i_3, i_4\}, \{i_1, i_2, i_3, i_4\}\}$ dado el umbral de mínimo soporte minSupp = 0,5 (en este caso es considerado un *itemset* frecuente si aparece al menos en el 50 % de las transacciones).

Los *itemsets* frecuentes pueden ser generados de dos formas:

- En amplitud: Se generan todos los conjuntos frecuentes de *ítems* de tamaño k antes de generar los conjuntos de *ítems* de tamaño $(k+1)$ [Apriori [27], Apriori-Tid [27], Apriori-Hybrid [27], DHP [28], Partition [29], IHP [30]].
- En profundidad: Recursivamente se generan los conjuntos *ítems* por cada rama de la estructura arbórea [FP-Growth [7], Patricia Trie-Mine [31], CT-ITL [32], CT-PRO [33], Apriori-TFP [34]].

El Algoritmo más representativo que genera todos los *itemsets* candidatos a frecuentes de tamaño $(k+1)$, este proceso es llamado generación de candidatos, es el Algoritmo

Apriori. Este algoritmo debe su nombre a que usa conocimiento a priori para la generación de *itemsets* frecuentes. Además, es el primero que hace uso de la propiedad Clausura descendente del soporte.

El Algoritmo 1 [27] muestra cómo se buscan los *itemsets* frecuentes usando generación de candidatos. En este algoritmo, C_i contiene los conjuntos de *ítems* candidatos a ser frecuentes de tamaño i y L_i los conjuntos frecuentes de *ítems* de tamaño $i-1$. Primeramente, son obtenidos y almacenados en C_i todos los *ítems* con sus respectivos valores de soporte. Luego para cada elemento de C_i se verifica si el soporte es mayor o igual que el mínimo soporte establecido, si es mayor se va agregar a L_i. Posteriormente en cada iteración del algoritmo se generan los conjuntos de *ítems* candidatos a frecuentes de tamaño $i+1$ combinando los conjuntos de L_i hasta que los *itemset* generados no cumplan con la condición de mínimo soporte. Lo que permite ir eliminando posibles combinaciones. Aquellas que no cumplan con los requerimientos de soporte no entrarán en el análisis, esto se aprovecha en la construcción de candidatos, para no considerar todas las opciones.

Algoritmo 1: Algoritmo *Apriori* [1]

Entrada: Conjunto de datos T

Mínimo Soporte $minSupport$

Salida: Retorna los *itemset* frecuentes L

```
1 L_1 = {itemsfrecuentes}
2 para k ← 2 a L_{k-1} ≠ ∅ hacer
3     C_k = GenerarCandidatosdeL_{k-1}
4     para cada t ∈ T hacer
5         Incrementar la cantidad de todos los candidatos en C_k si están presentes en t
6         L_k = candidatos en C_k con soporte mayor que el minSupport
7 devolver ∪_k L_k
```

A pesar de ser el algoritmo más usado en la búsqueda de patrones frecuentes, éste presenta como deficiencia que realiza varias pasadas al conjunto de datos, al buscar todos los conjuntos frecuentes unitarios contando sus ocurrencias directamente en ella. Además, el conteo de soporte de los candidatos es costoso debido a que el número de subconjuntos frecuentes en cada candidato es cada vez mayor. Otra desventaja es que hay que hacer tantos recorridos como sea necesario para encontrar todos los *itemsets*

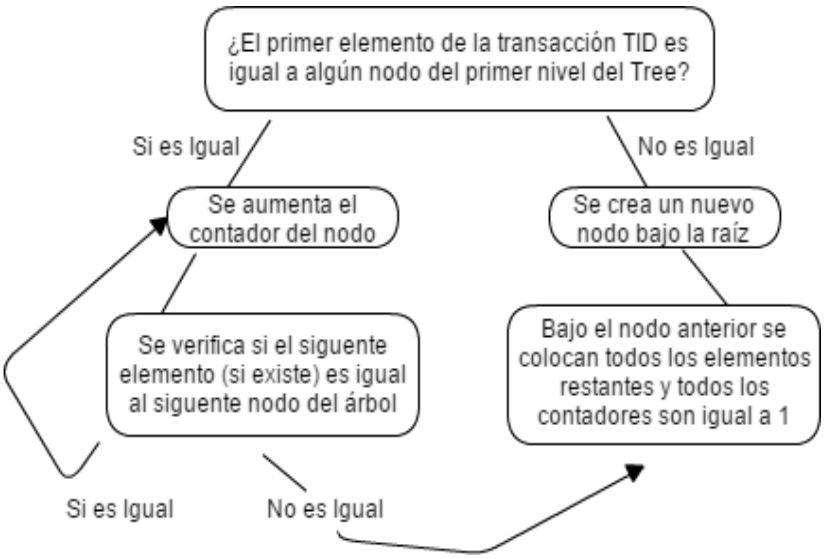

Figura 2.4: Creación del árbol FP-tree

frecuentes, por lo que no solo es costosa la solución en memoria, sino también en tiempo de ejecución.

Después de Apriori, se desarrollaron varios algoritmos para el minado de *itemsets* frecuentes uno de los más utilizados, el cual genera recursivamente los conjuntos frecuentes de *ítems*, es el algoritmo FP-Growth propuesto por Han et al en [7]. Este algoritmo está basado en una representación de árbol de prefijos del conjunto de transacciones llamado *FP-Tree* (*Frequent Pattern Tree*) [35]. Básicamente el algoritmo se puede descomponer en dos etapas:

- Crear el *FPTree* utilizando el conjunto de datos D y un umbral de soporte mínimo.
  - El primer paso en la construcción del árbol *FPTree* consiste en recorrer el conjunto de transacciones en busca de los *itemsets* frecuentes y luego contabilizar sus respectivas frecuencias. Los *ítems* son ordenados de manera decreciente, de acuerdo con sus frecuencias. De esta forma los *ítems* de mayor frecuencia van a ocupar los nodos de menor nivel (más cercanos a la raíz) en el árbol.
  - Seguidamente se recorre el conjunto de transacciones, solamente con los *itemsets* frecuentes ya ordenados, para componer cada rama del árbol (Figura 2.4).

- Invocar al algoritmo *FP-Growth* con el árbol generado como entrada y retorna como salida el conjunto de *itemset* frecuentes (Algoritmo 2).

Algoritmo 2: Algoritmo FP-Growth [36]

Entrada: Nodo α

Árbol *FPTree*

Salida: Retorna los *itemset* frecuentes

```
1  si FPTree contiene un camino único P entonces
2      para cada combinación β = α1α1...αk de nodos en el camino P hacer
3          generar el patrón α ∪ β con soporte = min(αi‖(1 <= i <= k))
4  en otro caso
5      para cada ai en la tabla de cabeceras de FPTree hacer
6          Generar el patrón β = α ∪ ai con soporte = ai.soporte
7          Construir la base de patrones condicional de β
8          Generar el FP-Tree condicional de β (FP Treeβ)
9          si FPTreeβ ≠ ∅ entonces
10             invocar FP-Growth(FPTreeβ, β);
```

Como puede verse en el Algoritmo 2, en el caso que un *FP-Tree* tenga un solo camino P, entonces todas las combinaciones de patrones que se pueden formar con los *ítems* que representa cada nodo contenido en P son frecuentes, con un soporte igual al mínimo de los *ítems* del subcamino.

De lo contrario, se construye el patrón condicional base y su árbol FP condicional para cada conjunto de elementos frecuentes a_i. El crecimiento del patrón se produce a medida que iteramos el método, ya que el primer *ítem* encontrado constituye el sufijo del *itemset* frecuente y los *ítems* frecuentes encontrados en la próxima iteración forman cada uno junto con el sufijo del paso anterior los nuevos sufijos de los *itemset* frecuentes. En cada paso recursivo, el algoritmo procede a incrementar el sufijo con los *ítems* frecuentes hasta alcanzar el nodo raíz.

El algoritmo *FP-Growth* tiene ventajas operacionales sobre el algoritmo *Apriori*, al no necesitar de la generación de *ítems* candidatos y ser computacionalmente más rápido. El algoritmo busca patrones frecuentes con una corta búsqueda recursiva de prefijos, lo

que en tiempo de ejecución es superior al *Apriori*, debido a que no requiere constantes accesos al conjunto de datos. Sin embargo, el algoritmo tiene como desventaja que es difícil de implementar debido a la compleja estructura de datos que maneja. Además, necesita mucho tiempo para construir la estructura del *FP-Tree* y también necesita más memoria para almacenar las transacciones [37].

2.3.3. Obtención de las reglas de asociación

Una vez que son obtenidos los patrones frecuentes se lleva a cabo el minado de reglas de asociación. El algoritmo más común utilizado en la generación de reglas de asociación fue propuesto en [27]. El mismo consiste en, por cada conjunto de *itemsets* frecuentes, generar todas las reglas posibles dividiendo el *itemsets* en dos subconjuntos disjuntos (Algoritmo 3).

Algoritmo 3: Algoritmo *GenRules* [1]

Entrada: Conjuntos frecuentes de *ítems* F

Mínimo Soporte $minSupport$

Salida: Reglas de asociación RA

```
1 RA ← ∅
2 para cada itemset Z ∈ F hacer
3     para cada X ⊂ Z tal que X ≠ ∅ hacer
4         si supp(Z)/supp(X) ≥ minSupport entonces
5             RA = RA ∪ X → Z/X
6 devolver RA
```

En la literatura, se han reportado otros algoritmos para generar reglas de asociación interesantes a partir de los patrones frecuentes encontrados como [38], [39], [40], [41], [42]. Todos estos algoritmos extraen todo el conjunto de reglas posibles.

2.4. Metaheurísticas para el minado de RAs

Como pudo ser visto en el Algoritmo 3 el proceso de descubrir reglas de asociación puede ser interpretado como un problema de optimización combinatorio, debido a que

las reglas son combinaciones de *ítems* donde cada ítem puede estar en el antecedente o en el consecuente de la regla [43]. Cuando el número de *ítems* es grande, el número de reglas posibles es exponencialmente grande. Por lo tanto, el uso de algoritmos eficientes capaces de lidiar con problemas de gran tamaño, algoritmos heurísticos y particularmente metaheurísticos, son necesarios. Las metaheurísticas son algoritmos de alto nivel que combinan diferentes heurísticas, aplicables a disímiles problemas de optimización con relativamente pocas modificaciones para adaptarlos a un problema específico [44].

Estos algoritmos han demostrado ser de gran utilidad en el minado de patrones frecuentes y minado de RAs, algunos de los algoritmos metaheurísticos de inspiración biológica más utilizados en esta tarea son:

- Colonia de Hormigas [9, 45]: Este algoritmo reproduce el comportamiento natural de las hormigas en la búsqueda de alimento. Cuando estas encuentran alimento (solución candidata al problema), evalúan la fuente y dejan un rastro de feromona, cuya cantidad depende de la calidad del alimento (calidad de la solución encontrada) y la distancia a este. Si las hormigas no detectan dicha sustancia se mueven de forma aleatoria, pero si la perciben es muy probable que se desplacen siguiendo la señal, lo cual a su vez aumenta la cantidad de la feromona. La expulsión de esta sustancia sirve para poder guiar a otras hormigas hacia una buena fuente de alimento.

- Enjambre de Abejas [46, 47]: Este algoritmo está inspirado en el comportamiento de las abejas en la búsqueda de miel. Las abejas se mueven en un espacio de búsqueda multidimensional eligiendo fuentes de néctar (soluciones candidatas al problema) dependiendo de su experiencia pasada y la de sus compañeros de colmena o ajustando su posición. Algunas abejas (exploradoras) vuelan y eligen las fuentes de alimento aleatoriamente sin usar experiencia. Cuando encuentran una fuente de néctar mayor (una solución de mejor calidad), memorizan su posición y olvidan la anterior. De este modo, el algoritmo combina métodos de búsqueda local y búsqueda global, intentando equilibrar el proceso de la exploración y de la explotación del espacio de búsqueda.

- Algoritmos Genéticos [12, 14, 15, 48]: Este algoritmo es una analogía a él proceso de adaptación de los seres vivos, en este algoritmo cada solución candidata es un individuo de una población y se codifica como un vector binario. En cada una de las iteraciones, se aplican operadores evolutivos (selección, cruzamiento y mutación) que combinan y modifican a los individuos de la población creándose una nueva.

Otra de las metaheurísticas utilizadas en el minado de RAs es el algoritmo Enjambre

de Partículas [11, 13, 10, 12, 49, 50]. De los algoritmos anteriormente mencionados el algoritmo Enjambre de Partículas presenta ventajas operacionales con respecto a sus homólogos en cuanto al corto tiempo de ejecución que emplea a la hora de realizar esta tarea [16, 17], debido a que este solo requiere del uso de un vector de velocidad para desplazarse a través del espacio de búsqueda. Es por ello que en la siguiente sección es analizado, con mayor detalle, los pasos específicos de este algoritmo.

2.4.1. Algoritmo Enjambre de Partículas

El algoritmo Enjambre de Partículas (*Particle Swarm Optimization, PSO*) propuesto por Kennedy y Eberhart [51], es una metaheurística poblacional basada en la simulación de modelos sociales simples e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces.

En PSO, se utiliza una población de tamaño fijo. Cada partícula de la población, es una solución candidata al problema y sus movimientos se encuentran acotados al espacio de búsqueda. Este espacio se encuentra definido de antemano y no se permite que las partículas se desplacen fuera de él. Cada individuo o partícula está siempre en continuo movimiento y está compuesta por tres vectores y dos valores de *fitness*:

- Vector $X_i = (x_1, x_2, ..., x_n)$ almacena la posición actual de la partícula en el espacio de búsqueda.
- Vector $PBest_i = (p_1, p_2, ..., p_n)$ almacena la mejor solución encontrada por la partícula hasta el momento.
- Vector velocidad $V_i = (v_1, v_2, ..., v_n)$ almacena el gradiente (dirección) según el cual se moverá la partícula.
- El valor $\mathit{fitness}(X_i)$ almacena el valor de aptitud de la solución actual.
- El valor $\mathit{fitness}(PBest_i)$ almacena el valor de aptitud de la mejor solución local encontrada hasta el momento (vector $PBest_i$).

La Figura 2.5 representa el movimiento de una partícula en el espacio de soluciones. Las flechas de línea punteada representan la dirección de los vectores de: velocidad (v_i^t), mejor posición local (P_{Best}^t) y mejor posición global (G_i^t). La flecha de línea completa representa la dirección que toma la partícula para moverse desde la posición X_i^t hasta la posición X_i^{t+1}. El cambio de dirección de la partícula depende de la influencia de los vectores que intervienen en su movimiento.

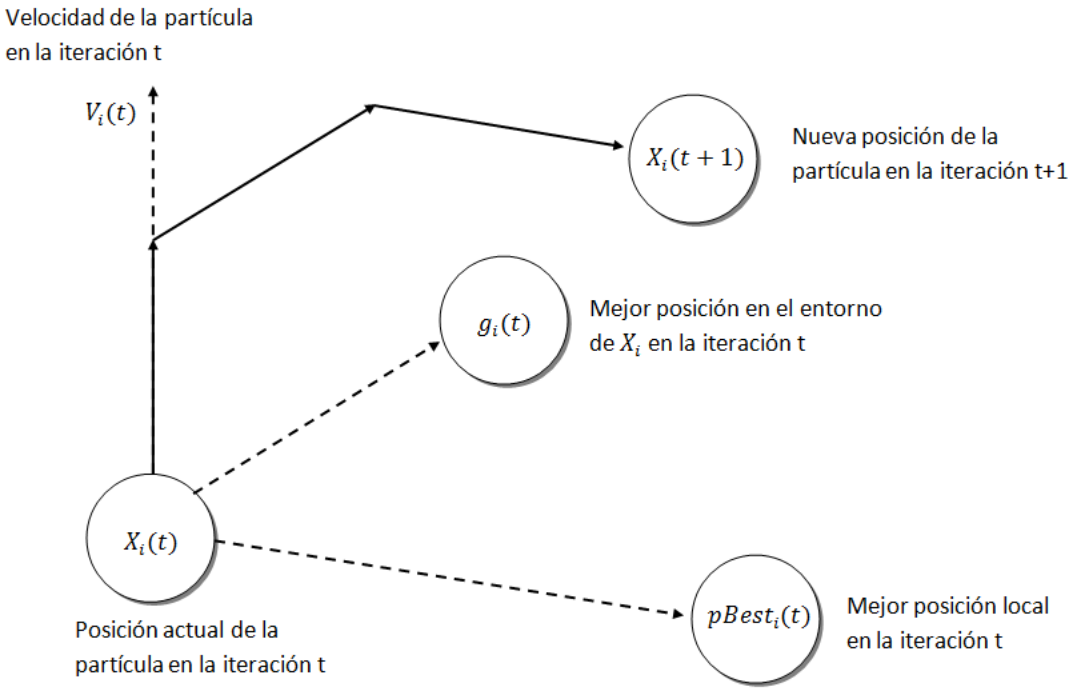

Figura 2.5: Movimiento de una partícula en el espacio de soluciones.

El Algoritmo 4 describe el pseudocódigo del algoritmo de búsqueda PSO, donde, para cada partícula, se inicializa su posición (X_i) y su velocidad (V_i) de manera aleatoria. Una vez que la población es inicializada los individuos comienzan a moverse por el espacio de búsqueda por medio de un proceso iterativo.

Con la nueva posición del individuo, se calcula y actualiza su *fitness* ($fitness_{X_i}$). Además, si el nuevo *fitness* del individuo es el mejor encontrado hasta el momento, se actualizan los valores de mejor posición $PBest_i$ y *fitness* $PBest_i$. En cada iteración el vector de velocidad de la partícula se actualiza tomando en cuenta su experiencia y su entorno mediante la siguiente expresión:

$$V_i^{(t+1)} = w * V_i^t + c_1 * rand_1 * (PBest_i^t - X_i^t) + c_2 * rand_2 * (G_i^t - X_i^t) \tag{2.3}$$

Donde w representa el factor inercia, c_1 y c_2 las constantes de aceleración, $rand_1$ y $rand_2$ son valores aleatorios pertenecientes al intervalo (0,1) y G_i representa la posición de la partícula con el mejor $PBest_i$ en el entorno de X_i. Finalmente, se actualiza la

posición de la partícula de la siguiente forma:

$$X_i^{(t+1)} = X_i^t + V_i^{(t+1)} \tag{2.4}$$

Algoritmo 4: Enjambre de Partículas

Entrada: Número de Partículas N

Número de Iteraciones T

Salida: Devuelve las mejores soluciones encontradas $GBest$

```
1  POP = CrearPoblacion(N)
2  mientras t < T hacer
3      para i = 1 a size(Pop) hacer
4          V_i^(t+1) = w * V_i^t + c1 * rand1 * (PBest_i - P_i^t) + c2 * rand2 * (GBest - P_i^t)
5          P_i^(t+1) = P_i^t + V_i^(t+1)
6          si Fitness(P_i) > Fitness(PBest_i) entonces
7              PBest_i ← P_i
8          si Fitness(P_i) > Fitness(GBest) entonces
9              GBest ← P_i
10 devolver GBest
```

2.4.2. Enjambre de Partículas Binario

El PSO expuesto anteriormente fue diseñado para la resolución de problemas de optimización con un espacio de búsqueda continuo. Para resolver problemas que involucren variables discretas, el cual es el caso de estudio del presente trabajo, fue propuesta la versión binaria de PSO por Kennedy y Eberhart (1997) [52]. En la versión binaria, una partícula se mueve en las esquinas de un hipercubo, cambiando los valores de los bits que determinan su posición. En este sentido, la velocidad de una partícula puede ser vista como el número de cambios ocurridos por iteración o la distancia de Hamming entre las posiciones de las partículas en el instante t y el $t + 1$. Una partícula con 0 bits cambiados, de una iteración a la otra, no se mueve mientras que otra partícula que cambia por el valor contrario todos sus bits, se desplaza a velocidad máxima.

Cada valor del vector de velocidad (v_{ij}) es utilizado como argumento de la función sigmoide (Ecuación 2.5) a fin de obtener el valor de P_{ij} el cual es determinado probabilísticamente dependiendo del valor v_{ij} que es equivalente a la probabilidad de que el ítem j aparezca en la regla P_i. Nótese, que ahora $GBest_{ij}$, $PBest_{ij}$ y P_{ij} toman los valores 0 o 1. El valor de la velocidad (v_{ij}) para la partícula i en la dimensión j se actualiza según la ecuación 2.3.

$$sig(v_{ij}^{(t)}) = \frac{1}{1 + e^{-v_{ij}^{(t)}}} \tag{2.5}$$

Luego, el vector posición de la partícula se actualiza de la siguiente forma:

Algoritmo 5: Actualización del vector de velocidad

```
si rand() < sig(v_ij^(t+1)) entonces
    P_ij^(t+1) = 1
en otro caso
    P_ij^(t+1) = 0
```

Donde rand() es un número random con distribución uniforme en [0,1] distinto para cada dimensión. El Algoritmo 6 detalla el pseudocódigo correspondiente.

El método PSO Binario previamente expuesto deja en evidencia la importancia que tiene una adecuada modificación del vector velocidad. El problema central de esta propuesta se encuentra en el escaso control que se realiza a la hora de acotar los valores de dicho vector.

En PSO binario, los números pequeños para el vector de velocidad estimulan la exploración. Es importante considerar que, si la velocidad de una partícula es el vector nulo, cada uno de los dígitos binarios que determina su posición tiene probabilidad 0.5 de cambiar a 1. Este es el caso más aleatorio que va a ocurrir. El caso contrario es cuando el vector velocidad toma valores grandes, por ejemplo, si $v_{ij} = 4$, entonces $sig(v_{ij}) = 0.982$, siendo 0.982 la probabilidad de que x_{ij} cambie al bit 1.

En este algoritmo la velocidad tiende a aumentar con el tiempo entonces todos los bits tienen mayor probabilidad de cambiar a 1. Este efecto en el minado de reglas de asociación tiene una incidencia negativa debido que más bits con el valor uno implica la presencia de mayor cantidad de *ítems* en la regla, por lo tanto, esto conlleva a una reducción de la métrica soporte, debido a que un mayor número de *ítems* deben aparecer conjuntamente en un mínimo de transacciones, situación que es bastante improbable. En la siguiente sección se describen los trabajos más recientes que utilizan el algoritmo PSO en el minado de reglas de asociación.

Algoritmo 6: Enjambre de Partículas Binario

Entrada: Número de Partículas N

Número de Iteraciones T

Salida: Mejores soluciones encontradas $GBest$

$POP = CrearPoblacion(N)$
mientras $t < T$ **hacer**
  para $i = 1$ **a** $size(Pop)$ **hacer**
    para $j = 1$ **a** $size(Particula)$ **hacer**
      $V_{ij}^{(t+1)} = w * V_{ij}^{t} + c_1 * rand_1 * (PBest_{ij} - P_{ij}^{(t)}) + c_2 * rand_2 * (GBest - P_{ij}^{(t)})$
      si $rand < \frac{1}{(1+e^{-V_{ij}^{(t+1)}})}$ **entonces**
        $P_{ij}^{(t+1)} = 1$ **en otro caso**
          $P_{ij}^{(t+1)} = 0$
    si $Fitness(P_i) > Fitness(PBest_i)$ **entonces**
      $PBest_i \leftarrow P_i$
    si $Fitness(P_i) > Fitness(GBest)$ **entonces**
      $GBest \leftarrow P_i$
devolver $GBest$

Capítulo 3

Estado del Arte

En la literatura, se han propuesto varios algoritmos basados en el algoritmo Enjambre de Partículas (PSO) para el minado de RA. Una de las primeras propuestas fue hecha por Osama et al. [53], en 2008, donde utilizan un algoritmo híbrido (PSO/ACO-AR), inspirado en los algoritmos de optimización de PSO y Colonia de Hormigas. El algoritmo híbrido propuesto no requiere las especificaciones de los umbrales de soporte y confianza, porque en cada iteración se seleccionan las mejores reglas generadas según los valores de la función de aptitud. La función de aptitud es la siguiente:

$$\begin{aligned} Fitness(X \Rightarrow Y) = (support(X \Rightarrow Y) - (marked * w) \\ -(amplitude * \psi) + (nAtr * \mu) \end{aligned} \tag{3.1}$$

Donde el soporte $(X \Rightarrow Y)$ representa el soporte de la regla, el parámetro marcado (marked) indica que un registro ha sido cubierto previamente por un *itemset*. Para penalizar los registros, se usa un valor llamado factor de penalización (w) para dar más o menos peso a los registros marcados. El parámetro de amplitud (amplitude) tiene como misión penalizar la amplitud de los intervalos que conforman el *itemset*. Por medio del factor ψ se logra que el algoritmo sea más o menos permisivo con respecto al crecimiento de los intervalos. El parámetro número de atributos (nAtr) recompensa los *itemset* frecuentes con un mayor número de *ítems*. Su efecto se incrementa o disminuye por medio del factor μ. Los experimentos fueron realizados sobre un conjunto de datos sintético conformado por 4 atributos y 1250 registros. Los resultados muestran que PSO/ACO-AR [53] es mejor que GAR [54] que se basa en algoritmos genéticos, en el conjunto de datos en que fue probado, en cuanto a la calidad de las reglas generadas y el tiempo de ejecución empleado a la hora de realizar esta tarea.

En [10], se introduce un algoritmo para el minado de RAs, que determina automáticamente los valores de los umbrales de soporte y confianza. El algoritmo incluye dos etapas, preprocesamiento y minería de RAs. En la primera etapa, los datos se transforman y almacenan en un formato binario. En la etapa de minería, el algoritmo PSO se usa para encontrar las RAs. En [10], cada partícula codifica una RA. En cada iteración del algoritmo, la aptitud de cada partícula se determina según su soporte, confianza y longitud (la longitud es el número de ítems presentes en la regla) (ver ecuación 3.2).

$$Fitness(X \Rightarrow Y) = Conf(X \Rightarrow Y) * Log(Supp(X \Rightarrow Y) \\ *Length(X \Rightarrow Y) + 1) \tag{3.2}$$

Después, se actualizan la velocidad y la posición de cada partícula. El procedimiento de búsqueda de PSO continúa hasta que se alcanza la condición de detención: cuando la población de partículas converge a la misma posición en el espacio de búsqueda o se alcanza el número especificado de iteraciones. Por último, después de encontrar la mejor partícula (regla), se recomiendan sus valores de soporte y confianza como el valor de los umbrales de soporte mínimo y confianza mínima para ese conjunto de datos, los cuales pueden ser empleados por otros algoritmos para el minado de RAs. La experimentación se realizó en un conjunto de datos conformado por 18 ítems y 1330 registros.

En [55], se propuso un framework para reducir el número de reglas generadas mediante la combinación del algoritmo PSO y técnicas de post-minería. Tanto el algoritmo propuesto [55], como en la propuesta anterior, los valores de soporte y confianza de la mejor regla se toman como el valor óptimo para los umbrales de soporte mínimo y confianza del conjunto de datos. En este algoritmo en particular, en la segunda etapa, las reglas generadas se reducen utilizando técnicas de post-minería, que filtran, las reglas generadas, mediante cuatro operadores: poda, conformidad, antecedente inesperado y consecuente inesperada, donde cada operador representa las expectativas del usuario en cuanto a la forma de las reglas generadas. La experimentación se realizó con el conjunto de datos Adult, extraído del repositorio *UCI Machine Learning Repository*, el cual está conformado por 97 ítems y 48842 registros.

En [11], se propone un algoritmo basado en PSO para extraer RAs de conjuntos de datos transaccionales. El algoritmo comprende dos partes, preprocesamiento y minería. En la primera parte, los datos se transforman y almacenan en un formato binario. En la segunda parte, el algoritmo PSO se usa para encontrar las reglas de asociación. En [11], los autores proponen usar el producto de las métricas soporte y confianza como la función de aptitud (Ecuación 3.3).

$$Fitness(X \Rightarrow Y) = Supp(X \Rightarrow Y) * Conf(X \Rightarrow Y) \tag{3.3}$$

Esta elección de la función de aptitud tiene el siguiente atractivo: cuando los valores de soporte y confianza se multiplican, los itemsets que tienen valores de soporte bajos, pero generan reglas interesantes no se eliminan. En cada iteración, solo se selecciona la mejor regla encontrada. Este proceso se repite M veces para generar las mejores M reglas que el algoritmo puede encontrar en el conjunto de datos, donde M es un parámetro del algoritmo. En la experimentación se utilizaron 2 conjuntos de datos, el primero de ellos contiene 2000 transacciones de ventas de 10 libros y el conjunto de datos restante contiene 20 productos y 1001 transacciones.

En [56], se propone un algoritmo híbrido entre el algoritmo Genético (AG) y PSO llamado GPSO para el minado de RAs. En esta propuesta es generada una población inicial de manera aleatoria de tamaño N, donde N es el número de individuos que la conforman, esta población es ordenada según el fitness de cada uno de los individuos y dividida en dos subpoblaciones, de tamaño $\frac{N}{2}$, la mitad superior contiene las reglas (individuos) con los valores de aptitud más altos estas reglas van a evolucionar usando AG, y la mitad inferior evoluciona con PSO. La calidad de cada regla se evalúa como en [10] y abarca las métricas: soporte, confianza y longitud. Las nuevas reglas generadas por ambos algoritmos se unen en una nueva población al final de cada generación y el proceso se repite por un número específico de iteraciones.

En la experimentación se utilizaron 5 conjuntos de datos extraídos del repositorio *UCI Machine Learning Repository*, el primero de ellos con nombre *Lenses* consta de 4 ítems y 24 transacciones, el segundo conjunto de datos *Car* está conformado por 6 ítems y 1728 transacciones, *Haberman's survival* contiene 3 ítems y 310 transacciones, *Post-operative patient care* contiene 8 ítems y 87 transacciones, por último *Zoo* contiene 16 ítems y 101 transacciones. El híbrido propuesto fue comparado con los algoritmos AG y PSO, las reglas de asociación generadas por el híbrido GPSO, en estos conjuntos de datos, tienen una mayor calidad en comparación con las reglas generadas por los algoritmos GA y PSO de manera independiente. En [57], los autores utilizaron esta metodología, pero también incluyeron en la función de aptitud otras medidas, como la comprensión y el interés. Otros trabajos como [12] han sido propuestos para formular el problema de la minería AR como un problema de optimización multiobjetivo, pero estos trabajos están fuera del alcance de este documento.

En [13], plantean como objetivo principal la generación de reglas de asociación eficientes y efectivas con la ayuda del algoritmo PSO. Para ello con el fin de elegir el valor de soporte adecuado y extraer las reglas de asociación interesantes, pero que son menos frecuentes en comparación con otros *itemsets* del conjunto de datos es diseñada la siguiente función de aptitud:

$$Fitness(X \Rightarrow Y) = \alpha_1 * \frac{Supp(X \Rightarrow Y)}{Supp(X)} * \frac{Supp(X \Rightarrow Y)}{Supp(Y)} * (1 - \frac{Supp(X \Rightarrow Y)}{D}) + \alpha_2 * \frac{NumberFields}{MaxFields} \tag{3.4}$$

En la ecuación 3.4 α_1 y α_2 son parámetros definidos por el usuario en el intervalo [0,1], D es el número total de registros en la base de datos, $\frac{Supp(X \Rightarrow Y)}{Supp(X)}$ indica la probabilidad de crear la regla basada en el antecedente, $\frac{Supp(X \Rightarrow Y)}{Supp(Y)}$ indica la probabilidad de crear la regla en función del consecuente, $(1 - \frac{Supp(X \Rightarrow Y)}{D})$ es la probabilidad de generar la regla de todo el conjunto de datos, *NumberFields* indica el número de atributos presentes en la regla y $maxfield$ indica la longitud máxima de la regla. Durante la experimentación se utilizaron 3 conjuntos de datos *BMSWebView1, FIFA, Chess* de ellos Chess es el que contiene mayor número de *ítems* y transacciones con 58 y 28056 respectivamente.

En [18] proponen un nuevo algoritmo basados en PSO para minar ARs, llamado PSO-DVAC, en este trabajo tratan de resolver la convergencia prematura controlando la velocidad de las partículas, para ello, se usa un método de control de coeficiente de aceleración adaptativo basado en la distancia. En cada iteración, cuando se actualiza el vector de velocidad de cada partícula, se calculan los valores de las variables de peso $c1_i$ y $c2_i$ de la siguiente manera:

$$c1_i = c_{max} - sin(\frac{\frac{\pi}{2}}{1 + d_i}) * (c_{max} - c_{min}) \tag{3.5}$$

$$c2_i = c_{min} + sin(\frac{\frac{\pi}{2}}{1 + d_i}) * (c_{max} - c_{min}) \tag{3.6}$$

$$d_i = (X_i - P_g)^2 \tag{3.7}$$

En la ecuación 3.5 y 3.6, $c1_i$ y $c2_i$ representan el coeficiente cognitivo y social de la partícula i-ésima, d_i es la distancia entre la posición de la i-ésima partícula y la mejor posición global. Los valores c_{max} y c_{min} representan el valor máximo y el valor mínimo del coeficiente social y el coeficiente cognitivo. De esta forma, al controlar la velocidad de la partícula pueden obtener el mismo resultado o un resultado más preciso en un número menor de iteraciones.

Para evaluar la calidad de las reglas generadas utilizan la siguiente función de aptitud:

$$Fitness(X \Rightarrow Y) = \alpha_1 * Supp(X \Rightarrow Y) + \alpha_2 * Conf(X \Rightarrow Y) \tag{3.8}$$

Donde, $\alpha_1 + \alpha_2 = 1$ y $\alpha_1 > 0$, $\alpha_2 > 0$. Los valores de α_1 y α_2 son ajustados en dependencia si se quiere preorizar los valores de la confianza o el soporte. El algoritmo devuelve al igual que [11] las M mejores reglas que pudo encontrar. El desempeño del algoritmo es evaluado utilizando los conjuntos de datos *FootMart2000* y *AdventureWorks*.

Los trabajos previamente mencionados funcionan mejor en cuanto al tiempo de ejecución empleado en el minados de reglas en comparación con los algoritmos exactos utilizados para extraer todo el conjunto de RAs [11, 55]. Sin embargo, tienden a converger prematuramente en soluciones (reglas) con valores bajos de aptitud, en conjuntos de datos que involucran un mayor número de *ítems* [18]. Por esta razón, en este trabajo, se propone un algoritmo basado en PSO para minar RAs que incluye una estrategia de exploración guiada y un método para la estimación rápida del soporte y la confianza de las reglas generadas, con el fin de garantizar la generación de reglas de alta calidad. En la siguiente sección se muestran cada una de las tareas específicas desarrolladas en cada una de las fases de la metodología.

Capítulo 4

Metodología

Como se mencionó en la Sección 2.1.1 para el desarrollo de la tesis se utilizó la metodología CRISP-DM, la cual consta de 6 fases:

4.1. Fase 1 Comprensión del negocio

En este trabajo se propone el diseño y desarrollo de una metaheurística para la identificación automática de reglas de asociación que permita predecir las posibles relaciones entre dos o más *ítems* en conjuntos de datos transaccionales, sin generar patrones frecuentes. Para ello es diseñada una metaheurística basada en PSO con una estrategia de búsqueda guiada con el objetivo de generar RAs de mayor calidad que otras propuestas existentes en el estado del arte en conjuntos de datos transaccionales de gran dimensión.

La metaheurística propuesta consta de 2 etapas: preprocesamiento y minería de RAs (Figura 4.1). En cada una de las fases de la metodología se detallan las tareas específicas desarrolladas en cada etapa del algoritmo.

4.2. Fase 2 Comprensión de los datos

Para validar la propuesta de solución, se usaron 30 conjuntos de datos. De ellos, 29 fueron tomados del repositorio de datos de la Universidad de Liverpool [1], el conjunto

[1] http://cgi.csc.liv.ac.uk/ frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html#datasets

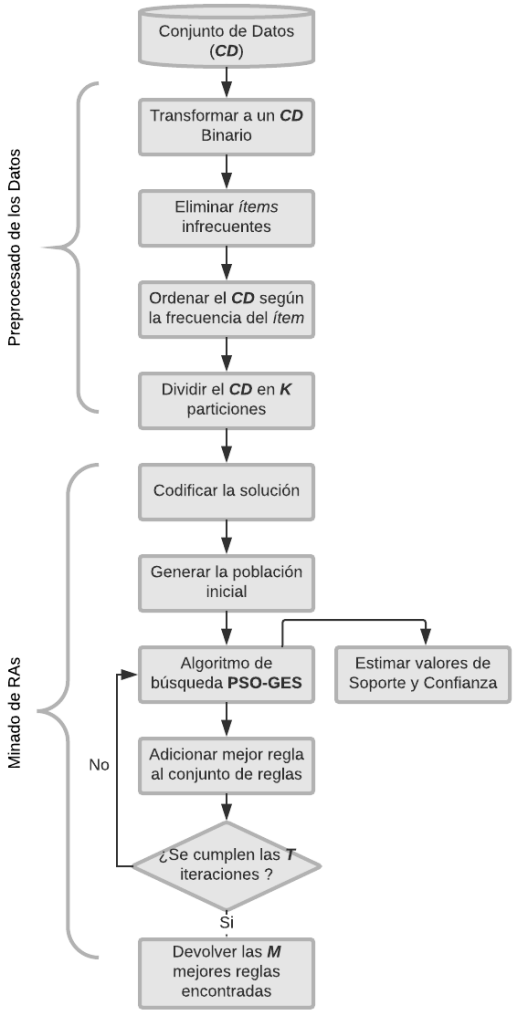

Figura 4.1: Metaheurística Propuesta

de datos restante llamado CharlesBook se tomó de la herramienta XLMINER [1], los conjuntos de datos seleccionados tienen la particularidad que ya tienen un preprocesado previo debido a que, los datos, ya están discretizados y normalizados. La Tabla 4.1 resume las características de los conjuntos de datos utilizados en los experimentos realizados, ordenados de acuerdo al número de transacciones.

Debido a que la mayoría de las propuestas reportadas en el estado del arte [11, 56, 13, 10, 53] presentan problemas al minar reglas de asociación utilizando PSO en conjuntos de datos que involucran un mayor número de *ítems*. Los conjuntos de datos utilizados en la experimentación (Tabla 4.1) fueron divididos en 2 grupos, los conjuntos con menos de 1000 transacciones y los conjuntos con más de 1000 transacciones. Con el objetivo de validar que la metaheurística propuesta, en grandes volúmenes de datos, genera reglas de asociación de mayor calidad que los algoritmos reportados en el estado del arte.

4.3. Fase 3 Preparación de los datos

El objetivo de esta fase es realizar el preprocesado de los datos, correspondiente a la primera etapa de la metaheurística propuesta (Figura 4.1), para ello:

- Los conjuntos de datos se transforman a conjuntos de datos binarios con el objetivo de acelerar el cálculo de las métricas soporte y confianza (Sección 4.3.1).
- Se eliminan aquellos *ítems* que individualmente no cumplen con el umbral de mínimo soporte.
- Las columnas en el conjunto de datos se ordenan de acuerdo con la frecuencia de los *ítems* (cada columna representa un *ítem*).
- Finalmente, los conjuntos de datos se dividen en K particiones (K es un parámetro), la frecuencia de cada *ítem* en cada partición se calcula y almacena en otra tabla resumen, de esta forma, el número de las filas en la tabla resumen es menor que el número de filas en el conjunto de datos original (Sección 4.3.2).

4.3.1. Representación de los Datos

Un tema importante para calcular el soporte de los *itemsets* es la forma en que se representan los datos (Sección 2.3.1). En esta propuesta, el conjunto de datos transaccional

[1] www.solver.com/xlminer-data-mining

Tabla 4.1: Conjuntos de Datos

Grupos	***CD***	***Columnas***	***Filas***
Menos de 1000 transacciones	Zoo	18	101
	Iris	19	150
	Hepatitis	20	155
	Wine	68	178
	Auto	137	205
	Glass	48	214
	Heart	52	303
	Ecoli	9	336
	Ionosphere	157	351
	Dermatology	49	366
	Horsecolic	85	368
	CylBands	124	540
	Soybean	118	683
	Breast	11	699
	Pima	38	768
	Anneal	73	898
	Tictactoe	29	958
Más de 1000 transacciones	Flare	39	1389
	CharlesBook	11	2000
	Led	24	3200
	Waveform	101	5000
	PageBlocks	46	5473
	Mushroom	90	8124
	PenDigits	89	10992
	Nursery	32	12960
	Letrecog	106	20000
	Chess	58	28056
	Adult	97	48842
	Connect	129	67557
	Accidents	468	340184

(DS) se representa como un DS binario a través de vectores horizontales donde 1 en la posición $i-th$ denota la presencia del *ítem* $i-th$ en la transacción, mientras que 0 denota la ausencia del *ítem*. Como ejemplo, en la Tabla 4.3 aparece la representación horizontal del conjunto de datos transaccional que se muestra en Tabla 4.2.

Tabla 4.2: Transactional DS

T_1	a,b,d,e,f
T_2	b,c,e
T_3	a,b,d,e
T_4	a,b,c,e
T_5	a,b,c,d,e,f
T_6	b,c,d

Tabla 4.3: Binary DS

	a	***b***	***c***	***d***	***e***	***f***
T_1	1	1	0	1	1	1
T_2	0	1	1	0	1	0
T_3	1	1	0	1	1	0
T_4	1	1	1	0	1	0
T_5	1	1	1	1	1	1
T_6	0	1	1	1	0	0

Como un *ítem* no frecuente no puede formar parte de un *itemset* frecuente (Propiedad Clausura Descendiente de Soporte, Sección 2.3.2), todos los *ítems* que no cumplen el umbral de mínimo soporte especificado no se almacenan en el DS binario.

4.3.2. Partición del Conjunto de Datos

Para permitir estimar rápidamente el soporte de un *ítem* sin escanear todo el conjunto de datos, con base en la propuesta de [58] se propone construir una tabla resumen que se usa para estimar fácilmente la frecuencia y soporte de los *itemsets*.

Primero, el conjunto de datos D se divide en K particiones de aproximadamente el mismo tamaño $(D_1, ..., D_K; K \geq 1)$, las transacciones de conjunto de datos se asignan a cada partición de la tabla resumen al azar, y la frecuencia de cada *ítem* en cada partición se calcula y almacena en la base de datos resumen.

La Tabla 4.4 muestra un ejemplo de la representación binaria de un conjunto de datos D con cinco *ítems* (a, b, c, d, e) y seis transacciones $(T_1, T_2, T_3, T_4, T_5, T_6)$. Si se denota por $\lambda(a, D)$ la frecuencia de a en D, se tiene $\lambda(a, D) = 4$; $\lambda(b, D) = 6$; $\lambda(c, D) = 4$; $\lambda(d, D) = 4$; $\lambda(e, D) = 5$. Si $K = 3$, D se divide en $D_1 = \{T_1, T_2\}$; $D_2 = \{T_3, T_4\}$ y $D_3 = \{T_5, T_6\}$ (ver Tabla 4.4). Si se calcula la frecuencia de los *ítems* a, b y e en el conjunto de datos particionado, los resultados son los siguientes:

$$(\lambda(a, D1)); (\lambda(a, D2)); (\lambda(a, D3)) = (1; 2; 1) \tag{4.1}$$

$$(\lambda(b, D1)); (\lambda(b, D2)); (\lambda(b, D3)) = (2; 2; 2) \tag{4.2}$$

$$(\lambda(e, D1)); (\lambda(e, D2)); (\lambda(e, D3)) = (2; 2; 1) \tag{4.3}$$

Tabla 4.4: DS con 5 *ítems* y 6 transacciones dividido en 3 particiones

	a	**b**	**c**	**d**	**e**	
T_1	1	1	0	1	1	D_1
T_2	0	1	1	0	1	
T_3	1	1	0	1	1	D_2
T_4	1	1	1	0	1	
T_5	1	1	1	1	1	D_3
T_6	0	1	1	1	0	

Tabla 4.5: DS Resumen

	b	***e***	***a***	***c***	***d***
D_1	2	2	1	1	1
D_2	2	2	2	1	1
D_3	2	1	1	2	2
D	6	5	4	4	4

La Tabla 4.5 muestra la tabla resumen construida a partir del conjunto de datos D de la Tabla 4.4 para $K = 3$. En esta tabla, cada fila está asociada a una partición (D_1, D_2, D_3) que contiene la frecuencia de cada *ítem* en la partición D_i. La última fila contiene la frecuencia total de los *ítem* en todo el conjunto de datos D.

4.3.2.1. Cálculo de la frecuencia estimada

Una vez que se ha obtenido la nueva estructura, de acuerdo con las ideas de [25], se propone estimar $\lambda(S, D)$ como la suma de la frecuencia mínima entre los *ítems* del *itemset* $S = a_1, a_2, ..., a_{|S|}$ en cada partición (ver ecuación 4.4):

$$\lambda(S, D) \approx \sum_{i=1}^{K} min(\lambda(a_1, D_i), \lambda(a_2, D_i), ..., \lambda(a_{|S|}, D_i)) \tag{4.4}$$

Se selecciona el valor de frecuencia mínima entre los *ítems* que conforman el *itemset* $S = a_1, a_2, ..., a_{|S|}$, porque si $\lambda(a_1, D_i) < \lambda(a_2, D_i) < \lambda(a_{|S|}, D_i)$ los *ítems* del *itemset*

S aparecen juntos en la partición D_i máximo $\lambda(a_1, D_i)$ veces. Por ejemplo, usando los datos en la Tabla 4.5:

$$\lambda(\{d,e\},D) \approx \sum_{i=1}^{3} min\{\lambda(d,D_i),\lambda(e,D_i)\} = min\{1,2\}+min\{1,2\}+min\{2,1\} = 3 \tag{4.5}$$

$$\lambda(\{b,d,e\},D) \approx \sum_{i=1}^{3} min\{\lambda(b,D_i), min\{\lambda(d,D_i),\lambda(e,D_i)\} = min\{2,1,2\}+min\{2,1,2\} + min\{2,2,1\} = 3 \tag{4.6}$$

4.3.2.2. Cálculo del soporte estimado

Finalmente, el soporte estimado de un *itemset* S en el conjunto de datos D se calcula dividiendo la frecuencia estimada $\lambda(\{S\},D)$ entre la cantidad total de transacciones T en el conjunto de datos D, como en la ecuación 4.7.

$$ESupp(S,D) \approx \frac{\lambda(S,D)}{T} \tag{4.7}$$

Por ejemplo, usando los datos de la Tabla 4.5, el soporte estimado del *itemset* $\{d,e\}$ es:

$$ESupp(\{d,e\},D) \approx \frac{\lambda(\{d,e\},D)}{T} \approx \frac{3}{6} \approx 0.5 \tag{4.8}$$

Nótese que el valor de soporte estimado va a ser generalmente superior o igual, en el mejor de los casos, al valor de soporte real. Además, cuando el número de particiones (K) es mayor, el mayor valor que puede tomar el parámetro K coincide con el número de transacciones en el conjunto de datos, la estimación del soporte es mejor. Sin embargo, el tiempo de ejecución se incrementa.

4.4. Fase 4 Modelado

En esta fase es donde se lleva a cabo el minado de las reglas de asociación y resume la segunda etapa de la metaheurística propuesta (Figura 4.1), para ello:

- El enjambre de partículas se codifica (Sección 4.4.1).
- Se genera la población inicial de manera aleatoria y se inicia el proceso de evolución.
- Se utiliza un proceso de evolución guiada que garantiza que en cada iteración del algoritmo cada partícula (RA) cumpla con los umbrales de confianza y soporte establecidos. Para esto, se utilizan los valores estimados de soporte de los *itemsets* en las particiones (Sección 4.3.2.1 y 4.3.2.2), para determinar rápidamente que *ítems* se deben incluir en la regla con el fin de tener una influencia positiva en el *fitness* de ésta (Sección 4.4.1.3).
- Finalmente, la metaheurística propuesta devuelve las mejores M reglas encontradas durante el proceso de búsqueda; M es un parámetro del algoritmo.

4.4.1. Representación de las Reglas de Asociación

En la literatura, las reglas de asociación pueden ser representadas de 2 formas. Un tipo de representación es el enfoque de Pittsburgh [59], donde cada partícula representa un conjunto de reglas y la otra representación es el enfoque de Michigan [59], donde cada partícula representa una sola regla. En este trabajo, para mayor claridad, se utiliza el enfoque de Michigan.

Para representar una regla, se usa la codificación binaria [60], en esta codificación una regla es un vector P de tamaño n donde n es el número de *ítems* presentes en el conjunto de datos, cada *ítem* puede tomar 3 posibles valores 11, 10 y 00; en este vector, si:

1. $P[i] = 11$, entonces el *ítem* i está incluido en el antecedente de la regla.
2. $P[i] = 10$, entonces el *ítem* i está incluido en el consecuente de la regla.
3. $P[i] = 00$, entonces el *ítem* i no está incluido en la regla.

Ejemplo: Sea $T_1 = \{a, b, d, e\}$ la primera transacción de la Tabla 4.4, si se desean representar las reglas $P_1 = \{b, e\} \Rightarrow \{d\}$ y $P_2 = \{a, e\} \Rightarrow \{c\}$, según la representación expuesta quedarían de la siguiente manera: $P_1 = 00|11|00|10|11$ y $P_2 = 11|00|10|00|11$.

4.4.1.1. Función *Fitness*

Para validar la calidad de las reglas generadas, como función *fitness*, en este trabajo, se utiliza una de las funciones *fitness* más utilizadas [[61], [56], [62], [10], [55]] (ver ecuación 4.11), que involucra las métricas soporte, confianza y Longitud.

$$Fitness(X \Rightarrow Y) = Conf(X \Rightarrow Y) * Log(Supp(X \Rightarrow Y) \\ *Length(X \Rightarrow Y) + 1) \tag{4.9}$$

Donde el soporte representa la fracción de transacciones en el conjunto de datos que contiene los *ítems* de la regla; la confianza representa la probabilidad de encontrar el consecuente de una regla en una transacción que contiene el antecedente; y la longitud es la cantidad de *ítems* en la regla. En esta ecuación se multiplican los valores de soporte y longitud para descartar aquellas reglas que tienen soporte igual a cero. Además, es sumado 1 a la multiplicación de los valores soporte y confianza para evitar el logaritmo de cero el cual es un valor indefinido y es calculado el logaritmo base 10 de la multiplicación del soporte y la confianza más uno para darle un peso similar a esta multiplicación que al valor de la confianza.

Sin embargo, en este trabajo, no se utilizan los valores reales de soporte y confianza, sino el soporte estimado (ecuación 4.7) y la confianza estimada que se calcula de la siguiente forma:

$$EConf(X \Rightarrow Y) \approx \frac{ESupp(X \cup Y)}{ESupp(X)} \tag{4.10}$$

Siendo $supp(X \cup Y)$ el soporte estimado del *itemset* $\{X, Y\}$ en el conjunto de datos D y $supp(X)$ el soporte estimado del *ítem* X en el conjunto de datos D. Reescribiendo la ecuación 4.11, respetando la nomenclatura definida para el soporte y la confianza estimada quedaría representada de la siguiente forma:

$$Fitness(X \Rightarrow Y) = EConf(X \Rightarrow Y) * Log(ESupp(X \Rightarrow Y) \\ *Length(X \Rightarrow Y) + 1) \tag{4.11}$$

El objetivo de esta función *fitness* (Ecuación 4.11) es permitir que los *ítems* que aparecen con menos frecuencia pero que generan reglas interesantes y potencialmente valiosas no se eliminen y se prioricen aquellas reglas que tienen un mayor número de *ítems*.

4.4.1.2. Metaheurística Propuesta (PSO-GES)

El algoritmo para el minado de RAs propuesto (PSO-GES, Algoritmo 7), tiene como parámetros de entrada el número de partículas (N) que evolucionarán durante varias iteraciones (T), además requiere el número de (M) reglas que se desean encontrar, así como los umbrales de soporte ($MinSup$) y confianza ($MinCof$), el conjunto de datos (D) a minar y el número de particiones (K) en que se dividirá el conjunto de datos. Además, de los parámetros específicos del algoritmo PSO: el factor de inercial (w) y las constantes de aceleración c_1 y c_2.

En PSO-GES, cada partícula está compuesta por 3 vectores y 2 valores de *fitness*, el primer vector $P_i = (a_1, a_2, ..., a_n)$, almacena la posición de la partícula actual (regla) en el espacio de búsqueda, el vector $PBest_i = (a_1, a_2, ..., a_n)$ almacena la mejor partícula (mejor regla) encontrada por la partícula P_i y $V_i = (v_1, v_2, ..., v_n)$ es un vector que almacena la dirección y la velocidad en que se moverá la partícula P_i en el espacio de búsqueda. El valor *Fitness* (P_i) almacena el valor de aptitud de la partícula actual (regla), mientras que *fitness* $PBest_i$ almacena el valor de la aptitud de la mejor regla encontrada por la partícula P_i.

Antes de comenzar la evolución, PSO-GES primero genera la Tabla Resumen (Algoritmo 7, línea 1) con un número de filas igual a K (Sección 4.3.2) a partir del conjunto de datos D. Luego es generada aleatoriamente la población inicial de tamaño N (Algoritmo 7, línea 2). Durante un número de iteraciones T (Algoritmo 7, línea 3), para cada partícula P_i^t de la población de tamaño N (Algoritmo 7, línea 4), se actualiza su posición P_i^{t+1} (Algoritmo 7, línea 7) y seguidamente se actualiza su *fitness*. Si el nuevo *fitness* es el mejor *fitness* encontrado hasta el momento por la partícula P_i (Algoritmo 7, línea 8), tanto $PBest_i$ como su *fitness* (*fitness* $PBest_i$) se actualizan (Algoritmo 7, línea 9). De igual manera, si el nuevo *fitness* es la mejor solución global encontrada hasta el momento por la partícula P_i (Algoritmo 7, línea 10), tanto $GBest_i$ como su *fitness* (*fitness* $GBest_i$) se actualizan (Algoritmo 7, línea 11).

En cada iteración, cada componente $v_{ij}(j = 1, .., n)$ del vector de velocidad para la partícula P_i se actualiza teniendo en cuenta su experiencia y su entorno (Algoritmo 7, línea 6) mediante la siguiente expresión:

$$v_{ij}^{(t+1)} = w * v_{ij}^t + c_1 * rand_1 * (PBest_{ij}^t - P_{ij}^t) + c_2 * rand_2 * (GBest_{ij}^t - P_{ij}^t) \quad (4.12)$$

Donde:

- w, es el factor de inercia y representa el grado de similitud entre la nueva velocidad y la velocidad actual.

Algoritmo 7: Enjambre de partículas utilizando una estrategia de exploración guiada (PSO-GES)

Entrada: Número de partículas N

Número de iteraciones T

Número de RAs que se desean encontrar M

Dataset D

Número de particiones K

Mínimo Soporte $MinSup$

Mínima Confianza $MinConf$

Factor de Inercia w

Constantes de aceleración c_1 and c_2

Salida: Devuelve las M mejores reglas de asociación encontradas $GBests$

$Table \leftarrow CrearTabla(D, K)$

$POP \leftarrow CrearPoblacion(N)$

mientras $t < T$ **hacer**

  para $i \leftarrow 0$ **a** $size(POP)$ **hacer**

    para $j \leftarrow 0$ **a** $size(P_i^t)$ **hacer**

      $v_{ij}^{(t+1)} \leftarrow w * v_{ij}^t + c_1 * rand_1 * (PBest_{ij}^t - P_{ij}^t) + c_2 * rand_2 * (GBest_{ij}^t - P_{ij}^t)$

      $P_{ij}^{(t+1)} \leftarrow ActualizarPosicion(P_i^{(t+1)}, j, v_{ij}^{(t+1)}, Table, MinSup, MinConf)$

    si $Fitness(P_i^{(t+1)}) > Fitness(PBest_i)$ **entonces**

      $PBest_i \leftarrow P_i^{(t+1)}$

    si $Fitness(P_i^{(t+1)}) > Fitness(GBest)$ **entonces**

      $GBest \leftarrow P_i^{(t+1)}$

    $GBests \leftarrow BestRules(P_i^{(t+1)})$

devolver $GBests$

- c_1 y c_2, son 2 constantes de peso utilizadas para dar mayor o menor peso a la mejor posición local o a la mejor posición global, estos parámetros son definidos por el usuario y pueden tomar valores en el intervalo [0,2].
- $rand_1$ y $rand_2$, son valores aleatorios generados en el intervalo [0,1].
- $PBest$, representa la mejor solución (mejor regla) encontrada por la partícula P_i.
- $GBest$, representa la mejor solución global (mejor regla encontrada).
- v_{ij}, se usa como argumento para la función sigmoide (Ecuación 2.5) para estimar la probabilidad de que el *ítem* j aparezca en la regla P_i.

En la Ecuación 4.12, cuando los valores de P_{ij}, $PBest_{ij}$ y $GBest_{ij}$ son iguales, indica que el valor v_{ij} del vector de velocidad solo se verá afectado por la variable w. Sin embargo, si P_{ij} es igual a 1 y los valores de $GBest_{ij}$ y $Best_{ij}$ son iguales a 0, el valor de v_{ij} se reduce y, por lo tanto, disminuye la probabilidad de que el ítem j aparezca en la regla. El caso opuesto es cuando $GBest_{ij}$ y $PBest_{ij}$ son iguales a 1 y P_{ij} es igual a 0; en este caso, el valor de v_{ij} se incrementa y, por lo tanto, la probabilidad de que el ítem j aparezca en la regla aumenta.

PSO-GES actualiza la partícula P_i desde el inicio del *ítem* 1 hasta el *ítem* n o hasta que hayan sido agregados a la regla asociada a la partícula P_i un número de *ítems* igual al mayor número de *ítems* existentes en una transacción. El *ítem* j de la partícula $P_i^{(t+1)}$ ($P_{ij}^{(t+1)}$) se actualiza (Algoritmo 8) generando un número aleatorio con distribución uniforme en [0,1] diferente para cada dimensión (j) pero verificando si este número es menor que el valor de la función sigmoidal (Ecuación 2.5) aplicada a la velocidad de la partícula en la dimensión correspondiente (Algoritmo 8, línea 1), si es así, el *ítem* se actualiza (Sección 4.4.1.3); de lo contrario, se excluye el *ítem* correspondiente (Algoritmo 8, línea 4).

Nótese que, aunque la actualización de la posición de la partícula solo tiene en cuenta el vector de velocidad, el vector de posición actual se tiene en cuenta indirectamente ya que la posición actual se usa para generar el vector de velocidad.

4.4.1.3. Método de búsqueda guiada

En esta propuesta, a diferencia de otras propuestas informadas en la literatura [[53], [12], [56], [10], [55], [11]], para actualizar la partícula P_i, se propone un método de búsqueda guiada que estimula que solo aquellos *ítem* que juntos son frecuentes se incorporan a la regla. Para ello, además de tener en cuenta los valores del vector de velocidad (Algoritmo 8), cada vez que se actualiza un *ítem* en la nueva posición de la partícula,

Algoritmo 8: Actualizar Posición

Entrada: Partícula P_i^t,

Ítem a actualizar j

Velocidad de la partícula $V_{ij}^{(t+1)}$

Dataset Resumen $Table$

Soporte Mínimo $MinSup$

Confianza Mínima $MinConf$

Salida: Devuelve la partícula actualizada $P_i^{(t+1)}$

1 **si** $rand() < sig(V_{ij}^{(t+1)})$ **entonces**

2 $\quad P_{ij}^{(t+1)} \leftarrow CalcularSoporteEst(P_i^{t+1}, j, Table, MinSup)$

3 **en otro caso**

4 $\quad P_{ij}^{(t+1)} \leftarrow 00$

5 **devolver** $P_{ij}^{(t+1)}$

se calcula el soporte estimado y la confianza estimada; evitando la inserción de *ítems* que influyen negativamente en la aptitud de la regla asociada a la nueva posición de la partícula, de esta manera, se promueve el minado de RAs con altos valores de *fitness*.

El Algoritmo 9 ilustra el proceso de actualización resumido anteriormente. En este algoritmo se consideran tres casos para actualizar el valor del *ítem* j de la partícula $P_i^{(t+1)}$ cuando $rand() < sig(v_{ij}^{(t+1)})$:

- Si no se ha incluido ningún *ítem* a la regla ($size(P_i^{(t+1)})$), incluya el *ítem* j en el antecedente de la regla ($P_{ij}^{(t+1)} = 11$) o el consecuente ($P_{ij}^{(t+1)} = 10$) de la regla de forma aleatoria (Algoritmo 9, línea 10).
- Si se ha incluido al menos un *ítem* en la regla y el soporte estimado del conjunto de *ítem* ya incluidos en la regla más el *ítem* j no es mayor que el umbral de mínimo de soporte(Algoritmo 9, línea 2), entonces el *ítem* j no se incluye en la regla ($P_{ij}^{(t+1)} = 00$) (Algoritmo 9, línea 8).
- Si se ha incluido al menos un *ítem* en la regla (Algoritmo 9, línea 1) y el soporte estimado del conjunto de *ítems* ya incluidos en la regla más el *ítem* j es mayor que el umbral de mínimo soporte (Algoritmo 9, línea 2), pero el antecedente o el

consecuente de la regla está vacío (Algoritmo 9, línea 3), incluya el *ítem* j en el antecedente o en el consecuente de la regla de forma aleatoria (Algoritmo 9, línea 6). De lo contrario, la confianza estimada se calcula para ambos, incluir el *ítem* j en el antecedente e incluir el ítem j en el consecuente de la regla, y el *ítem* j se actualiza con el valor (11 o 10) que arroja el mejor resultado (Algoritmo 9, línea 4).

Algoritmo 9: Calcular Soporte Estimado

Entrada: Partícula P_i^t,

Ítem a actualizar j

Dataset Resumen $Table$

Soporte Mínimo $MinSup$

Salida: Devuelve la partícula P_i actualizada $P_i^{(t+1)}$

```
1  si size(P_i^(t+1)) ≠ ∅ entonces
2      si SupEstimate(P_i^(t+1), j, Table) > MinSup entonces
3          si Contain(P_i^(t+1)) entonces
4              P_ij^(t+1) ← CalcularConfEst(P_i^t+1, j, Table, MinConf)
5          en otro caso
6              P_ij^(t+1) ← GenerarPosicionAleatoria()
7      en otro caso
8          P_ij^(t+1) ← 00
9  en otro caso
10     P_ij^(t+1) ← GenerarPosicionAleatoria()
11 devolver P_ij^(t+1)
```

La complejidad de nuestra propuesta depende del número de partículas (N), el número máximo de *ítems* en una transacción (I) y el número de particiones (K). Para explorar una nueva posición, la evaluación de cada solución requiere como máximo de $NxIxK$ operaciones. Por lo tanto, la complejidad computacional de PSO-GES en el peor de los casos es $O(NxIxK)$. El algoritmo 7 muestra el algoritmo para el minado de RAs propuesto.

4.5. Fase 5 Evaluación

Para validar el desempeño del algoritmo diseñado, éste fue comparado con las propuestas de los autores Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18], los cuales utilizan el algoritmo PSO para el minado de reglas. Los algoritmos son comparados en cuanto a calidad de las reglas generadas y tiempo de ejecución empleado. En la siguiente sección se detallan cada uno de los experimentos realizados.

Capítulo 5

Resultados

Para validar el rendimiento de PSO-GES, primeramente, es analizado el impacto del número de particiones (K) del conjunto de datos, este es un parámetro específico de la metaheurística propuesta, en el tiempo de ejecución y la calidad de las reglas generadas. En un segundo experimento, se comparó PSO-GES con tres de los trabajos más recientes en la literatura que utilizan PSO (formulados como problemas de optimización mono-objetivo) para extraer RAs, los algoritmos reportados por Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18]. Los algoritmos se compararon en términos de la calidad de las reglas generadas y el tiempo de ejecución requerido para encontrar las reglas. En un último experimento, se evaluó el desempeño de PSO-GES en términos del número de *ítems* existentes en el conjunto de datos.

5.1. Ambiente de pruebas

Todos los experimentos se realizaron en el entorno Microsoft Windows 8.1 utilizando una computadora de escritorio con un procesador Intel core-i7 con 4 núcleos físicos de 3.4 GHz y 8 GB de RAM. El algoritmo propuesto (PSO-GES) y el resto de los algoritmos con los que fue comparado, se codificaron en $C\#$ usando el entorno de desarrollo Visual Studio 2015. Para los experimentos, se usaron 30 conjuntos de datos expuestos en la Sección 4.2.

Para todos los experimentos, los parámetros comunes entre los algoritmos se definieron teniendo en cuenta los valores utilizados por Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18]; de la siguiente manera:

- La población se definió de 20 partículas.

- Las constantes c_1 y c_2 se establecieron en 2, por ser este valor el más utilizado para estas variables en el estado del arte.
- La inercia w tomó el valor de 1
- El número de iteraciones se definió igual a 10

PSO-GES requiere además los siguientes parámetros:

- Umbral de mínimo soporte igual a 0.2 (Definido durante la experimentación, Tabla 5.1).
- Umbral de mínima confianza igual a 0.9.
- Parámetro K (número de particiones) igual a 20 (Definido durante la experimentación, Sección 5.2).

La confianza mínima se definió con un valor igual a 0.9, porque es el valor más utilizado en la literatura para este parámetro, para definir el valor del umbral de soporte mínimo se experimentó con los siguientes conjuntos de valores 0.1, 0.2, 0.3, 0.5. Solo para los valores de soporte igual a 0.1 y 0.2, el 100 % de los conjuntos de datos generaron reglas válidas (Véase Tabla 5.1). Por esta razón, se estableció el umbral de soporte mínimo en 0.2 para el resto de los experimentos.

5.2. Influencia del número de particiones en el algoritmo PSO-GES

Una vez que se han definido los parámetros del algoritmo, el primer experimento consistió en el estudio del impacto del parámetro K (número de particiones del conjunto de datos), en el tiempo de ejecución y la calidad de las reglas generadas, que se utiliza para construir el conjunto de datos resumido en la primera etapa de PSO-GES.

5.2.1. Influencia del número de particiones en el tiempo de ejecución

Para este experimento, se ejecutó el algoritmo fijando el resto de los valores de los parámetros y variando el parámetro de K sobre los valores OTPP [1], 100, 20, 10, 5 y 3.

Tabla 5.1: Evaluación de la calidad de las reglas generadas para un valor de soporte igual a $Supp = \{0.1, 0.2, 0.3, 0.5\}$

Grupos	DS	0.1	0.2	0.3	0.5
Menos de 1000 transacciones	**Zoo**	0.597	0.600	0.598	0.639
	Iris	0.251	0.270	0	0
	hepatitis	0.619	0.660	0.63	0.648
	Wine	0.231	0.240	0.242	0
	Auto	0.573	0.550	0.596	0.679
	Glass	0.327	0.360	0.394	0.432
	Heart	0.764	0.730	0.622	0.689
	Ecoli	0.593	0.620	0.631	0.66
	Ionosphere	0.668	0.619	0.610	0.647
	Dematology	0.564	0.650	0.580	0.680
	Horsecolic	0.484	0.490	0.528	0.609
	CylBands	0.673	0.660	0.697	0.726
	Soybean	0.649	0.690	0.673	0.678
	Breast	0.822	0.992	0.772	0.864
	Pima	0.755	0.790	0.758	0.755
	Anneal	0.65	0.64	0.673	0.706
	Tictactoe	0.074	0.072	0	0
	Flare	0.656	0.656	0.656	0.656
Más de 1000 transacciones	**CharlesBook**	0.119	0.150	0	0
	Led	0.405	0.420	0.419	0.431
	Waveform	0.335	0.320	0.349	0.423
	PageBlocks	0.978	0.980	0.978	0.981
	Mushroom	0.704	0.710	0.698	0.785
	PenDigits	0.122	0.130	0	0
	Nursery	0.392	0.378	0	0
	LetR	0.467	0.440	0.468	0.486
	Chess	0.043	0.040	0	0
	Adult	0.6566	0.800	0.6842	0.789
	Connect	0.749	0.606	0.760	0.761
	Accident	0.875	0.750	0.703	0.882

Tabla 5.2: Tiempo en seg. de PSO-GES para $K = \{OTPP, 100, 20, 10, 5, 3\}$

Grupos	CD	OTPP	100	20	10	5	3
Menos de 1000 transacciones	**Zoo**	1.429	1.804	0.851	0.829	**0.721**	**0.721**
	Iris	0.978	0.714	0.704	**0.620**	0.795	0.795
	hepatitis	1.789	1.164	**1.043**	1.045	1.152	1.073
	Wine	1.806	1.383	0.963	**0.541**	0.894	0.894
	Auto	3.508	2.576	2.174	**1.479**	2.530	2.751
	Glass	1.380	1.062	0.853	**0.792**	1.095	1.087
	Heart	2.201	1.429	**0.970**	1.394	1.405	1.505
	Ecoli	1.523	1.173	0.813	**0.742**	1.073	0.912
	Ionosphere	6.529	4.351	2.597	2.945	**2.353**	2.435
	Dematology	2.210	1.438	0.910	**0.893**	1.619	1.033
	Horsecolic	3.369	2.038	**1.220**	1.581	1.733	1.996
	CylBands	10.658	7.672	4.214	4.390	**2.774**	3.371
	Soybean	12.699	8.436	6.457	5.772	2.980	**2.235**
	Breast	2.842	1.898	1.454	1.414	**0.862**	1.366
	Pima	3.933	1.484	1.156	**1.064**	1.098	1.085
	Anneal	8.222	3.758	2.608	3.428	**1.806**	2.335
	Tictactoe	3.538	1.883	1.869	1.857	1.783	**1.547**
Más de 1000 transacciones	**Flare**	6.833	2.005	2.751	1.528	**1.325**	1.632
	CharlesBook	2.809	1.115	0.886	0.888	**0.862**	0.889
	Led	7.409	2.480	**1.805**	2.432	2.640	2.597
	Waveform	56.188	19.467	17.177	17.552	17.539	**9.018**
	PageBlocks	47.525	7.843	7.729	7.592	7.584	**7.496**
	Mushroom	111.638	28.874	27.322	26.685	**14.902**	25.263
	PenDigits	124.383	25.597	24.776	24.608	**20.392**	24.451
	Nursery	76.426	14.680	14.255	14.254	**13.212**	14.305
	LetR	256.555	30.003	44.570	**29.441**	47.294	40.716
	Chess	140.473	24.663	24.715	**24.033**	25.069	24.456
	Adult	1130.482	210.958	208.808	209.042	146.196	**115.965**
	Connect	4148.446	857.706	**695.165**	**695.165**	857.706	776.726
	Accident	7373.234	4531.652	4359.498	**4136.862**	4624.105	4525.620
Suma		0	0	5	11	10	6

La Tabla 5.2 muestra el tiempo de ejecución alcanzado por PSO-GES en cada uno de los conjuntos de datos mencionados anteriormente, para los diferentes valores de K.

La Tabla 5.2 muestra el tiempo de ejecución alcanzado por PSO-GES para $K = \{OTPP, 100, 20, 10, 5, 3\}$, en esta tabla los valores en negritas representan los mejores tiempos de ejecución obtenidos, como se puede apreciar en los conjuntos de datos del primer grupo (con cientos de transacciones) el tiempo de ejecución alcanzado cuando los conjuntos de datos están particionados con $OTPP$ es similar al tiempo de ejecución alcanzado cuando K es igual a 100, porque la reducción en el número de filas en estos conjuntos de datos no es relevante. Por otro lado, en los conjuntos de datos del segundo grupo, se puede notar que al comparar el tiempo de ejecución de PSO-GES en los conjuntos de datos particionados con una transacción por partición (OTPP) con respecto a los tiempos obtenidos en los conjuntos de datos particionados con $K = \{100, 20, 10, 5, 3\}$, el tiempo de ejecución se reduce a más de la mitad, esto se debe a que al reducir drásticamente el número de transacciones, el costo de estimar los valores de soporte y confianza disminuyen. Como se puede apreciar al comparar los tiempos obtenidos entre los valores de $K = \{100, 20, 10, 5, 3\}$ no hay una diferencia significativa en los tiempos de ejecución alcanzados.

5.2.2. Influencia del número de particiones en la calidad de las reglas generadas

Otro punto importante a evaluar es la calidad de las reglas generadas con respecto al valor del parámetro K. En la Tabla 5.3, se puede apreciar que cuanto menor sea el número de particiones (valores bajos para K), menores serán los valores de aptitud de las RAs generadas. Esto se debe a que cuanto mayor sea el número de *ítems* por partición, peor es la estimación del soporte y la confianza. En este sentido, cuando el conjunto de datos se particiona con valores de $K = \{100, 20\}$, la calidad de las reglas generadas es cercana y en algunos casos superior a los resultados obtenidos con $K = OTPP$. Con base en este experimento, se concluye que el valor de K igual a 20 permite obtener valores de aptitud altos en corto tiempo de ejecución.

[1] una transacción por partición

Tabla 5.3: *Fitness* promedio de PSO-GES para $K = \{OTPP, 100, 20, 10, 5, 3\}$

Grupos	CD	OTPP	100	20	10	5	3
Menos de 1000 transacciones	**Zoo**	0.600	**0.629**	0.620	0.360	0.090	0.090
	Iris	0.270	0.280	**0.285**	0.240	0.220	0.220
	hepatitis	**0.660**	0.460	0.619	0.050	0.080	0.080
	Wine	**0.240**	0.150	0.220	0.090	0.010	0.010
	Auto	**0.640**	0.550	0.630	0.520	0.020	0.000
	Glass	**0.400**	0.360	0.330	0.320	0.140	0.040
	Heart	0.730	0.640	**0.780**	0.510	0.360	0.010
	Ecoli	0.620	0.680	**0.880**	0.590	0.380	0.440
	Ionosphere	0.610	0.550	**0.640**	0.470	0.050	0.000
	Dematology	**0.650**	0.560	0.640	0.430	0.360	0.090
	Horsecolic	0.490	0.400	**0.510**	0.480	0.380	0.000
	CylBands	0.660	**0.830**	0.650	0.000	0.260	0.030
	Soybean	**0.690**	0.560	0.560	0.120	0.000	0.140
	Breast	0.610	**0.990**	0.940	0.250	0.370	0.420
	Pima	0.790	0.740	**0.940**	0.750	0.770	0.750
	Anneal	0.640	0.450	**0.680**	0.180	0.270	0.400
	Tictactoe	**0.072**	0.070	0.070	0.010	0.000	0.000
Más de 1000 transacciones	**Flare**	0.656	**0.751**	0.742	0.144	0.346	0.154
	CharlesBook	**0.150**	0.110	0.110	0.110	0.110	0.090
	Led	**0.420**	0.220	0.390	0.230	0.100	0.140
	Waveform	**0.320**	0.230	0.230	0.000	0.000	0.000
	PageBlocks	0.980	**0.990**	0.982	0.980	0.980	0.980
	Mushroom	0.710	0.070	**0.890**	0.000	0.000	0.100
	PenDigits	**0.130**	0.090	0.122	0.000	0.000	0.000
	Nursery	**0.400**	0.378	0.392	0.070	0.164	0.178
	Letrecog	**0.537**	0.440	0.467	0.040	0.020	0.000
	Chess	**0.040**	**0.040**	**0.040**	**0.040**	**0.040**	**0.040**
	Adult	0.640	**0.800**	0.600	0.140	0.080	0.110
	Connect	**0.761**	0.606	**0.761**	0.035	0.189	0.180
	Accident	**0.750**	**0.750**	0.600	0.610	0.530	0.300
Suma		16	8	10	1	1	1

5.3. Evaluación del desempeño de PSO-GES con respecto a otras propuestas del estado del arte

Como segundo experimento, se comparó el rendimiento de PSO-GES con los algoritmos más recientes, reportados en la literatura, basados en PSO utilizados en el minado RAs: BPSO [11], GPSO [56] y PSO-DVAC [18]. Los algoritmos se compararon en términos de la calidad de las reglas generadas y del tiempo de ejecución alcanzado.

5.3.1. Evaluación del desempeño de PSO-GES respecto a la calidad de las reglas generadas

En la Tabla 5.4 se compara el rendimiento de los algoritmos BPSO, GPSO, PSO-DVAC y PSO-GES con respecto a la calidad de las reglas generadas. En esta tabla, se puede apreciar que los valores de aptitud de las RAs encontradas por BPSO, GPSO y PSO-DVAC, en la mayoría de los conjuntos de datos, son bajos. Esto se debe a que las RAs generadas por los algoritmos de Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18] tienen un mayor número de *ítem* presentes (son más grandes) y, en consecuencia, el soporte de estas reglas es menor. Sin embargo, gracias al método de búsqueda guiada introducido en esta Tesis, que promueve la incorporación de *ítems* que contribuyen de manera positiva a la calidad de las RAs, las reglas generadas por PSO-GES son más cortas y con un mayor valor de aptitud. En la Tabla 5.4 se puede observar que, independientemente del tamaño del conjunto de datos, PSO-GES genera RAs con mayor aptitud.

En esta Tesis, usamos las técnicas de prueba de hipótesis para proporcionar apoyo estadístico al análisis de los resultados obtenidos. Específicamente, para determinar si existe una diferencia estadística significativa entre los valores de aptitud de las reglas obtenidas por nuestra propuesta (Tabla 5.4) con respecto al resto de los algoritmos considerados. Se utilizó el Test de Friedman el cual informa sobre la presencia de diferencias entre los resultados comparados. El Test de Friedman, clasifica los algoritmos para cada conjunto de datos por separado, el algoritmo con los mejores valores de *fitness* obtiene el rango de 1, el segundo con mejor rendimiento rango 2, como se muestra en la Tabla 5.5. En el caso de que compartan los mismos resultados, se asignan rangos promedio.

Sea r_i^j el rango del j-ésimo algoritmo, de k algoritmos, en el i-ésimo conjunto de datos, de N conjuntos de datos. La prueba de Friedman compara los rangos promedio de los algoritmos, $R_j = \frac{1}{N}\sum_{i=1} r_i^j$. Si el número total de conjuntos de datos (N) es

Tabla 5.4: *Fitness* promedio de las reglas obtenidas por PSO-GES, la propuesta de Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18]

Grupos	CD	PSO-GES	BPSO	GPSO	PSO-DVAC
Menos de 1000 transacciones	**Zoo**	**0.620** (0.010)	0.000 (0.000)	0.000 (0.000)	0.376 (0.000)
	Iris	**0.285**(0.001)	0.001 (0.000)	0.000 (0.000)	0.282 (0.004)
	hepatitis	**0.619** (0.004)	0.000 (0.000)	0.000 (0.000)	0.155 (0.004)
	Wine	**0.220** (0.001)	0.000 (0.000)	0.000 (0.000)	0.036 (0.001)
	Auto	**0.630** (0.039)	0.056 (0.000)	0.000 (0.000)	0.028 (0.000)
	Glass	**0.330** (0.013)	0.000 (0.000)	0.000 (0.000)	0.317 (0.003)
	Heart	**0.780** (0.015)	0.000 (0.000)	0.000 (0.000)	0.445 (0.020)
	Ecoli	**0.880** (0.020)	0.000 (0.000)	0.000 (0.000)	0.843 (0.009)
	Ionosphere	**0.640** (0.019)	0.000 (0.000)	0.000 (0.000)	0.047 (0.001)
	Dematology	**0.640** (0.026)	0.000 (0.000)	0.000 (0.000)	0.291 (0.005)
	Horsecolic	**0.510** (0.057)	0.000 (0.000)	0.000 (0.000)	0.010 (0.000)
	CylBands	**0.650** (0.002)	0.000 (0.000)	0.000 (0.000)	0.010 (0.000)
	Soybean	**0.560** (0.005)	0.000 (0.000)	0.000 (0.000)	0.048 (0.000)
	Breast	**0.940** (0.021)	0.379 (0.006)	0.300 (0.011)	0.821 (0.054)
	Pima	**0.940** (0.007)	0.000 (0.000)	0.000 (0.000)	0.847 (0.001)
	Anneal	**0.680** (0.022)	0.000 (0.000)	0.000 (0.000)	0.272 (0.006)
	Tictactoe	**0.070** (0.000)	0.000 (0.000)	0.000 (0.000)	0.055 (0.000)
Más de 1000 transacciones	**Flare**	**0.742** (0.043)	0.000 (0.000)	0.000 (0.000)	0.592 (0.034)
	CharlesBook	**0.110** (0.001)	0.080 (0.001)	0.044 (0.001)	0.079 (0.001)
	Led	0.390 (0.015)	0.061 (0.001)	0.080 (0.001)	**0.399** (0.006)
	Waveform	**0.230** (0.003)	0.000 (0.000)	0.000 (0.000)	0.036 (0.000)
	PageBlocks	0.982 (0.000)	0.000 (0.000)	0.000 (0.000)	**0.991** (0.000)
	Mushroom	**0.890** (0.000)	0.000 (0.000)	0.000 (0.000)	0.080 (0.001)
	PenDigits	**0.122** (0.000)	0.000 (0.000)	0.000 (0.000)	0.022 (0.000)
	Nursery	**0.392** (0.002)	0.000 (0.000)	0.000 (0.000)	0.047 (0.047)
	Letrecog	**0.467** (0.000)	0.000 (0.000)	0.000 (0.000)	0.064 (0.002)
	Chess	**0.040** (0.043)	0.000 (0.000)	0.000 (0.000)	**0.040** (0.000)
	Adult	**0.600** (0.000)	0.000 (0.000)	0.000 (0.000)	0.566 (0.010)
	Connect	**0.761** (0.003)	0.000 (0.000)	0.000 (0.000)	0.001 (0.000)
	Accident	**0.600** (0.001)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)

Tabla 5.5: *Fitness* promedio de las reglas obtenidas por PSO-GES, la propuesta de Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18]

	PSO-GES	BPSO	GPSO	PSO-DVAC
Zoo	**0.620 (1)**	0.000 (3.5)	0.000 (3.5)	0.376 (2)
Iris	**0.285 (1)**	0.001 (3)	0.000 (4)	0.282 (2)
hepatitis	**0.619 (1)**	0.000 (3.5)	0.000 (3.5)	0.155 (2)
Wine	**0.220 (1)**	0.000 (3.5)	0.000 (3.5)	0.036 (2)
Auto	**0.630 (1)**	0.056 (2)	0.000 (4)	0.028 (3)
Glass	**0.333 (1)**	0.000 (3.5)	0.000 (3.5)	0.317 (2)
Heart	**0.780 (1)**	0.000 (3.5)	0.000 (3.5)	0.445 (2)
Ecoli	**0.880 (1)**	0.000 (3.5)	0.000 (3.5)	0.843 (2)
Ionosphere	**0.640 (1)**	0.000 (3.5)	0.000 (3.5)	0.047 (2)
Dematology	**0.640 (1)**	0.000 (3.5)	0.000 (3.5)	0.291 (2)
Horsecolic	**0.510 (1)**	0.000 (3.5)	0.000 (3.5)	0.010 (2)
CylBands	**0.650 (1)**	0.000 (3.5)	0.000 (3.5)	0.010 (2)
Soybean	**0.560 (1)**	0.000 (3.5)	0.000 (3.5)	0.048 (2)
Breast	**0.940 (1)**	0.379 (3)	0.300 (4)	0.821 (2)
Pima	**0.940 (1)**	0.000 (3.5)	0.000 (3.5)	0.847 (2)
Anneal	**0.680 (1)**	0.000 (3.5)	0.000 (3.5)	0.272 (2)
Tictactoe	**0.072 (1)**	0.000 (3.5)	0.000 (3.5)	0.055 (2)
Flare	**0.742 (1)**	0.000 (3.5)	0.000 (3.5)	0.592 (2)
CharlesBook	**0.110 (1)**	0.080 (2)	0.044 (4)	0.079 (3)
Led	0.390 (2)	0.061 (4)	0.080 (3)	**0.399 (1)**
Waveform	**0.230 (1)**	0.000 (3.5)	0.000 (3.5)	0.036 (2)
PageBlocks	0.982 (2)	0.000 (3.5)	0.000 (3.5)	**0.991 (1)**
Mushroom	**0.890 (1)**	0.000 (3.5)	0.000 (3.5)	0.080 (2)
PenDigits	**0.122 (1)**	0.000 (3.5)	0.000 (3.5)	0.022 (2)
Nursery	**0.392 (1)**	0.000 (3.5)	0.000 (3.5)	0.047 (2)
Letrecog	**0.467 (1)**	0.000 (3.5)	0.000 (3.5)	0.064 (2)
Chess	**0.040 (1.5)**	0.000 (3.5)	0.000 (3.5)	**0.040 (1.5)**
Adult	**0.600 (1)**	0.000 (3.5)	0.000 (3.5)	0.566 (2)
Connect	**0.761 (1)**	0.000 (3.5)	0.000 (3.5)	0.001 (2)
Accident	**0.600 (1)**	0.000 (3)	0.000 (3)	0.000 (3)
Rank Suma	32.500	101	106	60.5
Rank Promedio	1.083	3.366	3.533	2.016

Tabla 5.6: Valores concernientes al Test de Friedman

	Valores
N. de Algoritmos	4
N. de Datasets	30
Chi-Cuadrado	72.887
$X^2_{(3,0.05)}$	7.81

mayor de 10 la distribución de X_F^2 se aproxima a una distribución X^2 con k-1 grados de libertad. Bajo la hipótesis nula, que establece que todos los algoritmos son equivalentes y que sus rangos R_j deben ser iguales. La hipótesis nula es rechazada, si X_F^2 (Ecuación 5.1)> $X^2_{(k-1,\alpha)}$.

$$X_F^2 = \frac{12N}{k(k+1)}(\sum_{i=j} R_j^2 - \frac{k(k+1)^2}{4}) \tag{5.1}$$

Luego, $72.887 > 7.81$ entonces para un nivel de significancia de 5 %, se rechaza la hipótesis nula que establece que todos los algoritmos se desempeñan por igual, entonces existe al menos un algoritmo cuyo rendimiento es diferente del resto. El rechazo de la hipótesis nula no implica la detección de las diferencias estadísticas existentes entre los algoritmos comparados. Para detectar diferencias estadísticas, se procede con las pruebas post-hoc de Nemenyi.

La prueba post-hoc de Nemenyi, se utilizó para comparar todos los algoritmos entre sí. Esta prueba considera que el rendimiento de dos algoritmos se considera significativamente diferente si sus rangos medios difieren por lo menos en la diferencia crítica (DC):

$$DC = q_\alpha * \sqrt{\frac{k(k+1)}{6N}} \tag{5.2}$$

Donde k y N son la cantidad de algoritmos y conjuntos de datos respectivamente, y el valor q se basa en el estadístico del rango estudentizado dividida por $\sqrt{2}$. Los resultados de las diferencias de los rangos medios y el valor de DC se aprecian en la Tabla 5.7, a partir de los resultados de esta prueba se puede concluir que existe una diferencia estadística significativa entre los resultados obtenidos por el algoritmo PSO-GES con respecto al resto de las propuestas. Por tanto, el algoritmo PSO-GES produjo mejores reglas que todos los demás algoritmos (PSO, GPSO y PSO-DVAC) en los conjuntos de datos en donde fue evaluado, lo que justifica el uso de la propuesta.

Tabla 5.7: Tabla comparativa de diferencia de Ranks entre los algoritmos

Algoritmos	BPSO	GPSO	PSO-DVAC
PSO-GES	2.283	2.450	0.933
BPSO		0.167	1.350
GPSO			1.517
Diferencia Crítica = 0.855			

5.3.2. Evaluación del desempeño de PSO-GES respecto al tiempo de ejecución

En la Tabla 5.8, la primera columna representa el tiempo de ejecución alcanzado por PSO-GES, el resto de las columnas representan los tiempos de ejecución alcanzados por los algoritmos BPSO [11], GPSO [56], PSO-DVAC [18] respectivamente, en los 30 conjuntos de datos utilizados durante la experimentación. Los valores entre paréntesis indican el rendimiento del algoritmo según el tiempo de ejecución alcanzado, los menores tiempos toman el valor de 1 y el algoritmo con el tiempo de ejecución más alto toma el valor de 4 (Estos valores al igual que en la Sección 5.3.1 van a ser utilizados para determinar si existe diferencia estadística significativa, con respecto al tiempo). Como se puede apreciar, PSO-GES es más rápido que GPSO en el 97 % de los conjuntos de datos y, con respecto a BPSO y PSO-DVAC, el tiempo de ejecución es, en algunos casos, un poco más alto.

Utilizando la prueba post-hoc de Nemenyi, para un nivel de significancia de 5 %, se puede concluir que el algoritmo propuesto (PSO-GES) es significativamente más rápido que GPSO. Además, los tiempos de ejecución de PSO-GES y BPSO son similares es decir no hay diferencia estadística significativa entre sus tiempos de ejecución. Por último, PSO-DVAC es significativamente más rápido que PSO-GES. El aumento en el tiempo de ejecución de PSO-GES con respecto a PSO-DVAC es debido a que esta propuesta, independientemente de que trabaja con una tabla resumen durante toda la ejecución del algoritmo, incorpora dos nuevos pasos al algoritmo PSO que son: el cálculo de los valores estimados de soporte y confianza cada vez que un ítem es agregado a la regla. Sin embargo, a pesar de este ligero incremento en el tiempo de ejecución la calidad de las reglas generadas por PSO-GES es significativamente superior que el resto de los algoritmos con los que fue comparado.

Tabla 5.8: Tiempo de ejecución en segundos obtenido por PSO-GES, la propuesta de Sarath et al. [11], Indira et al. [56] y Qianxiang et al. [18]

Grupos	CD	PSO-GES	BPSO	GPSO	PSO-DVAC
Menos de 1000 transacciones	**Zoo**	0.851(2)	0.887(3)	3.244(4)	**0.591(1)**
	Iris	0.704(2)	0.876(3)	1.736(4)	**0.327(1)**
	hepatitis	1.043(2)	1.242(3)	4.059(4)	**0.733(1)**
	Wine	0.963(2)	1.444(3)	4.878(4)	**0.673(1)**
	Auto	2.174(2)	2.679(3)	10.104(4)	**1.040(1)**
	Glass	0.853(2)	1.027(3)	3.157(4)	**0.543(1)**
	Heart	0.970(2)	1.247(3)	3.84(4)	**0.667(1)**
	Ecoli	0.813(2)	0.951(3)	2.427(4)	**0.480(1)**
	Ionosphere	2.597(2)	3.955(3)	12.563(4)	**1.905(1)**
	Dematology	0.910(2)	1.578(3)	3.478(4)	**0.670(1)**
	Horsecolic	1.220(2)	2.642(3)	6.147(4)	**1.178(1)**
	CylBands	4.214(3)	3.298(2)	10.910(4)	**2.354(1)**
	Soybean	6.457(3)	3.103(2)	8.954(4)	**1.888(1)**
	Breast	1.454(3)	0.653(2)	1.965(4)	**0.571(1)**
	Pima	1.156(2)	1.510(3)	4.261(4)	**0.820(1)**
	Anneal	2.608(3)	2.191(2)	7.115(4)	**1.548(1)**
	Tictactoe	1.869(3)	1.227(2)	3.454(4)	**0.767(1)**
Más de 1000 transacciones	**Flare**	2.751(3)	1.281(2)	4.137(4)	**0.875(1)**
	CharlesBook	0.886(3)	**0.679(1)**	2.315(4)	0.775(2)
	Led	1.805(3)	1.731(2)	3.494(4)	**0.961(1)**
	Waveform	17.177(3)	16.892(2)	29.349(4)	**10.611(1)**
	PageBlocks	7.729(3)	6.655(2)	21.922(4)	**5.758(1)**
	Mushroom	27.322(3)	22.892(2)	37.629(4)	**11.848(1)**
	PenDigits	24.776(2)	25.710(3)	41.666(4)	**14.002(1)**
	Nursery	14.255(4)	7.450(2)	13.670(3)	**3.974(1)**
	Letrecog	44.570(3)	32.550(2)	78.051(4)	**26.109(1)**
	Chess	24.715(3)	23.270(2)	45.270(4)	**14.236(1)**
	Adult	208.808(3)	169.010(2)	445.114(4)	**117.47(1)**
	Connect	695.165(3)	**339.930(1)**	922.321(4)	350.177(2)
	Accident	4359.498(2)	6563.480(3)	7146.810(4)	**4196.228(1)**
	Rank Suma	76	72	119	32
	Rank Promedio	2.533	2.400	3.966	1.066

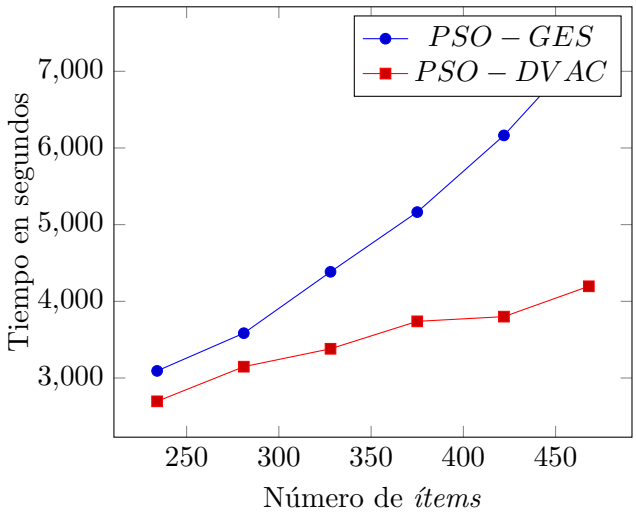

Figura 5.1: Desempeño del algoritmo PSO-GES según el número de *ítems*

5.4. Desempeño de PSO-GES con respecto al número de *ítems* existentes en el conjunto de datos

Debido a que el tamaño del CD es un factor que afecta de forma significativa el rendimiento de los algoritmos. Por esta razón, con la intención de verificar el comportamiento de PSO-GES en dependencia del número de *ítems* existentes en el conjunto de datos, se tomó *Accident* como conjunto de datos de prueba, por ser el conjunto de datos que presenta mayor cantidad de *ítems* y transacciones utilizado durante la experimentación.

Para el desarrollo de este experimento se fijaron los valores de los parámetros mínimo soporte igual a 0.2, mínima confianza igual a 0.9, número de particiones (K) igual al número de transacciones del conjunto de datos (340184). El algoritmo fue ejecutado 5 veces y en cada una de las ejecuciones fueron eliminados aleatoriamente el 50 %, 40 %, 30 %, 20 % y 10 % de los *ítems* respectivamente.

Para fines de comparación se utilizó PSO-DVAC como algoritmo de referencia por ser el algoritmo que mostró los mejores resultados de los algoritmos comparados. En la Figura 5.1 se muestran los tiempos de ejecución en segundos obtenidos por los algoritmos PSO-DVAC y el algoritmo PSO-GES, utilizando el conjunto de datos *Accident* con un número de ítems igual a 250, 300, 350, 400 y 450. De la gráfica de la Figura 5.1 se puede observar que a pesar de que PSO-DVAC tiene un mejor rendimiento que

Tabla 5.9: Reglas generadas del conjunto de datos perteneciente a la Comisión del agua del Estado de México (CAEM)

	Antecedente	**Consecuente**
Grupo 1	La causa sea por una fuerte precipitación y una precipitación pluvial extraordinaria	Infraestructura hidráulica insuficiente
	Saturación del sistema del drenaje	
	Falta de infraestructura, de limpieza y desazolve de la red de drenaje o retiro de basura	
	Inundación a viviendas de entre 15 a 39 casas afectadas	
Grupo 2	Se tengan tirantes interiores de 0 a 3 y tirantes viales de 4 a 61	Limpieza y desazolve del drenaje
	Una fuerte precipitación y una precipitación pluvial extraordinaria	
	Se identifique saturación del sistema de drenaje	
Grupo 3	Las causa son una precipitación pluvial extraordinaria y un desbordamiento de cause	Inundación de viviendas
	Acciones emergentes: reforzamiento de bordos o excavación y construcción de zanjas.	
	Inundación urbana con una población vulnerable de 800 a 1599 personas	

PSO-GES, con el algoritmo PSO-GES el tiempo de ejecución se incrementa conforme se incrementa el número de ítems presentes en el conjunto de datos. Es decir que independientemente del volumen de los datos el algoritmo propuesto tiene un desempeño estable.

5.5. Aplicación de PSO-GES a un problema de estudio real

Por último, con el objetivo de validar el algoritmo propuesto en un problema de estudio real, se aplicó el algoritmo propuesto PSO-GES al conjunto de datos perteneciente a la Comisión del agua del Estado de México (CAEM). Con el objetivo de determinar las relaciones existentes entre las inundaciones, sus causas e impactos. Para ello, se tomaron los datos recolectados en [63] del periodo 2002–2015. El conjunto de datos consta de 2447 ítems y 3572 filas, donde cada fila representa una inundación y cada ítem una situación presentada o no en la inundación.

Se estableció como criterio de éxito la selección de aquellas reglas con umbrales de soporte y confianza superiores a 0.1 y 0.7 respectivamente, el resto de los parámetros requeridos por el algoritmo PSO-GES coinciden con los definidos en la Sección 5.1. En la Tabla 5.9 se muestran las 10 mejores reglas encontradas por el algoritmo, estas fueron agrupadas según el consecuente de la regla.

Las reglas de asociación descritas en la Tabla 5.9 reflejan que de presentarse los antecedentes mencionados en el grupo 1 en una inundación se tiene una infraestructura hidráulica insuficiente. Como se observa en la Tabla 5.9, las causas más importantes para tener una infraestructura insuficiente son: precipitaciones fuertes y extraordinarias así como saturación del sistema del drenaje que es consecuencia de una falta de limpieza y desazolve de la red de drenaje, retiro de basura o una falta de infraestructura.

Las reglas de asociación del segundo grupo de la Tabla 5.9 describen los antecedentes que se deben cumplir para implementar una limpieza y desazolve de la red de drenaje. Los antecedentes más importantes observados son que se presenten tirantes interiores de 0 a 3 y tirantes viales de 4 a 61 así como que exista una precipitación fuerte y extraordinaria en una zona urbana donde se identifique una infraestructura hidráulica insuficiente y una saturación del sistema de drenaje.

Las reglas de asociación del tercer grupo de la Tabla 5.9 describen los antecedentes que indican cuales son los indicadores que deben presentarse para que haya inundaciones en viviendas. Las causas más importantes son que se presente una precipitación extraordinaria, el desbordamiento del cauce o se hayan implementado acciones emergentes como el reforzamiento de bordos, excavaciones o la construcción de zanjas.

Las 10 mejores reglas que el algoritmo minó, obtuvieron un valor de soporte superior al 50 % y la confianza de las reglas obtenidas osciló entre el 70 y 90 %. De está manera se puede validar la usabilidad del algoritmo propuesto en problemas de la vida real, debido a que extrae reglas de asociación de interés independientemente de la naturaleza del conjunto de datos.

Capítulo 6

Conclusiones y trabajos futuros

En esta Tesis, se desarrolló una nueva metaheurística de inspiración biológica guiado por la metaheurística PSO para resolver el problema de minado de RAs en conjuntos de datos con una gran cantidad de *ítems* y transacciones.

En su funcionamiento, la metaheurística propuesta, llamada PSO-GES, explora el espacio de los conjuntos de *ítems* de forma inteligente combinando el algoritmo PSO y la incorporación de un método de búsqueda guiada. Para la extracción de reglas de asociación genera una nueva estructura que tiene un número de filas menor que el conjunto de datos original, se implementa un método de búsqueda guiada que calcula los valores estimados de soporte y confianza cada vez que se agrega un ítem a la regla, que estimula la generación de reglas de asociación con altos valores de *fitness*.

Para analizar el rendimiento del enfoque propuesto, se llevaron a cabo varios experimentos en conjuntos de datos reales con diferentes números de *ítems* y transacciones. Estos conjuntos de datos fueron divididos en dos grupos: conjuntos con menos de mil transacciones y conjuntos con más de mil transacciones. Además, se validó la eficiencia del algoritmo propuesto en un problema de estudio real.

Durante la experimentación se pudo concluir que para todos los valores de K (número de particiones del conjunto de datos) evaluados, en los conjuntos de datos con más de mil transacciones, el tiempo de ejecución se redujo a más de la mitad (hasta a 8 veces). En los conjuntos de datos con menos de mil transacciones, aunque se redujo el tiempo de ejecución esta reducción no fue significativa, teniendo mayor impacto en conjuntos de datos que contienen más de 1000 transacciones. Además, con respecto al tiempo de ejecución empleado por PSO-GES, este solo fue superior al tiempo de ejecución empleado por el algoritmo PSO-DVAC.

Con respecto al número de particiones y la calidad de las reglas generadas el resultado del análisis mostró que, para valores pequeños de K, al incrementar el número

de *ítems* por partición, decrecen los valores de *fitnees* debido a que al existir un mayor número de *ítems* por partición los valores estimados se alejan de los valores reales de soporte y confianza. Por otro lado, se validó (estadísticamente) que PSO-GES genera RAs de mayor calidad que el resto de los algoritmos con los que fue comparado. Esto es posible gracias a la incorporación del método de búsqueda guiada que favorece la inserción de aquellos *ítems* que influyen de manera positiva en el *fitnees* de la regla. Además, fue comprobado que PSO-GES genera reglas de calidad independientemente de la cantidad de *ítems* y transacciones que contiene el conjunto de datos, a diferencia del resto de los algoritmos basados en PSO que tienden a converger prematuramente en soluciones de baja calidad al ver afectado su desempeño por el número de *ítems* presentes en el conjunto de datos.

Con base en lo anterior se da cumplimiento a la hipótesis de la investigación y se puede afirmar que la metaheurística desarrollada (PSO-GES) es una alternativa eficiente para minar reglas de asociación, que determinen conocimiento de interés para el problema de estudio, en grandes conjuntos de datos transaccionales.

Como trabajo futuro, se plantea adaptar la estrategia de búsqueda guiada y la estimación del soporte y la confianza a otros tipos de metaheurística de optimización bioinspirada para la extracción de RAs. También se propone la extensión de este algoritmo, para aprovechar las capacidades de cómputo paralelo masivo para disminuir el tiempo de ejecución en conjuntos de datos de mayor tamaño.

Referencias

1. D. T. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

2. G. Gupta, *Introduction to data mining with case studies*. PHI Learning Pvt. Ltd., 2014.

3. M. M. Mostafa, "Knowledge discovery of hidden consumer purchase behaviour: a market basket analysis," *International Journal of Data Analysis Techniques and Strategies*, vol. 7, no. 4, pp. 384–405, 2015.

4. Z. Abdullah, T. Herawan, N. Ahmad, and M. M. Deris, "Extracting highly positive association rules from students' enrollment data," *Procedia-Social and Behavioral Sciences*, vol. 28, pp. 107–111, 2011.

5. S. Gupta and R. Mamtora, "A survey on association rule mining in market basket analysis," *International Journal of Information and Computation Technology. ISSN*, pp. 0974–2239, 2014.

6. R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, pp. 207–216, ACM, 1993.

7. J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, vol. 29, pp. 1–12, ACM, 2000.

8. C. Kant *et al.*, "Mining association rules directly using aco without generating frequent itemsets," in *Energy Systems and Applications, 2015 International Conference on*, pp. 390–395, IEEE, 2015.

9. Y. Gao, J.-Q. Hu, and X.-L. Tang, "The application of hybrid ant colony algorithm in association rule mining," in *Computational Intelligence and Design (ISCID), 2016 9th International Symposium on*, vol. 2, pp. 329–333, IEEE, 2016.

10. R. J. Kuo, C. M. Chao, and Y. Chiu, "Application of particle swarm optimization to association rule mining," *Applied Soft Computing*, vol. 11, no. 1, pp. 326–336, 2011.

11. K. Sarath and V. Ravi, "Association rule mining using binary particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1832–1840, 2013.

12. V. Beiranvand, M. Mobasher-Kashani, and A. A. Bakar, "Multi-objective pso algorithm for mining numerical association rules without a priori discretization," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4259–4273, 2014.

13. S. Muppidi, M. R. Murty, and S. Ch, "Efficient frequent pattern mining using particle swarm optimization," *International Journal of Applied Engineering Research*, vol. 12, no. 16, pp. 5520–5524, 2017.

14. D. K. Hanirex and K. Kaliyamurthie, "Mining frequent itemsets using genetic algorithm," *Middle-East Journal of Scientific Research*, vol. 19, no. 6, pp. 807–810, 2014.

15. D. Martin, A. Rosete, J. Alcala-Fdez, and F. Herrera, "A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 54–69, 2014.

16. E. Mezura-Montes and B. C. Lopez-Ramirez, "Comparing bio-inspired algorithms in constrained optimization problems," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 662–669, IEEE, 2007.

17. B. S. Dhaliwal and S. S. Pattnaik, "Performance comparison of bio-inspired optimization algorithms for sierpinski gasket fractal antenna design," *Neural Computing and Applications*, vol. 27, no. 3, pp. 585–592, 2016.

18. S. Qianxiang and W. Ping, "Association rules mining based on improved pso algorithm," in *Computational Intelligence and Applications (ICCIA), 2017 2nd IEEE International Conference on*, pp. 145–149, IEEE, 2017.

19. A. I. R. L. Azevedo and M. F. Santos, "Kdd, semma and crisp-dm: a parallel overview," *IADS-DM*, 2008.

20. J. Vanrell, R. Bertone, and R. García-Martínez, *Un Modelo de Procesos para Proyectos de Explotación de Información.* PhD thesis, Tesis de Maestría en Ingeniería en Sistemas de Información. Escuela de Postgrado FRBA-UTN, 2011.

21. S. Moro, R. Laureano, and P. Cortez, "Using data mining for bank direct marketing: An application of the crisp-dm methodology," in *Proceedings of European Simulation and Modelling Conference-ESM'2011*, pp. 117–121, Eurosis, 2011.

22. D. M. Tank, "Improved apriori algorithm for mining association rules," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 6, no. 7, p. 15, 2014.

23. M. A. Malberti Riveros and G. Elida Beguerí, "Reglas de asociación con los datos de una biblioteca universitaria," *Revista Cubana de Ciencias Informáticas*, vol. 9, no. 4, pp. 30–45, 2015.

24. H. Orallo, J. RAMIREZ, C. R. QUINTANA, M. J. H. Orallo, M. J. R. Quintana, and C. F. Ramírez, *Introducción a la Minería de Datos.* Pearson Prentice Hall,, 2004.

25. J.-M. Adamo, *Data mining for association rules and sequential patterns: sequential and parallel algorithms.* Springer Science & Business Media, 2012.

26. Y. R. G. ANSEL, "Descubrimiento de patrones similares frecuentes para la minería de reglas de asociación sobre datos mezclados," 2011.

27. R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, 1994.

28. J. S. Park, M.-S. Chen, and P. S. Yu, "Using a hash-based method with transaction trimming for mining association rules," *IEEE transactions on knowledge and data engineering*, vol. 9, no. 5, pp. 813–825, 1997.

29. A. Savasere, E. R. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," tech. rep., Georgia Institute of Technology, 1995.

30. J. D. Holt and S. M. Chung, "Mining association rules using inverted hashing and pruning," *Information Processing Letters*, vol. 83, no. 4, pp. 211–220, 2002.

31. A. Pietracaprina, "Mining frequent itemsets using patricia tries," 2003.

32. Y. G. Sucahyo and R. P. Gopalan, "Ct-itl: Efficient frequent item set mining using a compressed prefix tree with pattern growth," in *Proceedings of the 14th Australasian database conference-Volume 17*, pp. 95–104, Australian Computer Society, Inc., 2003.

33. Y. G. Sucahyo and R. P. Gopalan, "Ct-pro: A bottom-up non recursive frequent itemset mining algorithm using compressed fp-tree data structure.," in *FIMI*, vol. 4, pp. 212–223, 2004.

34. S. Ahmed, F. Coenen, and P. Leng, "Tree-based partitioning of date for association rule mining," *Knowledge and information systems*, vol. 10, no. 3, pp. 315–331, 2006.

35. R. T. Pereira, "Equipasso: un algoritmo para el descubrimiento de conjuntos de ítems frecuentes sin generación de candidatos," *Ventana Informática*, no. 25, pp. 63–82, 2011.

36. W. M. Grandinetti, "Tesis de magister en ciencias de la computación,"

37. K. Malik, N. Raheja, and P. Garg, "Enhanced fp-growth algorithm," *IJCEM International Journal of Computational Engineering and Management*, vol. 12, pp. 54–56, 2011.

38. S. Ayubi, M. K. Muyeba, A. Baraani, and J. Keane, "An algorithm to mine general association rules from tabular data," *Information Sciences*, vol. 179, no. 20, pp. 3520–3539, 2009.

39. G. Chen and Q. Wei, "Fuzzy association rules and the extended mining algorithms," *Information Sciences*, vol. 147, no. 1-4, pp. 201–228, 2002.

40. D. W. Choi and Y. J. Hyun, "Transitive association rule discovery by considering strategic importance," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pp. 1654–1659, IEEE, 2010.

41. L.-M. Tsai, S.-J. Lin, and D.-L. Yang, "Efficient mining of generalized negative association rules," in *Granular Computing (GrC), 2010 IEEE International Conference on*, pp. 471–476, IEEE, 2010.

42. Y.-H. Hu and Y.-L. Chen, "Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism," *Decision Support Systems*, vol. 42, no. 1, pp. 1–24, 2006.

43. I. H. Osman and J. P. Kelly, *Meta-heuristics: theory and applications*. Springer Science & Business Media, 2012.

44. B. Jarboui, H. Derbel, S. Hanafi, and N. Mladenović, "Variable neighborhood search for location routing," *Computers & Operations Research*, vol. 40, no. 1, pp. 47–57, 2013.

45. C. Kant *et al.*, "Association rule mining using ant colony optimization," 2015.

46. P. Sharma, S. Tiwari, and M. Gupta, "Optimize association rules using artificial bee colony algorithm with mutation," in *Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on*, pp. 370–373, IEEE, 2015.

47. R. Miri, A. Agrawal, A. Miri, and S. Tandan, "A novel approach for reducing the candidate item sets and large item sets by fuzzy mining association rule," *Fuzzy Systems*, vol. 8, no. 5, pp. 126–128, 2016.

48. D. Martín, J. Alcalá-Fdez, A. Rosete, and F. Herrera, "Nicgar: A niching genetic algorithm to mine a diverse set of interesting quantitative association rules," *Information Sciences*, vol. 355, pp. 208–228, 2016.

49. H. Feng, R. Liao, F. Liu, Y. Wang, Z. Yu, and X. Zhu, "Optimization algorithm improvement of association rule mining based on particle swarm optimization," in *Measuring Technology and Mechatronics Automation (ICMTMA), 2018 10th International Conference on*, pp. 524–529, IEEE, 2018.

50. M. Agrawal, M. Mishra, and S. P. S. Kushwah, "Association rules optimization using particle swarm optimization algorithm with mutation," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 5, no. 1, 2015.

51. R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory in: Proceedings of the sixth international symposium on micro machine and human science, vol 43 ieee," *New York*, 1995.

52. J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, pp. 4104–4108, IEEE, 1997.

53. O. M. Badawy, A.-E. A. Sallam, and M. I. Habib, "Quantitative association rule mining using a hybrid pso/aco algorithm," tech. rep., PSO/ACO-AR, 2008.

54. B. Alatas and E. Akin, "Rough particle swarm optimization and its applications in data mining," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 12, no. 12, pp. 1205–1218, 2008.

55. M. Nandhini, M. Janani, and S. Sivanandham, "Association rule mining using swarm intelligence and domain ontology," in *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*, pp. 537–541, IEEE, 2012.

56. K. Indira and S. Kanmani, "Mining association rules using hybrid genetic algorithm and particle swarm optimisation algorithm," *International Journal of Data Analysis Techniques and Strategies*, vol. 7, no. 1, pp. 59–76, 2015.

57. A. Agarwal and N. Nanavati, "Association rule mining using hybrid ga-pso for multi-objective optimisation," in *Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on*, pp. 1–7, IEEE, 2016.

58. L. V. Lakshmanan, C. K.-S. Leung, and R. T. Ng, "The segment support map: Scalable mining of frequent itemsets," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 2, pp. 21–27, 2000.

59. H. Ishibuchi, T. Nakashima, and T. Murata, "Comparison of the michigan and pittsburgh approaches to the design of fuzzy classification systems," *Electronics and Communications in Japan(Part III Fundamental Electronic Science)*, vol. 80, no. 12, pp. 10–19, 1997.

60. X. Yan, C. Zhang, and S. Zhang, "Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3066–3076, 2009.

61. M. Gupta, "Application of weighted particle swarm optimization in association rule mining," *International Journal of Computer Science and Informatics*, vol. 1, pp. 2231–5292, 2012.

62. S. Kung, *Applying Genetic Algorithm and Weight Itemgupta2012application to Association Rule*. PhD thesis, YuanZe University, 2002. unpublished thesis.

63. S. Castro Soto, *Descubrimiento de patrones frecuentes en el atlas de inundaciones del Estado de México*. PhD thesis, Tesis de Licenciatura en Ingeniería en Computación. Facultad de Ingeniería UAEM, 2011.