Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Ridowati Gunawan
Assignment title:	Periksa similarity
Submission title:	Performance comparison of inertia weight and acceleration ...
File name:	cients_of_BPSO_in_the_context_of_high_utility_itemset_minin...
File size:	9.75M
Page count:	19
Word count:	10,421
Character count:	54,853
Submission date:	07-Mar-2022 11:27AM (UTC+0700)
Submission ID:	1778180464

Evolutionary Intelligence
https://doi.org/10.1007/s12065-022-00707-0

RESEARCH PAPER

Performance comparison of inertia weight and acceleration coefficients of BPSO in the context of high-utility itemset mining

Ridowati Gunawan[1] · Edi Winarko[2] · Reza Pulungan[2]

Received: 7 January 2021 / Revised: 4 October 2021 / Accepted: 4 February 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
One of the best performing methods in high-utility itemsets mining is based on binary particle swarm optimization. This paper aims to improve the quality of that method by finding ways to increase both the number of obtained itemsets and the associated total utility of those itemsets. The method used is based on binary particle swarm optimization, which is one of the best algorithms to obtain high-utility itemsets based on swarm intelligence. To get the maximum results we tuned up the method to determine the optimal configurations of four parameters: initial population, inertia weight, acceleration coefficients, and velocity clamping. Our experimental results on five datasets indicate that the best configuration is combining constriction factor or adaptive method for setting inertia weight and using self-adaptive for acceleration coefficient. The use of frequent itemset or transactions of the highest total utility for initializing population does not have a significant addition compared to the inertia weight and acceleration coefficient parameters. The correct configuration of the binary particle swarm optimization can increase the number of high-utility itemsets compared to the standard configuration. The percentage increase of the number of itemset compared to the existing standard method is 157.93% for chess dataset, 134.39% for connect dataset, 42.69% for accident dataset, 4.62% for foodmart dataset, and 1.58% for mushroom dataset.

Keywords Binary particle swarm optimization · High-utility itemsets · Population initialization · Inertia weight · Acceleration coefficient · Velocity clamping

1 Introduction

High-utility itemset mining is an effort to remove the main flaws of frequent itemset mining: frequent itemset mining only pays attention to the occurrence of itemsets, and hence all items are considered to have the same quantity and the same weight or utility. These two weaknesses led to studies of high utility itemsets. Itemsets obtained by considering both will be more acceptable to the user because the acquired itemsets become more meaningful and fit the actual real-life conditions. Researches into obtaining high utility itemsets have been carried out well using both classical algorithms as well as computational intelligence. Combining high-utility itemset mining with computational intelligence has been done using genetic algorithm (GA), ant colony optimization (ACO), and binary particle swarm optimization (BPSO).

High-utility itemset mining has not been as good as frequent itemset mining in producing "quality" itemsets when using computational intelligence. Previous researches indicate that BPSO-based approach [23–25] produces more high utility itemsets than GA-based approach [17]. For some datasets, however, even BPSO-based approach is still not optimal compared to classical algorithmic approaches; in the sense that it is still possible to increase the number of obtained itemsets.

In BPSO-based approach, high utility itemsets are obtained from particles whose utility is higher than a predetermined minimum utility threshold. Particles that do not meet the minimum utility threshold are discarded and will

✉ Ridowati Gunawan
rido@usd.ac.id

Edi Winarko
ewinarko@ugm.ac.id

Reza Pulungan
pulungan@ugm.ac.id

[1] Department of Informatics, Sanata Dharma University, Yogyakarta, Indonesia

[2] Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia

Published online: 03 March 2022

Performance comparison of inertia weight and acceleration coefficients of BPSO in the context of high-utility itemset mining

by Gunawan Ridowati

Submission date: 07-Mar-2022 11:27AM (UTC+0700)
Submission ID: 1778180464
File name: cients_of_BPSO_in_the_context_of_high_utility_itemset_mining.pdf (9.75M)
Word count: 10421
Character count: 54853

Evolutionary Intelligence
https://doi.org/10.1007/s12065-022-00707-0

RESEARCH PAPER

Performance comparison of inertia weight and acceleration coefficients of BPSO in the context of high-utility itemset mining

Ridowati Gunawan[1] · Edi Winarko[2] · Reza Pulungan[2]

Received: 7 January 2021 / Revised: 4 October 2021 / Accepted: 4 February 2022

Abstract
One of the best performing methods in high-utility itemsets mining is based on binary particle swarm optimization. This paper aims to improve the quality of that method by finding ways to increase both the number of obtained itemsets and the associated total utility of those itemsets. The method used is based on binary particle swarm optimization, which is one of the best algorithms to obtain high-utility itemsets based on swarm intelligence. To get the maximum results we tuned up the method to determine the optimal configurations of four parameters: initial population, inertia weight, acceleration coefficients, and velocity clamping. Our experimental results on five datasets indicate that the best configuration is combining constriction factor or adaptive method for setting inertia weight and using self-adaptive for acceleration coefficient. The use of frequent itemset or transactions of the highest total utility for initializing population does not have a significant addition compared to the inertia weight and acceleration coefficient parameters. The correct configuration of the binary particle swarm optimization can increase the number of high-utility itemsets compared to the standard configuration. The percentage increase of the number of itemset compared to the existing standard method is 157.93% for chess dataset, 134.39% for connect dataset, 42.69% for accident dataset, 4.62% for foodmart dataset, and 1.58% for mushroom dataset.

Keywords Binary particle swarm optimization · High-utility itemsets · Population initialization · Inertia weight · Acceleration coefficient · Velocity clamping

1 Introduction

High-utility itemset mining is an effort to remove the main flaws of frequent itemset mining: frequent itemset mining only pays attention to the occurrence of itemsets, and hence all items are considered to have the same quantity and the same weight or utility. These two weaknesses led to studies of high utility itemsets. Itemsets obtained by considering both will be more acceptable to the user because the acquired itemsets become more meaningful and fit the actual real-life conditions. Researches into obtaining high utility itemsets have been carried out well using both classical algorithms as well as computational intelligence. Combining high-utility itemset mining with computational intelligence has been done using genetic algorithm (GA), ant colony optimization (ACO), and binary particle swarm optimization (BPSO).

High-utility itemset mining has not been as good as frequent itemset mining in producing "quality" itemsets when using computational intelligence. Previous researches indicate that BPSO-based approach [23–25] produces more high utility itemsets than GA-based approach [17]. For some datasets, however, even BPSO-based approach is still not optimal compared to classical algorithmic approaches; in the sense that it is still possible to increase the number of obtained itemsets.

In BPSO-based approach, high utility itemsets are obtained from particles whose utility is higher than a predetermined minimum utility threshold. Particles that do not meet the minimum utility threshold are discarded and will

✉ Ridowati Gunawan
rido@usd.ac.id

Edi Winarko
ewinarko@ugm.ac.id

Reza Pulungan
pulungan@ugm.ac.id

[1] Department of Informatics, Sanata Dharma University, Yogyakarta, Indonesia

[2] Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia

Published online: 03 March 2022

not be reused in subsequent evolutionary processes. The particles themselves arise from two main processes, namely population initialization and particles updating processes. In population initialization, transactions are commonly used as initial population [21]. The performance of PSO can be improved by strategically selecting the starting positions of the particles [32]. Good initial population can facilitate the algorithm to locate optima, and bad initial population may prevent the algorithm from finding optima [18]. The process for updating particles is influenced by the velocity update process. Parameters that affect velocity updating process, include inertia weight, acceleration coefficients, and velocity clamping [12, 20, 22, 26, 31, 34].

Many studies have proposed many ways to determine inertia weight parameter. Acceleration coefficients are the key parameters used by PSO algorithms to control the movement of particles by modifying its cognitive and social components [30]. However, acceleration coefficients used in PSO algorithm are not directly applicable to BPSO algorithm [4, 9, 29, 33, 36]. Velocity clamping has also been shown to improve the performance of BPSO algorithm [6, 12].

Even with the flourishing previous studies, tuning of those parameters in both processes has not been attempted in the field of high-utility itemset mining. Because the basis of the search for high utility itemsets uses PSO, we expect that discovering and employing the best configuration of those parameters can significantly improve the performance of high-utility itemset mining. In this paper, we propose to improve the state-of-the-art BPSO-based high utility itemset mining by tuning the method's initial population, inertia weight, acceleration coefficient, and velocity clamping. The main contribution of the paper is an improved BPSO-based algorithm that can produce more and "better" high utility itemsets compared with existing algorithms.

The rest of this paper is organized as follows: Related work is briefly reviewed in Section 2. Preliminary study on the definition of high-utility itemset and BPSO approach to obtained high-utility itemset are described in Section 3. The proposed algorithms for mining high-utility itemset are presented in Section 4. Experimental design, result, and analysis are provided in Section 5. The paper is concluded in Section 6.

2 Related work

The focus of researches on high-utility itemset mining is to obtain itemsets with utility not less than a minimum utility threshold. There are two approaches on high-utility itemset mining: using classical algorithms and using computational intelligence.

The classical algorithm approach was initiated by [37], who performed the mining by first generating candidate itemsets obtained by the cross product of all items. They also proved that the apriori property, namely the downward closure, does not apply in the search for high utility itemsets. This implies that the search for high utility itemsets must be conducted on all combinations of itemsets. A two-phase algorithm proposed by [28] performed candidate itemset searching in two phases: first by carrying out a pruning process that removes items whose utility is less than the transaction weighted utilization (TWU), and second by generating itemsets from the remaining items of the previous phase. In the first phase the transaction-weighted downward closure (TWDC) applies, which is similar to the *apriori* property in the frequent itemset search. Although this algorithm makes use of TWDC to reduce itemset candidates, memory usage and computation time are still prohibitive.

An empirical study on the use of classical algorithms for high-utility itemset mining was carried out by [39], who concluded that EFIM [41] is efficient in memory usage and appropriate for small and dense datasets, while d2HUP [27] produces a better speed than EFIM, especially for large sparse datasets.

In computational intelligence approach, GA [17], BPSO [23–25], and ACO [35] have been used to find high utility itemsets.

[17] used genetic algorithm with two approaches: with a minimum utility threshold (HUPE$_{umu}$-GA) and without a minimum utility threshold (HUPE$_{wumu}$-GA). Their results indicated that their computation time and memory usage are better than using those of non-genetic-algorithm methods. In term of computation time, HUPE$_{umu}$-GA is faster than HUPE$_{wumu}$-GA. HUPE$_{umu}$-GA uses the TWDC property as specified by [28] as the first phase to remove items with utility below the minimum utility threshold, while for fitness function it uses the measurement utility as proposed by [38]. HUPE$_{wumu}$-GA also use the same fitness function but the selected itemsets are the Top-K itemsets from the population as produced by the evolutionary process.

[23, 24] proposed HUIM-BPSOsig, a BPSO-based algorithm that uses the TWU strategy [28] from the classical high-utility itemset mining to search for high transaction-weighted-utilization 1-itemsets (1-HTWUIs). Based on the TWDC property of the TWU strategy, items with TWU value below the minimum utility threshold will be pruned in order to reduce the particle's size. They also adopted sigmoid function in the BPSO [19]. Evaluation results showed that their approach is more efficient in identifying high utility itemsets and has a faster computation time than the GA-based algorithms.

Besides the TWU model and the sigmoid function, [25] used the OR/NOR-tree structure in the process of encoding particles to eliminate invalid combinations of particles. A combination is invalid if the resulting particle is not found in the transaction. The proposed algorithm is called

HUIM-BPSO-tree and it produces better results than HUIM-BPSOsig. Although successful in obtaining high utility itemsets using BPSO approach, the number of obtained itemsets is still fewer than obtained by the classical algorithm proposed in [28]. This indicates that the results are still not optimal and there is still room for further improvement.

Optimization on high-utility itemset mining by using BPSO is mainly influenced by two main processes, population initialization and particle updating. The particle updating process itself is, in turn, influenced by velocity updating. In general, population is initialized by using random values or zero vectors. [21], however, showed that, in association rules mining, better performance is achieved when population for is initialized transactions with high fitness. On the other hand, [22] reported that the main parameters that affect velocity updating include acceleration, inertia constants, maximum velocity, and the topology or neighborhood structure, while [12] signified the role played by inertia weight, and [20] emphasized the effects of inertia weight and velocity clamping.

[20] used probabilistic approach in velocity updating. They claimed that binary or discrete PSO encounters problems in determining inertia weight and decided to take a continuous approach to the binary version. They interpreted the velocity based on changes in individual particle and global particle and the velocity of a particle is the rate at which the particle changes its bit's value. The results obtained are better than velocity updating proposed by [19] and [34].

[31] conducted a comparative study for various inertia weight and constriction factor, and concluded that the use of adaptive inertia weight (AdaptW) produces the best performance, while the use of constriction factor can be the next alternative for improvement. [16] compared 27 variants of PSO algorithm, eight modifications using inertia weight parameter and one using constriction factor parameter. The best modification of PSO algorithm is the AdaptW variant. However, the studies in [31] and [16] were not on BPSO but instead on continuous PSO.

[26] analyzed the use of inertia weight in BPSO. Based on experimental studies on the 0/1 knapsack problem, an increasing inertia weight scheme performs much better than a decreasing or a constant scheme. Even though increasing inertia weight has been shown better, this scheme has not been used in high-utility itemset mining and [25] still used constant inertia weight.

Acceleration coefficients are of two types, namely individual acceleration c_1 and social acceleration coefficients c_2, affecting the personal best and neighborhood best solutions, respectively, when updating the velocity in the current iteration. In general, for BPSO, individual and social coefficients use the same value, namely 2 [25]. However, when we use constriction factor in the velocity updating process, individual and social acceleration coefficients can be different as long as their total is more than 4 [8].

In addition to using a static value of 2, various acceleration coefficient values can also be used. Using the idea that the individual values and social acceleration coefficients are different, the PSO self-tune uses the acceleration constants c_1 and c_2, both are reduced linearly throughout the time steps in the range of 2 to 1.49 [10]. Self-adaptive acceleration coefficient PSO (PSO-SAAC) allows the acceleration coefficient to be adjusted by adapting to local search and global search [10]. Self-adaptive acceleration coefficients functions can also be combined with the value of the fitness function. This can accelerate the convergence process like the experiments carried out by [30] and [9]. However, a wide variety of acceleration coefficients have not been attempted to perform enhanced searches of high-utility itemset. The acceleration coefficients used are then using the static acceleration coefficient with the same value of c_1 and c_2, which is 2.

To maintain a "stable" position for each particle, [34] used a velocity-clamping method. The velocity of a particle in each dimension is clamped down to a maximum velocity v_{max} if it is more than v_{max}. This method is called "v_{max} method" [11].

The computational intelligence approach has also been applied to various other fields, including text clustering process [1–5]. A comprehensive analysis of the use of meta-heuristic algorithms in the context of text clustering has been carried out by [3]. This will help researchers when conducting research using a meta-heuristic algorithm. The context of the problem being resolved is text-clustering. Different problems can also be applied using various meta-heuristic methods. In general, the algorithm used uses standard parameters, such as the PSO and does not change the parameters of the PSO itself. Changing parameters should also be tried in order to gain more optimal results.

Experiments using seven nature-inspired optimization algorithms against 7 sets of datasets show that in general the optimization algorithm performance shows almost the same results, only with a slight difference. However, when viewed from the accuracy and also the F-measure, Gray Wolf Optimizer (GWO) shows the best results while Genetic Algorithm (GA) shows the worst results. The proposal to use a new nature-inspired algorithm could be considered for application both for text clustering and for high-utility itemsets mining [1].

To improve the performance of text-clustering, we have succeeded in using PSO in the feature selection process (feature and new feature) as well as using the enhanced krill herd algorithm for the text clustering process [4]. PSO is not applied to the entire text clustering process but for the feature selection process. The PSO used still uses standard parameters, testing using PSO parameter modifications can

be tried to do so that it can improve the performance of the PSO algorithm for the feature selection process.

Arithmetic Optimization Algorithm (AOA) gives the idea that in order to get optimum results, a meta-heuristic based optimization algorithm needs to be combined and not just stand alone. Algorithm modification is carried out using the idea of arithmetic as a fundamental component of number theory and is an important part of modern mathematics. The arithmetic operators of multiplication, division, subtraction and addition are applied to two phases in the population-based optimization method, namely the exploration and exploitation phases. Exploration to get the search space of an algorithm and exploitation to improve accuracy [2]. Currently the optimization-based high-utility itemset mining method still focuses on using only one algorithm (for example using PSO), it has not yet combined several algorithms. The AOA algorithm can be tried to be applied more widely, especially when using PSO, it can use the most optimal PSO parameters, as proposed in this paper.

3 Preliminaries

3.1 Definitions

Given a finite set of items $I = \{i_1, i_2, \ldots, i_m\}$, an itemset $X \subseteq I$ is a set of distinct k items. A transaction database $D = \{T_1, T_2, \ldots, T_n\}$ is a set of transactions T_c, which are basically itemsets also. Each item $i \in I$ is associated with a positive value $pr(i)$, called the external utility (*e.g.*, profit). Each item i in a transaction T_c is also associated with a positive value $q(i, T_c)$, called the internal or transaction utility (*e.g.*, quantity).

Definition 1 The following definitions are needed [14, 25, 40]:

1. The utility of an item i in transaction T_c is defined by $u(i, T_c) = pr(i) \times q(i, T_c)$.
2. The utility of itemset X in transaction T_c is defined by:

$$u(X, T_c) = \begin{cases} \Sigma_{i \in X} u(i, T_c), & \text{if } X \subseteq T_c, \\ 0, & \text{otherwise.} \end{cases}$$

3. The utility of itemset X in database D is defined by $u(X) = \Sigma_{T_c \in g(X)} u(X, T_c)$, where $g(X)$ is the set of all transactions containing itemset X.
4. The transaction utility of transaction T_c is defined by $TU(T_c) = \Sigma_{X \subseteq T_c} u(X, T_c)$.
5. The transaction weighted utility (TWU) of itemset X in database D is defined by $TWU(X) = \Sigma_{T_c \in g(X)} TU(T_c)$.
6. The total utility of database D is defined by $TU = \Sigma_{T_c \in D} TU(T_c)$.

Definition 2 An itemset X in database D is a *high utility itemset* if its utility is no less than a minimum utility threshold $TU \times \delta$, for $0 \leq \delta \leq 1$. *HUIs, the set of all high utility itemsets*, is then defined by:

$$HUIs = \{X \mid u(X) \geq TU \times \delta\}. \quad (1)$$

3.2 BPSO-based high-utility itemset mining

In the following, we briefly describe the state-of-the-art BPSO-based method [25], on which our proposed method is based. This method mines high utility itemsets according to the model of [19], which consists of four phases: pre-processing, particle encoding, fitness evaluation, and updating phases. The pseudo-code of this method is shown in Algorithm 1.

Algorithm 1 The standard BPSO-based high-utility itemset mining (Lin et al., 2017)

```
1: function HUIM-BPSO(D, δ, pop_size, iterations): HUIs
2:     for each transaction T ∈ D do                          ▷ Phase 1: pre-processing
3:         Compute utility of each item i ∈ T: u(i, T)
4:         Compute total utility of T: TU(T)
5:         Compute transaction weighted utility of item i ∈ T: TWU(i)
6:     end for
7:     Compute total utility in database D: TU
8:     Prepare transactions in D                              ▷ Tuneable #1
9:     Population P ← ∅                ▷ Phase 2: particles encoding and initialization
10:    for j ← 1 to pop_size do
11:        T ← Dequeue(D)
12:        Encode T into particle p
13:        Generate velocity v
14:        Compute fitness of p: fit(p)                       ▷ Phase 3: fitness evaluation
15:        if fit(p) > δ × TU then
16:            P ← P ∪ { (p, v, fit(p)) }
17:        end if
18:    end for
19:    HUIs ← Copy(P)
20:    P_(b) ← Copy(P)
21:    p_(g) ← FindBestParticle(P_(b))
22:    for i ← 1 to iterations do                             ▷ Phase 4: particles updating
23:        for each (p, v, fit(p)) ∈ P and corr. (p_(b), v_(b), fit(p_(b))) ∈ P_(b) do
24:            Update velocity v                              ▷ Tuneable #2 and #3
25:            Update particle p                              ▷ Eq. (4)
26:            Compute fitness of p: fit(p)                   ▷ Eq. (3)
27:            if fit(p) > fit(p_(b)) then
28:                p_(b) ← Copy(p)
29:                p_(g) ← FindBestParticle(P_(b))
30:            end if
31:            if fit(p) > δ × TU then
32:                HUIs ← HUIs ∪ { p }
33:            end if
34:        end for
35:    end for
36:    return HUIs
37: end function
```

Pre-processing In this phase, the utility of each item, the total utility of each transaction, the transaction weighted utility of each item as well as the total utility of the whole database are calculated. Based on these, the minimum utility threshold is determined. The transactions in the database are then prepared (line 8), which may involve pruning items, whose TWU is less than the minimum utility threshold, from the transactions.

Particles encoding and initialization In this phase, particles are encoded and put in the population. Particles usually originate from the transactions prepared in the previous phase. Let m be the number of distinct items in all transactions. To each transaction T, we associate a particle $\mathbf{p} \in \{0,1\}^m$ (hence, a vector) such that:

$$p_i = \begin{cases} 1, & \text{if item } i \in T, \\ 0, & \text{if item } i \notin T. \end{cases} \tag{2}$$

Once encoded, the velocity of each particle is usually randomly initialized (line 13) and the resulting particle becomes a member of the original population.

Fitness evaluation In this phase, the fitness of each particle $\mathbf{p}$ is evaluated according to fitness function:

$$fit(\mathbf{p}) = u(X), \tag{3}$$

where $u(X)$ is the utility according to Definition 1.3, and X is an itemset containing precisely all items in particle $\mathbf{p}$.

Particles updating In the last phase, particles are updated according to its updated velocity (line 24), local best particle $\mathbf{p}_{(b)}$, and global best particle $\mathbf{p}_{(g)}$ using the sigmoid function [19]:

$$p_i(t+1) = \begin{cases} 1, & \text{if } r() < S(v_i(t+1)), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $r()$ is a random number generated for each component of each particle, $S(x) = \dfrac{1}{1+e^{-x}}$, and:

$$v_i(t+1) = wv_i(t) + c_1 r_1 \left(p_{(b)i} - p_i(t)\right) + c_2 r_2 \left(p_{(g)i} - p_i(t)\right), \tag{5}$$

In (5), $v_i(t)$ represents the velocity of the i-th component of particle $\mathbf{p}$ at the t-th iteration, w is the inertia weight, which plays a balancing role between the global and local searches, $r_1, r_2 \in (0,1)$ are two random numbers generated in each iteration, and c_1 and c_2 are the individual and social acceleration factors, respectively. Particle $\mathbf{p}_{(g)}$ is the particle with the best fitness among the local best particles $\mathbf{p}_{(b)}$ ever existed.

The fitness of each updated particle is re-evaluated using (3). If the newly updated particle is fitter than its corresponding best pair in P_b, the new particle replaces it and a new global best particle is sought among particles in P_b, by invoking function FindBestParticle(). Particles with a utility more than the minimum utility threshold represents a high utility itemset.

4 Proposed method

In our effort to devise a superior BPSO-based high-utility itemset mining with a pre-determined minimum utility threshold, we tune four parameters up to find an optimal configuration, namely initial population, inertia weight, acceleration coefficients, and velocity clamping. In the following, we elaborate on how these parameters are tuned up or determined.

4.1 Initial population

In forming the initial population, particles are generated from the transactions, which have previously been prepared, and the initial velocity of those particles is determined. We will study three ways of generating initial particles: with zero vectors, based on frequent itemsets, and based on the total utility of the transactions. Hence, a population of size n will be formed by n zero vectors, by particles representing the top n frequent itemsets, or by particles formed by n transactions of the highest total utility, respectively. Initial velocity, on the other hand, will be initialized by a random number in the interval $[0, 1]$.

4.2 Inertia weight

Four alternatives will be experimented on for inertia weight parameter, namely increasing [26], adaptive [31], constant inertia weight with probabilistic velocity [34]. For the fourth alternative, we propose to use constriction factor, which has a similar role as inertia weight in velocity updating.

Increasing. In increasing inertia weight, the weight w is set by [22]:

$$w = \begin{cases} \underline{w} + \frac{\pi(\overline{w}-\underline{w})}{\rho\overline{\pi}}, & \text{if } \pi \le \rho\overline{\pi}, \\ \overline{w} & \text{if } \rho\overline{\pi} < \pi \le \overline{\pi}, \end{cases} \tag{6}$$

where π and $\overline{\pi}$ stand for the number of elapsed iterations and the maximal number of iterations, respectively, and $\underline{w}$ and $\overline{w}$ are the lower and upper bound of w, respectively. Constant ρ, $0 \le \rho \le 1$, is a parameter to control the number of iterations to make w increase from $\underline{w}$ to $\overline{w}$, after which w is fixed to $\overline{w}$ and there is no adaption any longer. According to previous studies, ρ is set to 0.9.

Adaptive. Adaptive inertia weight depends on particles' success, which at iteration t is defined by:

$$s(\mathbf{p}, t) = \begin{cases} 1, & \text{if } fit(\mathbf{p}(t)) > fit(\mathbf{p}(t-1)), \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where $fit()$ is the function to optimize, which in our case (3).

Based on each particle' success, the percentage of the particles that experience improved fitness in the last iteration is:

$$P_s(t) = \frac{\sum_{\forall \mathbf{p} \in P_{(b)}} S(\mathbf{p}, t)}{n}, \tag{8}$$

where n is the number of particles.

Inertia weight can then be set as any increasing function of P_s:

$$w(t) = f(P_s(t)). \tag{9}$$

Furthermore, a linear function may be used to map the value of P_s to a possible range of inertia weight:

$$w(t) = (w_{\max} - w_{\min}) P_s(t) + w_{\min}. \tag{10}$$

In this study, we set $w_{\min} = 0$ and $w_{\max} = 1$.

Constant with probabilistic velocity. In this alternative, we use a constant inertia weight—namely 0.9 [25]—coupled with probabilistic velocity as proposed by [34]. To update velocity, we don't use (5), but instead:

$$v_i(t+1) = \begin{cases} v_{(0)}(t+1), & \text{if } p_i = 0, \\ v_{(1)}(t+1), & \text{if } p_i = 1, \end{cases} \quad (11)$$

where:

$$v_{(0)}(t+1) = wv_{(0)}(t) + (-1)^{p_{(b)i}} c_1 r_1 + (-1)^{p_{(g)i}} c_2 r_2, \quad (12)$$

$$v_{(1)}(t+1) = wv_{(1)}(t) + (-1)^{1-p_{(b)i}} c_1 r_1 + (-1)^{1-p_{(g)i}} c_2 r_2, \quad (13)$$

and w is the inertia weight (0.9), $r_1, r_2 \in (0, 1)$ two random numbers generated in each iteration, and c_1 and c_2 are the individual and social acceleration factors, respectively, which are determined by the user.

Constriction factor. Updating velocity using a constriction factor was first proposed by [8], as follows:

$$v_i(t+1) = K[v_i(t) + c_1 r_1 (p_{(b)i} - p_i(t)) + c_2 r_2 (p_{(g)i} - p_i(t))], \quad (14)$$

where K is the constriction factor:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad (15)$$

and $\varphi = c_1 + c_2$ and $\varphi \geq 4$.

4.3 Acceleration coefficients

For acceleration coefficient, four alternatives are carried out, namely the constant acceleration coefficient, self-tuned (ST) [10], self-adaptive acceleration coefficient (SAAC) [10] dan fitness-based acceleration coefficient (FR) [30].

Constant. in constant acceleration coefficient c_1 and c_2 values are always the same, which is 2

Self-Tuned (ST). in self-tuned acceleration coefficient, the value of c_1 and c_2 [10] as follow:

$$1.49 \leq c_1 = c_2 < 2 \quad (16)$$

where:

$$c_1 = c_2 = c_{max} - [(c_{max} - c_{min}) * iteration]/max_{iteration} \quad (17)$$

Self-adaptive (SAAC). in self-tuned acceleration coefficient, the value of c_1 and c_2 as follow:

$$1.35 < c_1 = c_2 < 2.45 \quad (18)$$

where:

$$c_1 = c_{max} - [(c_{max} - c_{min}) * iteration]/max_{iteration} \quad (19)$$

$$c_2 = c_{max} + [(c_{max} - c_{min}) * iteration]/max_{iteration} \quad (20)$$

Fitness-based (FR) in fitness-based acceleration coefficient, the value of c_1 and c_2 as follow :

$$c = \begin{cases} c_1 = 4.5 - (1 - FR)^2 \\ c_2 = 0.5 - (1 - FR)^3 \end{cases} \quad (21)$$

where the 'FR' is the fitness rank, 4.5, 0.5 and 1 are the empirical constants.

Table 1 Summary method of the four parameters

Parameter	Method	Reference
Initial population	Zero-vector	Standard PSO, uses the zero-vector and has been used by [25]
	Frequent itemset	Our proposed
	Total high-utility transaction	Our proposed
Inertia weight	Static inertia weight	Standard PSO, uses 0.9 were proposed by [19] and has been used by [25].
	Increasing inertia weight	Our proposed based on [26].
	Adaptive inertia weight	Our proposed based on [31].
	Constant inertia weight with probabilistic velocity	Our proposed based on [34].
	Constriction factor	Our proposed based on [8].
Acceleration coefficient	Constant acceleration coefficient	Our proposed based on [19].
	Self-tuned	Our proposed based on [10].
	Self-adaptive acceleration coefficient	Our proposed based on [10]
	Fitness based	Our proposed based on [30]
Velocity clamping	V-max	BPSO standard, v-max = 4 as proposed by [11] and has been used by [25].
	V-max with piece wise linear function	Our proposed based on [11] and [34].

Table 2 Proposed configurations I and their characteristics

Id	Initial population	Inertia weight
ZS	Zero vectors	Static
FS	Frequent itemsets	Static
TS	Transactions of the highest total utility	Static
ZI	Zero vectors	Increasing
ZA	Zero vectors	Adaptive
ZP	Zero vectors	Constant with probabilistic velocity
ZC	Zero vectors	Constriction factor
FI	Frequent itemsets	Increasing
FA	Frequent itemsets	Adaptive
FP	Frequent itemsets	Constant with probabilistic velocity
FC	Frequent itemsets	Constriction factor
TI	Transactions of the highest total utility	Increasing
TA	Transactions of the highest total utility	Adaptive
TP	Transactions of the highest total utility	Constant with probabilistic velocity
TC	Transactions of the highest total utility	Constriction factor

Table 3 Proposed configurations II and their characteristics

Id	Inertia weight	Acceleration coefficient
SC	Static	Constant
ST	Static	Self-Tune
SA	Static	Self-Adaptive
SF	Static	Fitnes-based
AC	Adaptive	Constant
AT	Adaptive	Self-Tune
AA	Adaptive	Self-Adaptive
AF	Adaptive	Fitnes-based
PC	Probabilistic	Constant
PT	Probabilistic	Self-Tune
PA	Probabilistic	Self-Adaptive
PF	Probabilistic	Fitnes-based
IC	Increasing	Constant
IT	Increasing	Self-Tune
IA	Increasing	Self-Adaptive
IF	Increasing	Fitnes-based
CC	Constriction	Constant
CT	Constriction	Self-Tune
CA	Constriction	Self-Adaptive
CF	Constriction	Fitnes-based

4.4 Velocity clamping

For velocity clamping, we proposed to use the v[15] method [11] along a with piece-wise linear function [34], as follows:

$$v_i(t+1) = \begin{cases} v_{\max}, & \text{if } v_i(t+1) > v_{\max}, \\ v_i(t+1), & \text{if } |v_i(t+1)| \le v_{\max}, \\ -v_{\max}, & \text{if } v_i(t+1) < -v_{\max}. \end{cases} \quad (22)$$

For all experimented alternatives, the proposed method is fixed to use velocity clamping to maintain a "stable" position for each particle when searching for high utility itemsets with $v_{\max}$ is set to 4.

4.5 Putting them all together for tuning up

[16]

Table 1 shows a summary of the four parameters used in BPSO to obtain high-utility itemset, along with the method for each parameter and reference used. Furthermore, the experiment was carried out using the configuration of the parameters used as in the Table 2 and Table 3.

Out of the three parameters (initial population, inertia weight. and acceleration coefficient) and their respective

Table 4 Characteristics of the datasets used in the experiments

Dataset	Size	$\|I\|$	$\|D\|$	Avg	Total utility	Density
Chess	335	75	3,192	37.00	1,484,915	Dense
Mushroom	558	119	8,124	23.00	2,217,907	Dense
Connect	9,309	129	33	43.00	32,782,315	Dense
Accident	34,678	468	340,183	33.81	12,659,954	Dense
Foodmart	176	1,559	4,141	4.42	12,011,023	Sparse

Table 5 The minimum utility thresholds of each dataset

Dataset	Minimum utility threshold δ (%)				
Chess	25.00	25.50	26.00	26.50	27.00
Mushroom	14.00	14.25	14.50	14.75	15.00
Connect	28.90	29.10	29.30	29.50	29.70
Accident	13.10	13.40	13.70	14.00	14.30
Foodmart	0.11	0.12	0.13	0.14	0.15

possible values, we construct 15 configurations as shown in Table 2 for variouns initial population, inertia weight and acceleration coefficient is constant. We also construct 20 configuration as shown Table 3 for various configuration inertia weight, acceleration coefficient and initial population with zero vectors. Note that the previous method of BPSO-based high-utility itemset mining [25] used initial populations with zero vectors, static inertia weight and constant acceleration coefficient, namely ZS in the Table 2.

5 Experiments, result and analysis

5.1 Datasets

The datasets used are the same as those used in HUIM-BPSO-tree, namely chess, mushroom, connect, accident, and foodmart [25]. All datasets with internal and external utility can be downloaded from https://bit.ly/30otNXu. Characteristics of the datasets used in the experiments same as used in [15] as seen in Table 4.

5.2 Experimental setup

All algorithms are implemented in Java, executed on a PC with AMD Ryzen 7 2700 Eight-Core Processor, 3200 MHz, 8 cores, 16 logical processors and 16.0 GB of RAM, running 64-bit Windows 10. Implementation of the algorithms is based on SPMF, the open-source data mining library [13]. The number of iteration is 10,000 and population size is 20. The number of experiments for each configuration and each minimum utility threshold is five times. The average values of four measurements are taken from these five experiments, namely the number of itemsets, the total utility of those obtained itemsets, the computation time, and memory consumption.

The minimum utility threshold is set differently for each dataset, which we adopt from previous studies [23–25], and is specified in Table 5.

We have conducted experiments to observe and study:

1. the effects of initial population on the obtained high utility itemsets, by comparing the performance of existing method (ZS) with that of FS and TS,
2. the effects of inertia weight on the obtained high utility itemsets, by comparing the performance of ZS with that of ZI, ZA, ZP, and ZC,
3. the combined effects of initial population and inertia weight on the obtained high utility itemsets, by comparing the performance of FI, FA, FP, and FC with that of TI, TA, TP, and TC.
4. the combined effects of inertia weight and acceleration coefficients on obtained high utility itemset, by comparing the peformance of ZS/SC with all combination in Table 3.

5.3 Result and analysis

5.3.1 Effects of initial population

Two alternative methods for initializing the population are proposed, namely using frequent itemsets and transactions of the highest total utility. The number of transactions used as an initial population in both methods is the population size. In this part, the role of initial population setting on itself is studied; hence it is coupled only with static inertia weight.

The first row of Fig. 1 shows that the number of itemsets produced by the proposed methods FS and TS are almost always more, although not significantly, than ZS in almost all cases. ZS only produces more itemsets for accident dataset at minimum utility threshold 13.1 and 13.4. When FS compared with TS, the number of itemsets obtained by the two methods for three datasets, namely chess, mushroom and connect, are almost the same. Furthermore, FS produces more itemsets than TS for the accident dataset, while TS produces more itemsets for the foodmart dataset. Although not significantly, initializing population by frequent itemsets or transactions of the highest total utility, instead of just using zero vectors, increases the number of itemsets obtained. The second row of the same figure shows that total utility of the obtained itemsets follows similar patterns to the number of itemsets in almost all cases, which indicates that the three methods obtain itemsets of similar utilities. Table 6 shows the ranking of initial population based on the average number of itemsets. The best ranking or initial population that produces the highest number of itemsets is using frequent itemset.

The computation time and the memory consumption of the associated experiments on the three alternatives are shown in the third and the fourth rows of Fig. 1, respectively. Setting the initial population by the two proposed methods certainly has an impact on the computation time:

Fig. 1 Comparing ZS, FS, and TS to assess the effects of initial population on the high utility itemsets, showing the number of produced itemsets (first row), the total utility of those itemsets (second row), the computation time (third row), and the memory consumption (fourth row)

Table 6 Initial population ranking based on number of itemsets

Dataset	Ranking		
	ZS	FS	TS
Chess	3	1	2
Mushroom	3	1	2
Connect	3	1	2
Accident	1	2	3
Foodmart	3	2	1
Sum Ranking	13	7	10
Avg Ranking	3	1	2

FS and TS have a longer computation time compared to ZS. This is due to additional processing to set up the initial population, by first determining frequent itemsets or by first searching through the database for transactions of the highest total utility. However, the difference in computation times for all of them is not too long. Furthermore, computation time is almost not affected by the minimum utility threshold and tends to be stable for every minimum utility threshold specified. For memory consumption, in general, ZS requires more memory compared to FS and TS. When

Fig. 2 Comparing ZS, ZI, ZA, ZP, and ZC to assess the effects of inertia weight on the high utility itemsets, showing the number of produced itemsets (first row), the total utility of those itemsets (second row), the computation time (third row), and the memory consumption (fourth row)

FS compared with TS, TS consumes more memory than FS, but not significantly.

From the results of the four measurements, the proposed methods for setting initial population produce more and better high utility itemsets than the previous method, although the difference is not really noteworthy.

5.3.2 Effects of inertia weight

In this part, four alternative methods for setting inertia weight are proposed, namely increasing, adaptive, constant with probabilistic velocity, and with a constriction factor. These alternatives will be compared with existing method, namely static. Since the focus is the role of inertia weight

Table 7 Inertia weight rangking based on number of itemsets

Dataset	Ranking				
	ZS	ZI	ZA	ZP	ZC
Chess	5	3	1	4	2
Mushroom	1	1	1	1	1
Connect	4	3	1	5	2
Accident	4	5	1	3	2
Foodmart	5	2	4	3	1
Sum ranking	19	14	8	16	8
Avg ranking	4	3	2	3	2

setting on itself, all alternatives are coupled only with initial population by zero vectors.

The number of itemsets obtained by the five alternatives is depicted in the first row of Fig. 2. The proposed methods for setting inertia weight (ZI, ZA, ZP, and ZC) produce more itemsets than the existing method ZS in almost all cases. In chess and foodmart datasets, they produce more itemsets for all minimum utility thresholds, while in other datasets, they produce more itemsets but not for all minimum utility thresholds. ZP method in connect dataset, however, produces less itemsets than the existing method.

ZA produces the most itemsets of the others, and the second is ZC. It seems that the use of non-static inertia weight is a commanding factor in producing more itemsets. ZA uses adaptive inertia weight, while ZC uses constriction factor values. Likewise, ZI uses incremental inertia weight, although the results are not as good as ZA or ZC. ZP, which uses static inertia weight, the number of itemsets is less than the others, even less than ZS for connect dataset. Table 7 shows the ranking of inertia weight based on the average number of itemsets. The best ranking or inertia weight parameter that produces the highest number of itemsets is using adaptive inertia weight.

The second row of Fig. 2 also indicates that total utility of the obtained itemsets does not differ much in pattern compared to that of the number of itemsets in almost all cases. This suggests that all alternative methods obtain itemsets of similar utilities for the different datasets or different minimal utility thresholds.

The computation times needed to obtain the high utility itemsets are shown in the third row of Fig. 2. ZP is the fastest among the alternatives for chess, connect, accident, and foodmart datasets, while ZI is the slowest for chess, connect, accident, and foodmart datasets. However, the computation times of all methods are not too different, because their computation does not involve increasing the number of iterations, but only makes changes to the associated parameters when updating velocity, especially inertia weight, which only affects the way particles is updated. The computation times are also not affected by the specified minimum utility threshold; for each minimum utility threshold, the computation is stable. Computation times needed to obtain itemsets with BPSO are mostly affected by the number of iterations and the population size.

The memory consumption required to obtain the high utility itemsets can be seen in the fourth row of Fig. 2. In general, memory consumption of ZS is more than the proposed methods. Among the proposed methods themselves, ZC tends to need more memory than others, except in chess dataset. However, the use of memory of the proposed methods cannot be easily generalized. The experimental results show that the required memory varies almost with no patterns for each dataset and each minimum utility threshold.

5.3.3 Effects of acceleration coefficients

In general, the main factor to get more itemsets is the use of inertia weight, but acceleration coefficients (individual and social factors) also have an influence. [25] uses acceleration coefficients; both individual and social factors are set the same, namely 2, while when using the constriction factor, the total individual and social factors is 4 [8].

We conduct an experiment to look into the effects of the acceleration coefficients, as in Table 3. The value of acceleration coefficients varies, not only being constant. However, it depends on the number of iterations, the maximum and minimum values of acceleration coefficients, and the value of the fitness range obtained from each iteration.

In general, the use of acceleration coefficients as in Table 3 produces a larger number of itemsets compared to the ZS method. In this section, we will show a comparison of 4 methods of acceleration coefficients using inertia weight static and initial population zero vectors. The SC, ST, SA, SF methods will be compared to existing methods, namely ZS, where ZS does not use acceleration coefficients.

The number of itemsets obtained by the five alternatives is depicted in the first row of Fig. 3. The proposed methods for setting acceleration coefficients (SC, ST, SA, and SF) produce more itemsets than the existing method ZS in almost all cases. Only in mushroom dataset, they produce almost same for all minimum utility datasets, while in other datasets, they produce more itemsets for all minimum utility thresholds. The second row of Fig. 3 indicates that total utility obtained, it the same pattern as the number of itemsets in almost all cases. Table 8 shows the ranking of acceleration coefficient based on the average number of itemsets. The best ranking or acceleration coefficient parameter that produces the highest number of itemsets is using self-adaptive acceleration coefficient (SAAC).

The third and fourth row of Fig. 3 show the running times and memory consumption to get high-utility itemsets. As the

Fig. 3 Comparing ZS, SC, ST, SA and SF to assess the effects of acceleration coefficients on the high utility itemsets, showing the number of produced itemsets (first row), the total utility of those itemsets (second row), the computation time (third row), and the memory consumption (fourth row)

same with inertia weights, computation times are also not affected by the specified minimum utility threshold; for each minimum utility threshold, the computation is stable. And in our experiment, memory consumption from our proposed methods less than ZS.

From the experiments conducted, we can conclude that the acceleration coefficients affect the number of itemsets obtained.

5.3.4 Effects of velocity clamping

Table 9, shows a representative result demonstrating the effects of velocity clamping changes for various minimum utility thresholds. The chess dataset and ZC method again are selected to represent the effect of changes in velocity clamping. Acceleration coefficients, both individual and social factors, are set the same, namely 2 [25]. The maximum

Table 8 Acceleration coefficient rangking based on number of itemsets

Dataset	Ranking				
	ZS	SC	ST	SA	SF
Chess	5	3	4	1	2
Mushroom	1	1	1	1	1
Connect	5	2	3	1	4
Accident	5	1	3	2	4
Foodmart	5	4	2	1	3
Sum ranking	21	11	13	6	14
Avg ranking	4	2	3	1	3

Table 9 The number of itemsets obtained for various velocity clamping settings

Minimum utility threshold	Velocity clamping (v_{max})				
(%)	0	2	3	**4**	5
25.0	189	203	430	**458**	441
25.5	128	127	284	**314**	272
26.0	83	78	169	**194**	181
26.5	47	46	96	**104**	100
27.0	33	27	57	**59**	54

velocity clamping v_{max} used starts from 0, meaning that it does not use velocity clamping, up to 5. While the minimum velocity clamping value is $-v_{max}$. The number of itemsets obtained for each minimum utility threshold increases when v_{max} increases from 2 to 4, and then the number decreases when the value of v_{max} is more than 4. In Table 9, the best velocity clamping is shown in bold. From experiments in the chess dataset, we conclude that the best velocity clamping is $[-4, 4]$; and this setting will be used for other experiments. From the experiments, we can see that the velocity clamping also affects the number of itemsets obtained.

5.3.5 Combined effects of initial populations, inertia weight and acceleration coefficients

Initial populations did not have a significant effect on producing high-utility itemset, however, when combined with inertia weight or constriction factor, there was an increase in the number of itemset obtained. We attempt to find the optimal configuration when the two parameters are combined and set at the same time. Eight methods exist, namely all possible configurations obtained from two ways of setting initial population and four ways of setting inertia weight. The configuration is as shown in Table 2, namely the configuration for FI, FA, FP, FC, TI, TA, TP, and TC. The focus

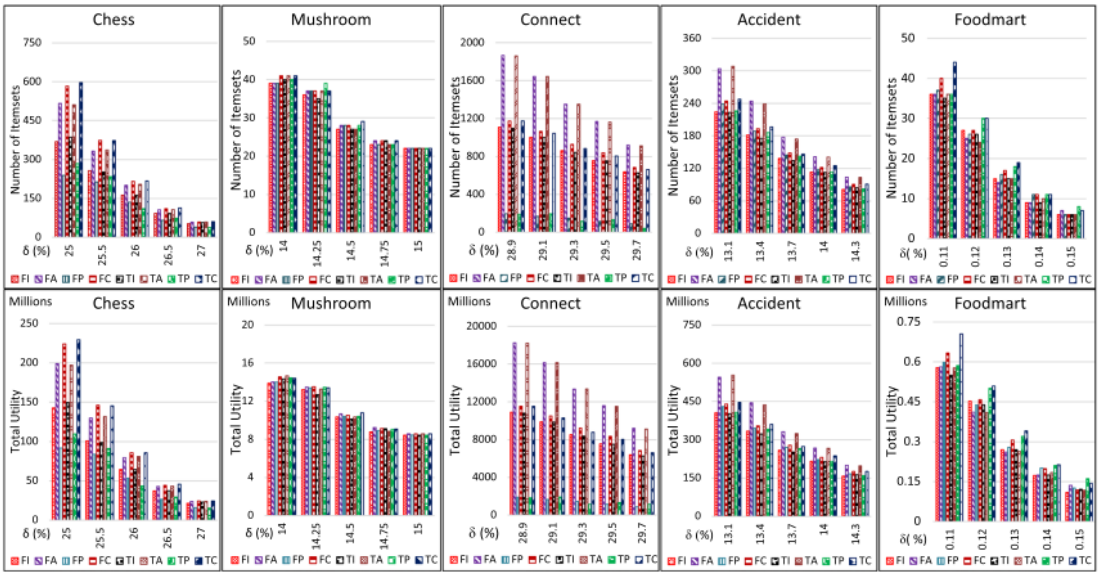

Fig. 4 Comparing FI, FA, FP, FC, TI, TA, TP, and TC to assess the combined effects of initial population and inertia weight on the high utility itemsets, showing the number of produced itemsets (first row), and the total utility of those itemsets (second row)

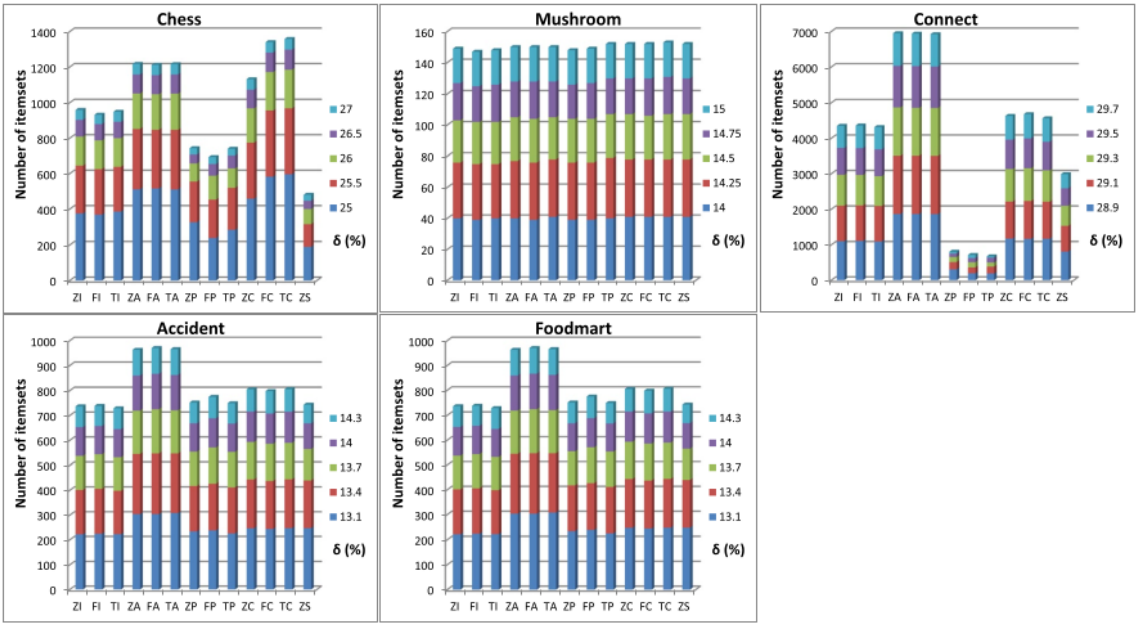

Fig. 5 Showing number of itemsets: comparing ZI, FI, TI, ZA, FA, TA, ZP, FP, TP, ZC, FC, TC, and ZS

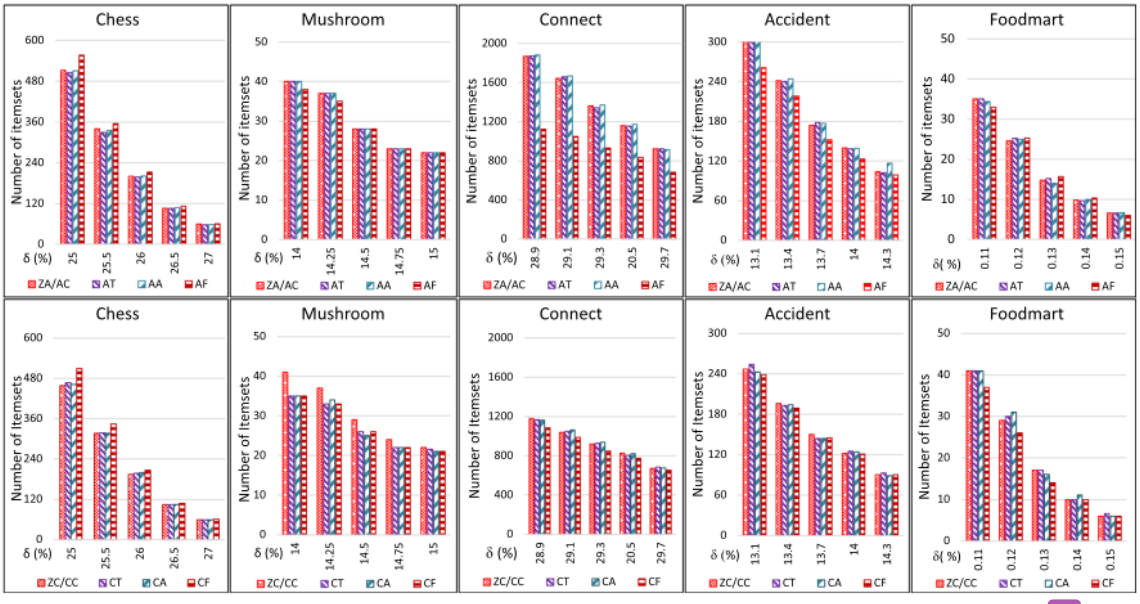

Fig. 6 Comparing ZA/AC, AT, AA, and AF to assess the combined effects of adaptive inertia weight and acceleration coefficient on the high utility itemsets, showing the number of produced itemsets (first row), comparing ZC/CC, CT, CA, and CF to assess the combined effects of constriction factor and acceleration coefficient on the high utility itemsets, showing the number of produced itemsets (second row)

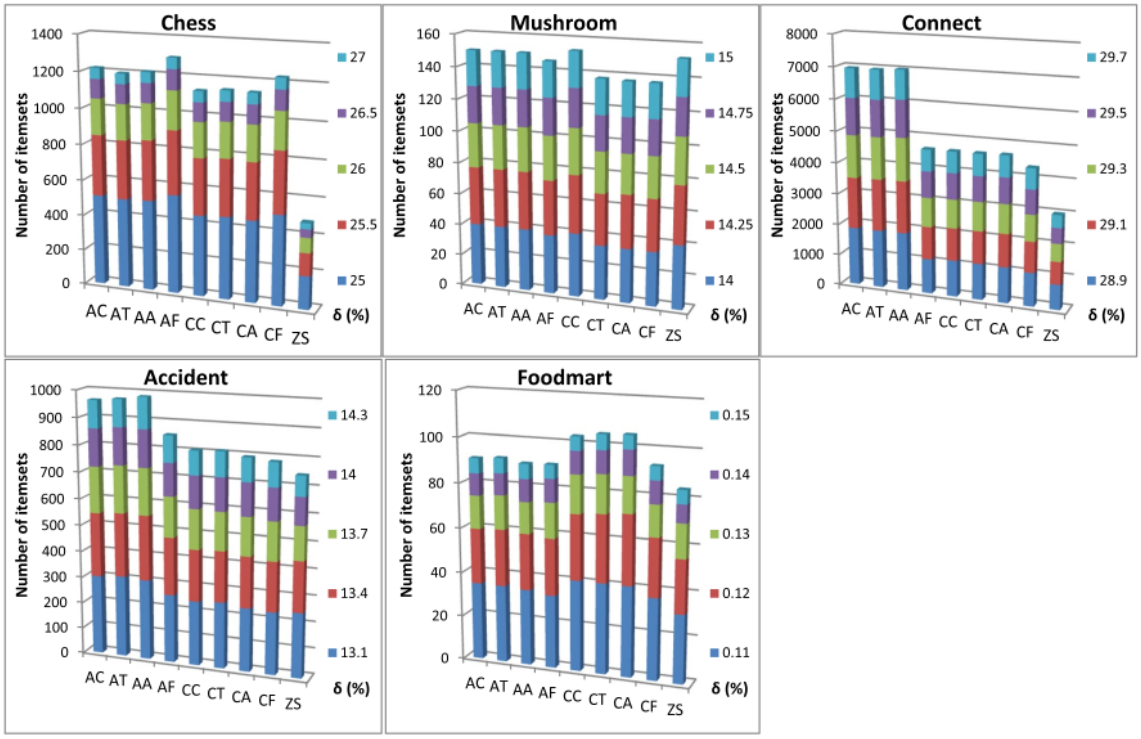

Fig. 7 Showing number of itemsets: comparing AC, AT, AA, AF, CC, CT, CA, CF, and ZS as combining adaptive inertia weight (A), constriction factor (C) and acceleration coefficient

of the comparisons is on the number of itemset and the total utility obtained.

The first row of Fig. 4 shows the number of itemsets obtained by these alternative methods. The proposed methods always produce more itemsets than the existing method ZS, except for FP and TP in connect dataset. The number of itemsets obtained in this scenario is almost identical to the previous scenario (inertia weight). When using proposed inertia weight setting produces more itemsets than existing method, the combination with the initial population settings, both frequent itemsets and transactions of the highest total utility, also produces more itemsets, and vice versa. Similar results can also be observed for the total utility of those obtained itemsets in the second row of Fig. 4. Overall this indicates that the contribution of the inertia weight setting to the number and quality of the obtained itemsets is much more significant than that of the initial population setting.

Fig. 5 shows a clearer picture of the total number of itemsets obtained from the combination of the initial population (zero-vector (Z), frequent itemset (F), and total high-utility transaction (T)) and the inertia weight (Increasing(I), Adaptive(A), Probabilistic (P), and Constriction (C)). Compared to existing method, ZS, this combination produces a higher number of itemsets except FP and TP for connect dataset. Fig. 5 also shows that there is no significant difference between using zero-vector, frequent itemset, and total high-utility transaction as the initial population.

Table 10 The two best configurations inertia weight for each dataset

Dataset	First	(% inc.)	Second	(% inc.)
Chess	TC	157.93	FC	154.48
Mushroom	TC	1.58	FC	0.87
Connect	FA	132.67	TA	131.93
Accident	FA	33.24	TA	32.38
Foodmart	TC	29.55	TP	26.39

The first row of Fig. 6 shows the adaptive inertia weight combined with various acceleration coefficients. Of the four combinations of the acceleration coefficient, the number of itemset obtained was more than the ZS. When compared between the four acceleration coefficient methods using adaptive inertia weight, for the chess dataset, variations in

Table 11 The two best configurations acceleration coefficient for each dataset

Dataset	First	(% inc.)	Second	(% inc.)
Chess	AF	149.69	CF	140.18
Mushroom	PF	4.62	PT	2.86
Connect	AA	134.39	AC	133.01
Accident	AA	42.69	AC	35.23
Foodmart	CA	21.12	CT	21.12

fitness range and adaptive acceleration give the best results; for the connect and accident datasets, adaptive acceleration gives the best results; while for the mushroom and foodmart datasets the four acceleration methods give the number itemset which is not much different.

The second row of Fig. 6 shows the number of itemset obtained from the combination of constriction factor values and various acceleration coefficients. The four combinations provide a greater number of itemsets compared to ZS. When compared between the four, for the chess dataset, the fitness range acceleration configuration gives the best results; for the mushroom dataset constant acceleration coefficient gives the best results; the connect, accident and foodmart datasets provide almost the same number of itemsets except for the fitness range acceleration coefficient.

When compared between the combination of the use of adaptive inertia weight and constriction factor value with various combinations of acceleration coefficients, in general the combination of adaptive inertia weight produces a larger number of itemsets, except for the foodmart dataset. In general, the best combination is in adaptive inertia weight with self-adaptive acceleration coefficent, except for the chess dataset, the best combination is on adaptive inertia weight and fitness range.

Fig. 7 shows a clearer picture of the total number of itemsets obtained from the combination of the two best inertia weight (adaptive inertia weight (A) and constriction factor (C)) and acceleration coefficient (constant (C), self-tuned(T), self-adaptive acceleration coefficient (A), and fitness based (F)). Compared to existing method, ZS, this combination produces a higher number of itemsets. Fig. 7 also shows that the best combination is adaptive inertia weight with self-adaptive acceleration coefficient for chess, connect, and accident dataset. For chess dataset the best combination is adaptive inertia weight and fitness based. And only foodmart dataset the best combination is using constriction factor inertia weight.

5.4 Discussion

The two best combinations of the eight proposed combinations (from Fig. 3) for each dataset compared to the previous method can be seen in Table 10. The percentage increase in the number of itemsets is calculated from the percentage increase in the number of itemsets compared to the number of itemsets obtained by the previous method for each minimum utility threshold. From Table 10 we can observe that FA, TA, TC, and FC are the best combinations that appear most often. This shows that the combinations of either initial population settings with adaptive inertia weight or constriction factor are the best combinations.

While the two best combinations obtained from the acceleration coefficient configuration in Table 3 is shown in Table 11, we can observe that self-adaptive acceleration coefficient are the best combinations that appear most often. This shows that the combinations with adaptive inertia weight or constriction factor are the best combinations.

Our experiments have also shown that:

1. Improved performance of binary-PSO to search for high utility itemsets can be achieved through similar approaches to improving the performance of PSO in general. The entire proposed methods for tuning the inertia weight have previously been applied to PSO. Almost all methods applied, especially in the process of updating particles with various inertia weight formulas and various acceleration coefficients can produce good results for the search of high utility itemsets. Adaptive inertia weight, besides producing good results on PSO [31], has also been shown to have good performance in producing high utility itemsets.
2. The choice of the values of the acceleration coefficients and velocity clamping together with piece-wise linear functions is something that needs attention. The combination of acceleration coefficient, both individual and social factors, is very likely.

   The acceleration coefficient value does not always have to be constant. Experiments that have been conducted have shown that the accelaration coefficient value can use various formulas. Self-adaptive and fitness range acceleration coefficient can be used as an alternative to using the acceleration coefficient, but even though it produces more itemset, it still needs to be proven that the best acceleration coefficient configuration is in accordance with the type of the dataset.
3. Velocity clamping for various cases will be different. In the case of binary-PSO for high-utility itemset mining, the best value is in $[-4, 4]$, but the exact formula needs to be considered. Although our proposed method still uses standard values as in previous studies for both, an increased number of high utility itemsets has been achieved.

6 Conclusion

From our experiments, it can be seen that the effect of inertia weights is very significant in producing high utility itemsets. When combined initial population settings, either with frequent itemsets or with transactions of the highest total utility, inertia weight settings produce more itemsets than using zero vector as initial population. Beside inertia weight, the number of the itemsets is influenced by the proper setting of BPSO parameters, especially acceleration coefficients and velocity clamping. The use of acceleration coefficients can affect the number of itemsets obtained. However, initial population does not influence much on the number of itemsets obtained.

The combination inertia weight with acceleration coefficients can be used to get more itemsets. In real conditions, the higher the number of itemsets means that the total utility will increase, thus increasing profits. The proposed methods with combination of both settings can be used as alternatives when the acquisition of profit becomes the primary goal of high-utility itemset search. The best combinations are adaptive inertia weight and constriction factor with self-adaptive acceleration coefficient. Computation speed and memory consumption can be ignored because they have almost the same insignificant effect.

For the future, a combination of other parameters can be tried to increase the number of high utility itemsets obtained. Other initial population settings that are more appropriate or other inertia weight formulas can be devised, in addition to searching for better formulas for acceleration coefficients and velocity clamping. Besides BPSO approach, obtaining high-utility itemsets can also be tested using various methods of swarm intelligence or meta-heuristic such as: Harmony Search (HS) Algorithm, Krill Herd Algorithm (KHA), Cuckoo Search (CS) Algorithm, Gray Wolf Optimizer (GWO), Arithmetic Optimization Algorithm (AOA) and others.

References

1. Abualigah L, Gandomi AH, Elaziz MA, Hussien AG, Khasawneh AM, Alshinwan M, Houssein EH (2020) Nature-inspired optimization algorithms for text document clustering–a comprehensive analysis. Algorithms 13(12):345. https://doi.org/10.3390/a13120345
2. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Method Appl Mech Eng. https://doi.org/10.1016/j.cma.2020.113609
3. Abualigah L, Gandomi AH, Elaziz MA, Hamad HA, Omari M, Alshinwan M, Khasawneh AM (2021) Advances in meta-heuristic optimization algorithms in big data text clustering. Electronics 10(2):101. https://doi.org/10.3390/electronics10020101
4. Abualigah LM, Khader AT, Hanandeh ES (2018) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25(September):456–466. https://doi.org/10.1016/j.jocs.2017.07.018
5. Abualigah LMQ (2019) Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering, Studies in Computational Intelligence, vol 816. Springer International Publishing, Cham, https://doi.org/10.1007/978-3-030-10674-4
6. Alhussein M, Haider SI (2015) Improved particle swarm optimization based on velocity clamping and particle penalization. In: 2015 3rd international conference on artificial intelligence, modelling and simulation (AIMS), pp 61–64, https://doi.org/10.1109/AIMS.2015.20
7. Banerjee C, Sawal R (2014) PSO with dynamic acceleration coefficient based on multiple constraint satisfaction: implementing fuzzy inference system. In: 2014 International conference on advances in electronics computers and communications, IEEE Explore, Bangalore, India, pp 1–5, https://doi.org/10.1109/ICAECC.2014.7002381
8. Clerc M (1999) The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. Proceedings of the 1999 congress on evolutionary computation, CEC 1999 3:1951–1957
9. Dong C, Chen Z, Shilei S (2013) The acceleration coefficients self-adapting in PSO. Int J Digit Content Technol Appl 7(5):672–678. https://doi.org/10.4156/jdcta.vol7.issue5.79
10. Drahansky M, Paridah M, Moradbak A, Mohamed A, Owolabi FAT, Asniza M, Abdul Khalid SH (2018) Performance comparison of PSO and its new variants in the context of VLSI global routing, particle swarm optimization with applications. Pakize Erdoğmuş, IntechOpen i:13, https://doi.org/10.5772/intechopen.72811, https://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics
11. Eberhart R, Kennedy J (1998) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science (MHS'95), pp 39–43
12. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the 2000 congress on evolutionary computation, CEC 2000 1(February 2000):84–88, https://doi.org/10.1109/CEC.2000.870279
13. Fournier-Viger P, Lin JCW, Gomariz A, Gueniche T, Soltani A, Deng Z, Lam HT (2016a) The SPMF open-source data mining library version 2. In: Berendt B, Bringmann B, Fromont É, Garriga G, Miettinen P, Tatti N, Tresp V (eds) European conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III, Springer International Publishing, pp 36–40
14. Fournier-Viger P, Lin JCW, Wu CW, Tseng VS, Faghihi U (2016b) Mining minimal high-utility itemsets. In: Hartmann S, Ma H (eds) International conference on database and expert systems applications, Springer International Publishing, pp 88–101
15. Gunawan R, Winarko E, Pulungan R (2020) A BPSO-based method for high-utility itemset mining without minimum utility threshold. Know-Based Syst. https://doi.org/10.1016/j.knosys.2019.105164
16. Jakubcová M, Máca P, Pech P (2014) A comparison of selected modifications of the particle swarm optimization algorithm. J Appl Math 2014(August):1–10. https://doi.org/10.1155/2014/293087
17. Kannimuthu S, Premalatha K (2013) Discovery of high utility itemsets using genetic algorithm. Int J Eng Technol 5(6):4866–4880
18. Kazimipour B, Li X, Qin AK (2014) A review of population initialization techniques for evolutionary algorithms. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2585–2592, https://doi.org/10.1109/CEC.2014.6900618

19. Kennedy J, Eberhart R (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol 5, pp 4–8
20. Khanesar MA, Teshnehlab M, Shoorehdeli MA (2007) A novel binary particle swarm optimization. In: 2007 Mediterranean conference on control automation, pp 1–6, https://doi.org/10.1109/MED.2007.4433821
21. Kuo R, Chao C, Chiu Y (2011) Application of particle swarm optimization to association rule mining. Appl Soft Comput 11(1):326–336
22. Lee S, Soak S, Oh S, Pedrycz W, Jeon M (2008) Modified binary particle swarm optimization. Prog Natural Sci 18(9):1161–1166. https://doi.org/10.1016/j.pnsc.2008.03.018
23. Lin JCW, Yang L, Fournier-Viger P, Frnda J, Sevcik L, Voznak M (2015) An evolutionary algorithm to mine high-utility itemsets. Adv Electric Electron Eng 13(4):392–398
24. Lin JCW, Yang L, Fournier-Viger P, Wu JMT, Hong TP, Wang LSL, Zhan J (2016) Mining high-utility itemsets based on particle swarm optimization. Eng Appl Artif Intell 55:320–330
25. Lin JCW, Yang L, Fournier-Viger P, Hong TP, Voznak M (2017) A binary PSO approach to mine high-utility itemsets. Soft Comput 21(17):5103–5121
26. Liu J, Mei Y, Li X (2016) An analysis of the inertia weight parameter for binary particle swarm optimization. IEEE Trans Evolu Comput 20(5):666–681. https://doi.org/10.1109/TEVC.2015.2503422
27. Liu M, Qu J (2012) Mining high utility itemsets without candidate generation categories and subject descriptors. In: Proceedings of the 21st ACM international conference on information and knowledge management (CIKM), pp 55–64
28. Liu Y, Liao Wk, Choudhary A (2005) A two-phase algorithm for fast discovery of high utility itemsets. In: Proceeding PAKDD'05 Proceedings of the 9th Pacific-Asia conference on advances in knowledge discovery and data mining, Springer-Verlag Berlin, Heidelberg, pp 689–695
29. Ma G, Gong R, Li Q, Yao G (2016) A improved particle swarm optimization algorithm with dynamic acceleration coefficients. Bull Elect Eng Info 5(4):489–494. https://doi.org/10.11591/eei.v5i4.561
30. Mehmood Y, Sadiq M, Shahzad W, Amin F (2019) Fitness-based acceleration coefficients to enhance the convergence speed of novel binary particle swarm optimization. In: Proceedings - 2018 international conference on frontiers of information technology, FIT 2018, IEEE, December, pp 355–360, https://doi.org/10.1109/FIT.2018.00069
31. Nickabadi A, Ebadzadeh MM, Safabakhsh R (2011) A novel particle swarm optimization algorithm with adaptive inertia weight. Appl Soft Comput J 11(4):3658–3670. https://doi.org/10.1016/j.asoc.2011.01.037
32. Richards M, Ventura D (2004) Choosing a starting configuration for particle swarm optimization. In: 2004 IEEE International joint conference on neural networks (IEEE Cat.No.04CH37541), IEEE Explore, vol 3, pp 2309–2312, https://doi.org/10.1109/IJCNN.2004.1380986
33. Tang Z, Zhang D (2009) A modified particle swarm optimization with an adaptive acceleration coefficient. Proceedings - 2009 Asia-Pacific conference on information processing, APCIP 2009 2:330–332, https://doi.org/10.1109/APCIP.2009.217
34. Tasgetiren MF, Liang YC (2004) A binary particle swarm optimization algorithm for lot sizing problem. J Econ Social Res 5(2):1–20
35. Wu JMT, Zhan J, Lin JCW (2016) An ACO-based approach to mine high-utility itemsets. Know-Based Syst 116:102–113
36. Wu Z, Zhou J (2007) A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment. Proceedings - 2007 International conference on computational intelligence and security, CIS 2007 pp 133–136, https://doi.org/10.1109/CIS.2007.68
37. Yao H, Hamilton HJ (2006) Mining itemset utilities from transaction databases. Data Know Eng 59:603–626
38. Yao H, Hamilton H, Butz C (2004) A foundational approach to mining itemset utilities from databases. In: Proceedings of the 2004 Society for industrial and applied mathematics (SIAM) international conference on data mining, SIAM, Lake Buana Vista, pp 482–486
39. Zhang C, Almpanidis G, Wang W, Liu C (2018) An empirical evaluation of high utility itemset mining algorithms. Expert Syst Appl 101:91–115
40. Zida S, Fournier-Viger P, Lin JCW, Wu CW, Tseng VS (2015) EFIM: A highly efficient algorithm for high-utility itemset mining. Lecture Notes in computer science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9413:530–546
41. Zida S, Fournier-Viger P, Lin JCW, Wu CW, Tseng VS (2017) EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Know Info Syst 51(2):595–625

Performance comparison of inertia weight and acceleration coefficients of BPSO in the context of high-utility itemset mining

ORIGINALITY REPORT

15%	6%	14%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"High-Utility Pattern Mining", Springer Science and Business Media LLC, 2019 Publication	1%
2	Jianhua Liu, Yi Mei, Xiaodong Li. "An Analysis of the Inertia Weight Parameter for Binary Particle Swarm Optimization", IEEE Transactions on Evolutionary Computation, 2016 Publication	1%
3	Ridowati Gunawan. "Online Retail Pattern Quality Improvement: From Frequent Sequential Pattern to High-Utility Sequential Pattern", 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2021 Publication	1%
4	Yasir Mehmood, Marium Sadiq, Waseem Shahzad, Faryal Amin. "Fitness-Based Acceleration Coefficients to Enhance the	1%

	Convergence Speed of Novel Binary Particle Swarm Optimization", 2018 International Conference on Frontiers of Information Technology (FIT), 2018 Publication	
5	Submitted to Wright State University Student Paper	<1 %
6	Yan Chen, Aijun An. "Approximate Parallel High Utility Itemset Mining", Big Data Research, 2016 Publication	<1 %
7	www.philippe-fournier-viger.com Internet Source	<1 %
8	epdf.pub Internet Source	<1 %
9	"IMPROVED VARIABLE-LENGTH PARTICLE SWARM OPTIMIZATION FOR STRUCTURE-ADJUSTABLE EXTREME LEARNING MACHINE", Control and Intelligent Systems, 2014. Publication	<1 %
10	mafiadoc.com Internet Source	<1 %
11	"Database and Expert Systems Applications", Springer Science and Business Media LLC, 2019 Publication	<1 %

12	philippe-fournier-viger.com Internet Source	<1%
13	Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh, Reza Safabakhsh. "A novel particle swarm optimization algorithm with adaptive inertia weight", Applied Soft Computing, 2011 Publication	<1%
14	dipot.ulb.ac.be Internet Source	<1%
15	Zhou, Yu Ting, Jian Bin Nie, Ning Han, Chen Chen, and Zi Feng Yue. "Study on PID Parameters Tuning Based on Particle Swarm Optimization", Advanced Materials Research, 2013. Publication	<1%
16	Lecture Notes in Computer Science, 2016. Publication	<1%
17	e-hir.org Internet Source	<1%
18	www.hindawi.com Internet Source	<1%
19	Gutha Jaya Krishna, Vadlamani Ravi. "High utility itemset mining using binary differential evolution: An application to customer	<1%

segmentation", Expert Systems with Applications, 2021
Publication

20	Jianzhong Zhou. "A Self-Adaptive Particle Swarm Optimization Algorithm with Individual Coefficients Adjustment", 2007 International Conference on Computational Intelligence and Security (CIS 2007), 12/2007 Publication	<1 %
21	Mohammad Karim Sohrabi. "An efficient projection-based method for high utility itemset mining using a novel pruning approach on the utility matrix", Knowledge and Information Systems, 2020 Publication	<1 %
22	coek.info Internet Source	<1 %
23	researchspace.ukzn.ac.za Internet Source	<1 %
24	"Intelligent Information and Database Systems", Springer Science and Business Media LLC, 2021 Publication	<1 %
25	Krishan Kumar Sethi, Dharavath Ramesh. "A fast high average-utility itemset mining with efficient tighter upper bounds and novel list	<1 %

	structure", The Journal of Supercomputing, 2020 Publication	
26	Lecture Notes in Computer Science, 2014. Publication	<1 %
27	Submitted to SASTRA University Student Paper	<1 %
28	T. K. Sethuramalingam, B. Nagaraj. "Design model on ship trajectory control using particle swarm optimisation", 2015 Online International Conference on Green Engineering and Technologies (IC-GET), 2015 Publication	<1 %
29	www.tkl.iis.u-tokyo.ac.jp Internet Source	<1 %
30	"Advanced Data Mining and Applications", Springer Science and Business Media LLC, 2014 Publication	<1 %
31	"Advanced Data Mining and Applications", Springer Science and Business Media LLC, 2019 Publication	<1 %
32	"Advances in Network-Based Information Systems", Springer Science and Business Media LLC, 2018 Publication	<1 %

33	Jimmy Ming-Tai Wu, Justin Zhan, Sanket Chobe. "Mining Association rules for Low-Frequency itemsets", PLOS ONE, 2018 Publication	<1 %
34	van den Bergh, F.. "A study of particle swarm optimization particle trajectories", Information Sciences, 20060422 Publication	<1 %
35	"Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices", Springer Science and Business Media LLC, 2020 Publication	<1 %
36	Bahareh Rahmati, Mohammad Karim Sohrabi. "A Systematic Survey on High Utility Itemset Mining", International Journal of Information Technology & Decision Making, 2019 Publication	<1 %
37	Lecture Notes in Computer Science, 2015. Publication	<1 %
38	Nan Qiao, Lihui Wang, Mingjie Liu. "An autonomous driving controller with heading adaptive calibration for agricultural robots", Industrial Robot: the international journal of robotics research and application, 2020 Publication	<1 %
39	d-nb.info Internet Source	<1 %

40	dspace.vsb.cz Internet Source	<1 %
41	journals.plos.org Internet Source	<1 %
42	opus.lib.uts.edu.au Internet Source	<1 %
43	Jerry Chun-Wei Lin, Wensheng Gan, Philippe Fournier-Viger, Lu Yang, Qiankun Liu, Jaroslav Frnda, Lukas Sevcik, Miroslav Voznak. "High utility-itemset mining and privacy-preserving utility mining", Perspectives in Science, 2016 Publication	<1 %
44	Chun, Semin, Young-Tark Kim, and Tae-Hyoung Kim. "A Diversity-Enhanced Constrained Particle Swarm Optimizer for Mixed Integer-Discrete-Continuous Engineering Design Problems", Advances in Mechanical Engineering, 2013. Publication	<1 %
45	Tjiong, Anthony S. J., and Sildomar T. Monteiro. "Feature selection with PSO and kernel methods for hyperspectral classification", 2011 IEEE Congress of Evolutionary Computation (CEC), 2011. Publication	<1 %
46	Submitted to Vels University Student Paper	<1 %

47	research-repository.griffith.edu.au Internet Source	<1%
48	www.statmt.org Internet Source	<1%
49	"Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology – ISAT 2018", Springer Science and Business Media LLC, 2019 Publication	<1%
50	Chu, C.J.. "An efficient algorithm for mining temporal high utility itemsets from data streams", The Journal of Systems & Software, 200807 Publication	<1%
51	S. Kannimuthu, K. Premalatha. "Discovery of High Utility Itemsets Using Genetic Algorithm with Ranked Mutation", Applied Artificial Intelligence, 2014 Publication	<1%
52	Tzung-Pei Hong, Cho-Han Lee, Shyue-Liang Wang. "Effective utility mining with the measure of average utility", Expert Systems with Applications, 2011 Publication	<1%

53	Volkan Soner Özsoy. "The determination of the most suitable inertia weight strategy for particle swarm optimization via the minimax mixed-integer linear programming model", Engineering Computations, 2021 Publication	<1 %
54	www.cs.uoi.gr Internet Source	<1 %
55	www.scilit.net Internet Source	<1 %
56	"Advances in Knowledge Discovery and Data Mining", Springer Science and Business Media LLC, 2018 Publication	<1 %
57	"Computational Collective Intelligence", Springer Science and Business Media LLC, 2017 Publication	<1 %
58	"Data Science and Big Data: An Environment of Computational Intelligence", Springer Science and Business Media LLC, 2017 Publication	<1 %
59	"Implementation of Bio-Inspired Algorithms in High Utility Itemset Mining", International Journal of Engineering and Advanced Technology, 2019 Publication	<1 %

60	Anindita Septiarini, Agus Harjoko, Reza Pulungan, Retno Ekantini. "Optic disc and cup segmentation by automatic thresholding with morphological operation for glaucoma evaluation", Signal, Image and Video Processing, 2016 Publication	<1 %
61	Jerry Chun-Wei Lin, Shifeng Ren, Philippe Fournier-Viger, Jeng-Shyan Pan, Tzung-Pei Hong. "Efficiently updating the discovered high average-utility itemsets with transaction insertion", Engineering Applications of Artificial Intelligence, 2018 Publication	<1 %
62	Jimmy Ming-Tai Wu, Justin Zhan, Jerry Chun-Wei Lin. "An ACO-based approach to mine high-utility itemsets", Knowledge-Based Systems, 2017 Publication	<1 %
63	Tseng, Vincent S., Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu. "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases", IEEE Transactions on Knowledge and Data Engineering, 2012. Publication	<1 %
64	Anita Bai, Parag S. Deshpande, Meera Dhabu. "Selective Database Projections Based	<1 %

Approach for Mining High-Utility Itemsets", IEEE Access, 2018
Publication

---

65 B Anup Bhat, S V Harish, M Geetha. "A Dynamic Itemset Counting Based Two-Phase Algorithm for Mining High Utility Itemsets", 2018 15th IEEE India Council International Conference (INDICON), 2018
Publication

<1%

---

66 Ivan Vlašić, Marko Đurasević, Domagoj Jakobović. "Improving genetic algorithm performance by population initialisation with dispatching rules", Computers & Industrial Engineering, 2019
Publication

<1%

---

67 Jerry Chun-Wei Lin, Tzung-Pei Hong, Philippe Fournier-Viger, Qiankun Liu, Jia-Wei Wong, Justin Zhan. "Efficient hiding of confidential high-utility itemsets with minimal side effects", Journal of Experimental & Theoretical Artificial Intelligence, 2017
Publication

<1%

---

68 Mohammed Elbes, Shadi Alzubi, Tarek Kanan, Ala Al-Fuqaha, Bilal Hawashin. "A survey on particle swarm optimization with emphasis on engineering and network applications", Evolutionary Intelligence, 2019
Publication

<1%

69	S.N. Sivanandam. "Dynamic task scheduling with load balancing using parallel orthogonal particle swarm optimisation", International Journal of Bio-Inspired Computation, 2009 Publication	<1 %
70	Sang-Shin Byun, Kyeong Ok Kim. "A Study on the Impacts of the 1741 Tsunami Recorded in the Annals of Joseon Dynasty", Journal of Korean Society of Coastal and Ocean Engineers, 2021 Publication	<1 %
71	Youcef Djenouri, Jerry Chun-Wei Lin, Kjetil Nørvåg, Heri Ramampiaro, Philip S. Yu. "Exploring Decomposition for Solving Pattern Mining Problems", ACM Transactions on Management Information Systems, 2021 Publication	<1 %
72	Yulan Shen, Ji Zhang, Jin Liu, Pu Zhan, Ruizhi Chen, Yanbo Chen. "Short-term load forecasting of power system based on similar day method and PSO-DBN", 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), 2018 Publication	<1 %
73	Zhang, Y.. "Handling multi-objective optimization problems with a multi-swarm cooperative particle swarm optimizer", Expert Systems With Applications, 201110	<1 %

	Publication	
74	downloads.hindawi.com Internet Source	<1 %
75	ebin.pub Internet Source	<1 %
76	www.informatica.si Internet Source	<1 %
77	Ja-Hwung Su, Wen-Yang Lin, Yi-Wen Liao, Guan-Hua Lai. "An Efficient Data Mining Algorithm by Multi-Utility Minimum Support and Prefix-Search Strategy", 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019 Publication	<1 %
78	Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Tzung-Pei Hong, Miroslav Voznak. "A binary PSO approach to mine high-utility itemsets", Soft Computing, 2016 Publication	<1 %
79	Jerry Chun-Wei Lin, Matin Pirouz, Youcef Djenouri, Chien-Fu Cheng, Usman Ahmed. "Incrementally updating the high average-utility patterns with pre-large concept", Applied Intelligence, 2020 Publication	<1 %
80	Laith Abualigah, Amir H. Gandomi, Mohamed Abd Elaziz, Husam Al Hamad, Mahmoud	<1 %

Omari, Mohammad Alshinwan, Ahmad M. Khasawneh. "Advances in Meta-Heuristic Optimization Algorithms in Big Data Text Clustering", Electronics, 2021
Publication

---

81 Wu, . "Some Variants of Particle Swarm OptimiSation", Chapman & Hall/CRC Numerical Analy & Scient Comp Series, 2011. <1%
Publication

---

82 Wu, Cheng Wei, Bai-En Shie, Vincent S. Tseng, and Philip S. Yu. "Mining top-K high utility itemsets", Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 12 KDD 12, 2012. <1%
Publication

---

83 "Advances in Knowledge Discovery and Data Mining", Springer Science and Business Media LLC, 2019 <1%
Publication

---

84 Huong Bui, Bay Vo, Tu-Anh Nguyen-Hoang, Unil Yun. "Mining frequent weighted closed itemsets using the WN-list structure and an early pruning strategy", Applied Intelligence, 2020 <1%
Publication

---

85 Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Jimmy Ming-Thai Wu, Tzung-Pei Hong, <1%

Leon Shyue-Liang Wang, Justin Zhan. "Mining high-utility itemsets based on particle swarm optimization", Engineering Applications of Artificial Intelligence, 2016
Publication

86 Krishan Kumar Sethi, Dharavath Ramesh. "High average-utility itemset mining with multiple minimum utility threshold: A generalized approach", Engineering Applications of Artificial Intelligence, 2020 <1%
Publication

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		