Brought to you by IPB University

Knowledge-Based Systems • Volume 190 • 29 February 2020 • Article number 105164

Document type
Article

Source type
Journal

ISSN
09507051

DOI
10.1016/j.knosys.2019.105164

Publisher
Elsevier B.V.

CODEN
KNSYE

Original language
English

View less

A BPSO-based method for high-utility itemset mining without minimum utility threshold

Gunawan, Ridowati[a, b] ; Winarko, Edi[a] ; Pulungan, Reza[a]

Save all to author list

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

[b] Department of Informatics, Faculty of Science and Technology, Sanata Dharma University, Yogyakarta, Indonesia

13 78th percentile Citations in Scopus | 1.26 FWCI | 49 Views count | View all metrics

Full text options Export

Abstract

High-utility itemset mining is used to obtain high utility itemsets by taking into account both the quantity as well as the utility of each item, which have not been considered in frequent itemset mining. Many algorithms compute high utility itemsets by setting a minimum utility threshold in advance. However, determining the minimum utility threshold is not easy. Too high or too low a threshold may result in incorrect high utility itemsets. In this paper, we propose a method based on binary particle swarm optimization to optimize the search for high utility itemsets without setting the minimum utility threshold beforehand. Instead, the application of the minimum utility threshold is performed as a post-processing step. Experiments on five datasets indicate that the proposed method is better than existing methods in finding high utility itemsets, and the time to obtain those itemsets is faster than that with setting the minimum utility threshold first. © 2019 Elsevier B.V.

Author keywords

Binary particle swarm optimization; Computational intelligence; Data mining; High-utility itemset mining; Minimum utility threshold

Cited by 13 documents

Heuristically mining the top-k high-utility itemsets with cross-entropy optimization

Song, W. , Zheng, C. , Huang, C.
(2022) Applied Intelligence

Discovery of Interesting Itemsets for Web Service Composition Using Hybrid Genetic Algorithm

Kannimuthu, S. , Chakravarthy, D.G.
(2022) Neural Processing Letters

High utility itemset mining using genetic algorithm assimilated with off policy reinforcement learning to adaptively calibrate crossover operation

Logeswaran, K. , Suresh, P.
(2022) Computational Intelligence

View all 13 citing documents

Inform me when this document is cited in Scopus:

Set citation alert

Related documents

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

Gunawan, R. , Winarko, E. , Pulungan, R.
(2022) Evolutionary Intelligence

An Adaptive Utility Quantification Strategy for Penetration Semantic Knowledge Mining

Zang, Y. , Hu, T. , Cao, R.
(2020) Communications in Computer and Information Science

Online Retail Pattern Quality Improvement: From Frequent Sequential Pattern to High-Utility Sequential Pattern

Gunawan, R.
(2021) 2021 4th International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2021

View all related documents based on references

Find more related documents in Scopus based on:

Authors Keywords

Indexed keywords

SciVal Topics

Metrics

Funding details

References (38)

View in search results format >

All

CSV export | Print | E-mail | Save to PDF | Create bibliography

1 Yan, X., Zhang, C., Zhang, S.
Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support
(2009) *Expert Systems with Applications*, Part 2 36 (2 PART 2), pp. 3066-3076. Cited 162 times.
doi: 10.1016/j.eswa.2008.01.028
View at Publisher

2 Kuo, R.J., Chao, C.M., Chiu, Y.T.
Application of particle swarm optimization to association rule mining
(2011) *Applied Soft Computing Journal*, 11 (1), pp. 326-336. Cited 161 times.
doi: 10.1016/j.asoc.2009.11.023
View at Publisher

3 Gupta, M.
Application of weighted particle swarm optimization in association rule mining
(2012) *Int. J. Comput. Sci. Inform.*, 1 (3), pp. 69-74. Cited 19 times.

4 Kuo, R.J., Shih, C.W.
Association rule mining through the ant colony system for National Health Insurance Research Database in Taiwan (Open Access)
(2007) *Computers and Mathematics with Applications*, 54 (11-12), pp. 1303-1318. Cited 46 times.
doi: 10.1016/j.camwa.2006.03.043
View at Publisher

5 Mangat, V.
Swarm intelligence based technique for rule mining in the medical domain
(2010) *Int. J. Comput. Appl.*, 4 (1), pp. 19-24. Cited 14 times.

6 Kannimuthu, S., Premalatha, K.
Discovery of high utility itemsets using genetic algorithm
(2013) *International Journal of Engineering and Technology*, 5 (6), pp. 4866-4880. Cited 8 times.
http://www.enggjournals.com/ijet/docs/IJET13-05-06-306.pdf

7 Lin, J.C.-W., Yang, L., Fournier-Viger, P., Frnda, J., Sevcik, L., Voznak, M.
An evolutionary algorithm to mine high-utility itemsets (Open Access)
(2015) *Advances in Electrical and Electronic Engineering*, 13 (4), pp. 392-398. Cited 6 times.
http://advances.utc.sk/index.php/AEEE/article/download/1474/1096
doi: 10.15598/aeee.v13i4.1474
View at Publisher

8 Lin, J.C.-W., Yang, L., Fournier-Viger, P., Wu, J.M.-T., Hong, T.-P., Wang, L.S.-L., Zhan, J.
Mining high-utility itemsets based on particle swarm optimization

(2016) *Engineering Applications of Artificial Intelligence*, 55, pp. 320-330. Cited 68 times.
doi: 10.1016/j.engappai.2016.07.006

View at Publisher

9 Lin, J.C.-W., Yang, L., Fournier-Viger, P., Hong, T.-P., Voznak, M.
A binary PSO approach to mine high-utility itemsets

(2017) *Soft Computing*, 21 (17), pp. 5103-5121. Cited 76 times.
http://springerlink.metapress.com/app/home/journal.asp?wasp=h83ak0wtmr5uxkah9j5m&referrer=parent&backto=browsepublicationsresults,466,533;
doi: 10.1007/s00500-016-2106-1

View at Publisher

10 Wu, J.M.-T., Zhan, J., Lin, J.C.-W.
An ACO-based approach to mine high-utility itemsets

(2017) *Knowledge-Based Systems*, 116, pp. 102-113. Cited 81 times.
doi: 10.1016/j.knosys.2016.10.027

View at Publisher

11 Tseng, V.S., Wu, C.-W., Fournier-Viger, P., Yu, P.S.
Efficient Algorithms for Mining Top-K High Utility Itemsets

(2016) *IEEE Transactions on Knowledge and Data Engineering*, 28 (1), art. no. 7164333, pp. 54-67. Cited 168 times.
http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=69
doi: 10.1109/TKDE.2015.2458860

View at Publisher

12 Yao, H., Hamilton, H.J., Butz, G.J.
A foundational approach to mining itemset utilities from databases

(2004) *SIAM Proceedings Series*, pp. 482-486. Cited 322 times.
http://www.siam.org/proceedings/
doi: 10.1137/1.9781611972740.51

View at Publisher

13 Yao, H., Hamilton, H.J.
Mining itemset utilities from transaction databases

(2006) *Data and Knowledge Engineering*, 59 (3 SPEC. ISS.), pp. 603-626. Cited 294 times.
doi: 10.1016/j.datak.2005.10.004

View at Publisher

14 Agrawal, R., Srikant, R.
Fast algorithms for mining association rules
(1994) *Proceedings of the 20th International Conference on Very Large Data Bases, Vol. 1215*, pp. 487-499. Cited 12858 times.

15 Liu, Y., Liao, W.-K., Choudhary, A.
A two-phase algorithm for fast discovery of high utility itemsets

(2005) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3518 LNAI, pp. 689-695. Cited 500 times.
https://www.springer.com/series/558
ISBN: 3540260765; 978-354026076-9
doi: 10.1007/11430919_79

View at Publisher

16 Erwin, A., Gopalan, R.P., Achuthan, N.R.
CTU-mine: An efficient high utility itemset mining algorithm using the pattern growth approach

(2007) *CIT 2007: 7th IEEE International Conference on Computer and Information Technology*, art. no. 4385059, pp. 71-76. Cited 83 times.
ISBN: 0769529836; 978-076952983-7
doi: 10.1109/CIT.2007.4385059

View at Publisher

17 Tseng, V.S., Wu, C.-W., Shie, B.-E., Yu, P.S.
UP-Growth: An efficient algorithm for high utility itemset mining

(2010) *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 253-262. Cited 356 times.
ISBN: 978-145030055-1
doi: 10.1145/1835804.1835839

View at Publisher

18 Lin, C.-W., Hong, T.-P., Lu, W.-H.
An effective tree structure for mining high utility itemsets

(2011) *Expert Systems with Applications*, 38 (6), pp. 7419-7424. Cited 190 times.
doi: 10.1016/j.eswa.2010.12.082

View at Publisher

19 Liu, M., Qu, J.
Mining high utility itemsets without candidate generation

(2012) *ACM International Conference Proceeding Series*, pp. 55-64. Cited 460 times.
ISBN: 978-145031156-4
doi: 10.1145/2396761.2396773

View at Publisher

20 Liu, J., Wang, K., Fung, B.C.M.
Mining High Utility Patterns in One Phase without Generating Candidates

(2016) *IEEE Transactions on Knowledge and Data Engineering*, 28 (5), art. no. 7360176, pp. 1245-1257. Cited 93 times.
http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=69
doi: 10.1109/TKDE.2015.2510012

View at Publisher

21 Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.
FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning

(2014) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8502 LNAI, pp. 83-92. Cited 309 times.
http://springerlink.com/content/0302-9743/copyright/2005/
ISBN: 978-331908325-4
doi: 10.1007/978-3-319-08326-1_9

View at Publisher

22 Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S.
EFIM: a fast and memory efficient algorithm for high-utility itemset mining

(2017) *Knowledge and Information Systems*, 51 (2), pp. 595-625. Cited 135 times.
https://www.springer.com/journal/10115
doi: 10.1007/s10115-016-0986-0

View at Publisher

23 Zhang, C., Almpanidis, G., Wang, W., Liu, C.
An empirical evaluation of high utility itemset mining algorithms

(2018) *Expert Systems with Applications*, 101, pp. 91-115. Cited 25 times.
doi: 10.1016/j.eswa.2018.02.008

View at Publisher

24 Duong, Q.-H., Liao, B., Fournier-Viger, P., Dam, T.-L.
An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies

(2016) *Knowledge-Based Systems*, 104, pp. 106-122. Cited 64 times.
doi: 10.1016/j.knosys.2016.04.016

View at Publisher

25 Fournier-Viger, P., Li, Z., Lin, J.C.-W., Kiran, R.U., Fujita, H.
Efficient algorithms to identify periodic patterns in multiple sequences

(2019) *Information Sciences*, 489, pp. 205-226. Cited 31 times.
http://www.journals.elsevier.com/information-sciences/
doi: 10.1016/j.ins.2019.03.050

View at Publisher

26 Fournier-Viger, P., Zhang, Y., Chun-Wei Lin, J., Fujita, H., Koh, Y.S.
Mining local and peak high utility itemsets

(2019) *Information Sciences*, 481, pp. 344-367. Cited 52 times.
http://www.journals.elsevier.com/information-sciences/
doi: 10.1016/j.ins.2018.12.070

View at Publisher

27 Yun, U., Kim, D., Yoon, E., Fujita, H.
Damped window based high average utility pattern mining over data streams

(2018) *Knowledge-Based Systems*, 144, pp. 188-205. Cited 90 times.
doi: 10.1016/j.knosys.2017.12.029

View at Publisher

28 Nguyen, L.T.T., Vu, V.V., Lam, M.T.H., Duong, T.T.M., Manh, L.T., Nguyen, T.T.T., Vo, B., (...), Fujita, H.
An efficient method for mining high utility closed itemsets

(2019) *Information Sciences*, 495, pp. 78-99. Cited 40 times.
http://www.journals.elsevier.com/information-sciences/
doi: 10.1016/j.ins.2019.05.006

View at Publisher

29 Kennedy, James, Eberhart, Russell C.
Discrete binary version of the particle swarm algorithm

(1997) *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 5, pp. 4104-4108. Cited 4455 times.

View at Publisher

30 Lin, J.C.-W., Gan, W., Fournier-Viger, P., Hong, T.-P., Tseng, V.S.
Efficient algorithms for mining high-utility itemsets in uncertain databases

(2016) *Knowledge-Based Systems*, 96, pp. 171-187. Cited 87 times.
doi: 10.1016/j.knosys.2015.12.019

View at Publisher

31 Song, W., Huang, C.
Mining High Utility Itemsets Using Bio-Inspired Algorithms: A Diverse Optimal Value Framework (Open Access)

(2018) *IEEE Access*, 6, pp. 19568-19582. Cited 42 times.
http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639
doi: 10.1109/ACCESS.2018.2819162

View at Publisher

32 Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K.
Efficient tree structures for high utility pattern mining in incremental databases

(2009) *IEEE Transactions on Knowledge and Data Engineering*, 21 (12), art. no. 4782959, pp. 1708-1721. Cited 450 times.
doi: 10.1109/TKDE.2009.46

View at Publisher

33 Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S.
EFIM: A highly efficient algorithm for high-utility itemset mining
(Open Access)

(2015) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9413, pp. 530-546. Cited 108 times.
http://springerlink.com/content/0302-9743/copyright/2005/
ISBN: 978-331927059-3
doi: 10.1007/978-3-319-27060-9_44

View at Publisher

34 Clerc, M.
The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization

(1999) *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3, art. no. 785513, pp. 1951-1957. Cited 1635 times.
doi: 10.1109/CEC.1999.785513

View at Publisher

35 Eberhart, Russell, Kennedy, James
New optimizer using particle swarm theory

(1995) *Proceedings of the International Symposium on Micro Machine and Human Science*, pp. 39-43. Cited 13625 times.

36 Tasgetiren, M.F., Liang, Y.-C.
A binary particle swarm optimization algorithm for lot sizing problem
(2004) *J. Econ. Soc. Res.*, 5 (2), pp. 1-20. Cited 97 times.

37 Fournier-Viger, P., Lin, J.C., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.
The SPMF open-source data mining library version 2

(2016) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9853 LNCS, pp. 36-40. Cited 308 times.
http://springerlink.com/content/0302-9743/copyright/2005/
ISBN: 978-331946130-4
doi: 10.1007/978-3-319-46131-1_8

View at Publisher

38 Krishnamoorthy, S.
A Comparative Study of Top-K High Utility Itemset Mining Methods
(Open Access)

(2019) *Studies in Big Data*, 51, pp. 47-74. Cited 5 times.
springer.com/series/11970
doi: 10.1007/978-3-030-04921-8_2

View at Publisher

Winarko, E.; Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia; email:ewinarko@ugm.ac.id
© Copyright 2020 Elsevier B.V., All rights reserved.

About Scopus

What is Scopus
Content coverage
Scopus blog
Scopus API
Privacy matters

Language

日本語版を表示する
查看简体中文版本
查看繁體中文版本
Просмотр версии на русском языке

Customer Service

Help
Tutorials
Contact us

ELSEVIER

Terms and conditions Privacy policy

Copyright © Elsevier B.V. All rights reserved. Scopus® is a registered trademark of Elsevier B.V.
We use cookies to help provide and enhance our service and tailor content. By continuing, you agree to the use of cookies.

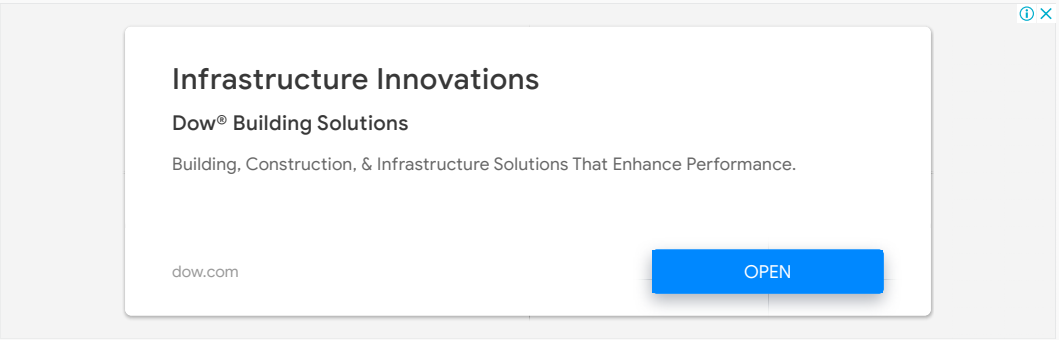
RELX

also developed by scimago: SCIMAGO INSTITUTIONS RANKINGS

SJR Scimago Journal & Country Rank

Enter Journal Title, ISSN or Publisher Name

Home Journal Rankings Country Rankings Viz Tools Help About Us



Knowledge-Based Systems

COUNTRY

Netherlands

Universities and research institutions in Netherlands

SUBJECT AREA AND CATEGORY

Business, Management and Accounting
- Management Information Systems

Computer Science
- Artificial Intelligence
- Software

Decision Sciences
- Information Systems and Management

PUBLISHER

Elsevier

H-INDEX

135

PUBLICATION TYPE

Journals

ISSN

09507051

COVERAGE

1987-2021

INFORMATION

Homepage

How to publish in this journal

Contact

Dow® Infrastructure Solutions

High Performance Silicones

Building, Construction, & Infrastructure Solutions That Enhance Performance.

dow.com

OPEN

Submit Your Manuscript

Submit Your Paper

Gavin Publishers is an International Open Access Peer Reviewed Online Journal Publication.

gavinpublishers.com

OPEN

SCOPE

Knowledge-Based Systems is an international, interdisciplinary and applications-oriented journal. This journal focuses on systems that use knowledge-based (KB) techniques to support human decision-making, learning and action; emphases the practical significance of such KB-systems; its computer development and usage; covers the implementation of such KB-systems: design process, models and methods, software tools, decision-support mechanisms, user interactions, organizational issues, knowledge acquisition and representation, and system architectures. This journal's current leading topics are but not limited to: • Big data techniques and methodologies, data-driven information systems, and knowledge acquisition • Cognitive interaction and intelligent human interfaces • Recommender systems and E-service personalization • Intelligent decision support systems, prediction systems and warning systems • Computational and artificial intelligence based systems and uncertain information processes • Swarm intelligence and evolutionary computing • Knowledge engineering, machine learning-based systems and web semantics The journal also welcomes papers describing novel applications of knowledge based systems in any human endeavor: ranging from financial technology to engineering to health science or any other domain impacted by Artificial Intelligence technologies and its associated techniques and systems.

Join the conversation about this journal

Submit Your Manuscript

Submit Your Paper

Gavin Publishers is an International Open Access Peer Reviewed Online Journal Publication.

gavinpublishers.com

OPEN

Quartiles

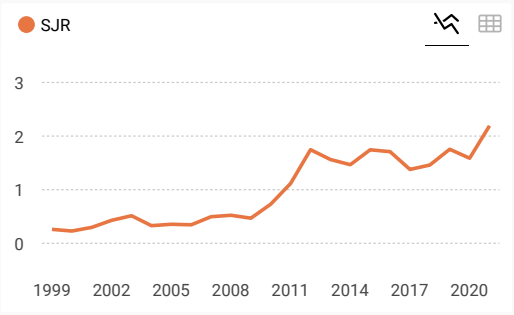
FIND SIMILAR JOURNALS

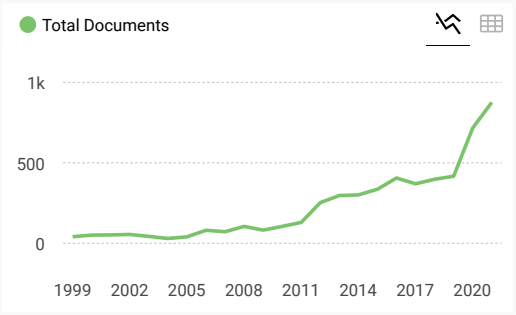
1
International Journal of Machine Learning and
USA
69%
similarity

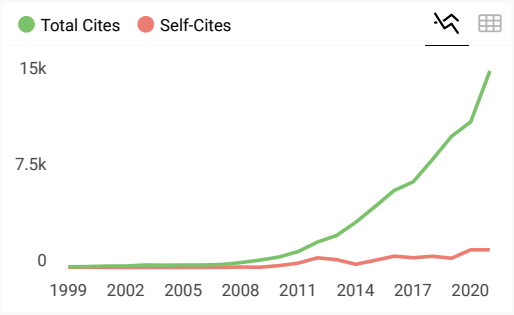
2
Applied Intelligence
NLD
66%
similarity

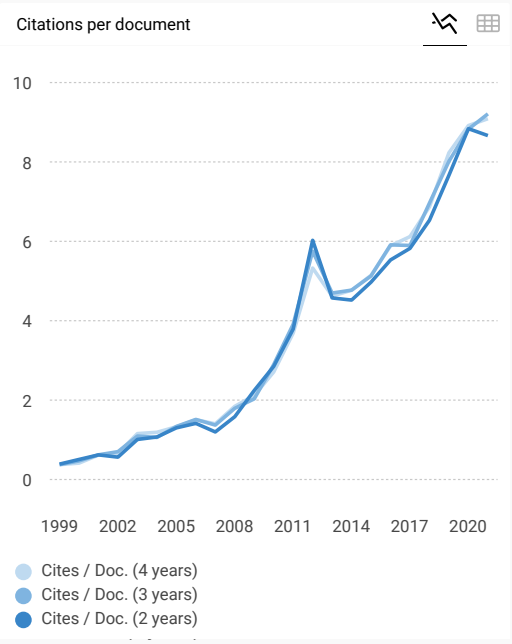
3
International Journal of Computational Intelligence
GBR
62%
similarity

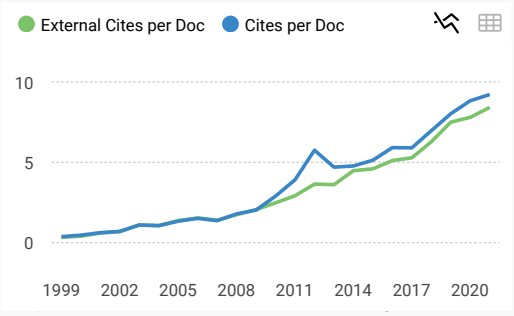
4
Intelligent Data Analysi
NLD
57%
similarity

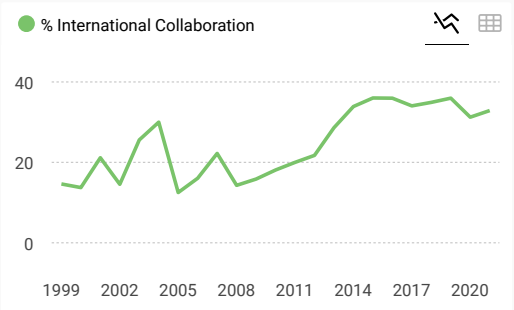


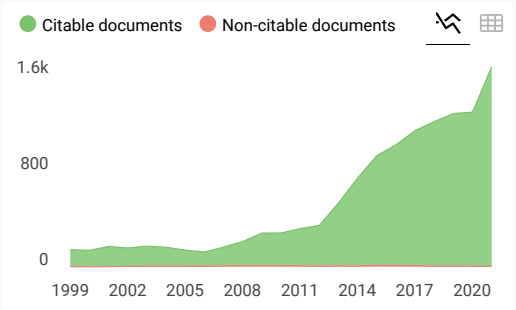


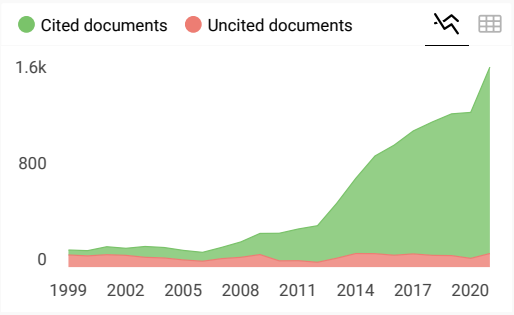


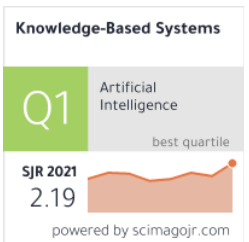












← Show this widget in your own website

Just copy the code below and paste within your html code:

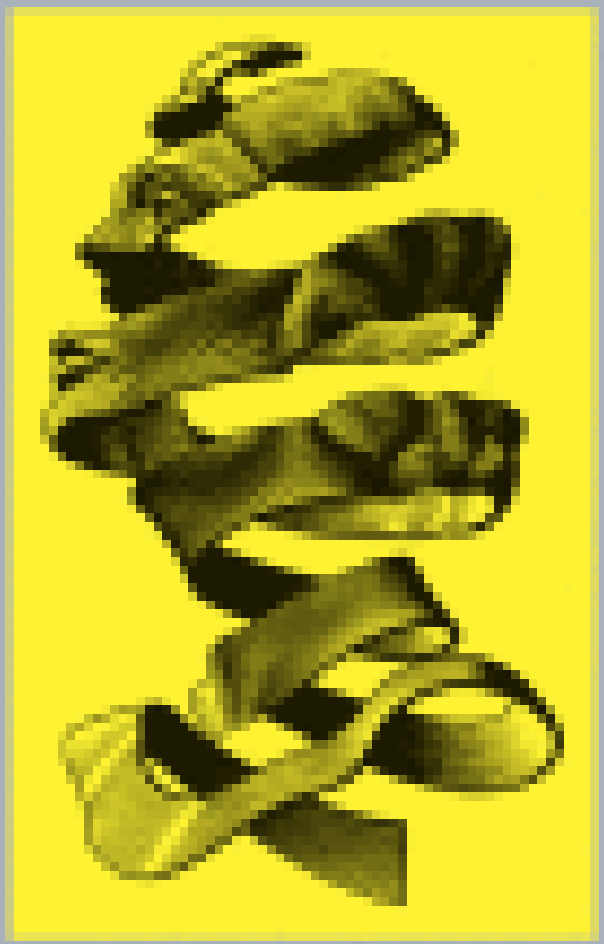
<a href="https://www.scim

SCImago Graphica

Explore, visually communicate and make sense of data with our **new data visualization tool**.

Volume 190, 29 FEBRUARY 2020 ISSN 0950-7051

Knowledge-Based SYSTEMS



Editor-in-Chief

Professor J. Lu, Fac. of Engineering and Information Technology, University of Technology Sydney, 15 Broadway, P.O. Box 123, Sydney, NSW 2007, New South Wales, Australia
E-mail: jie.lu@uts.edu.au

Emeritus Editor (2010–2019)

Professor H. Fujita, Fac. of Software and Information Science, Iwate Prefectural University, 020-0193, Iwate, Japan
E-mail: issam@iwate-pu.ac.jp

Associate Editors

Associate Professor E. Cambria
Nanyang Technological University, Singapore

Professor R.M. de Moraes
Universidade Federal da Paraíba, Brazil

Professor H. Hagras
University of Essex, UK

Professor E. Herrera-Viedma
Universidad de Granada, Spain

Professor C. Kahraman
Istanbul Technical University, Turkey

Professor Z. Li
Fern University, Germany

Professor J. Liu
Ulster University, UK

Professor L.M. López
University of Jaen, Spain

Professor L. Oneto
DIBRIS - University of Genoa, Italy

Professor N. Pal
Indian Statistical Institute, India

Professor I. Perfilieva
University of Ostrava, Czech Republic

Editorial Advisory Board

Professor U.R. Acharya
Ngee Ann Polytechnic, Singapore, Singapore

Professor D. Benyon
Edinburgh Napier University, Edinburgh, Scotland, UK

Professor E. Bonjour
Université de Lorraine, Nancy Cedex, France

Professor S.M. Chen
National Taiwan University of Science and Technology, Taipei, Taiwan

Professor F. Chiclana
De Montfort University, Leicester, England, UK

Professor W. Ding
Nantong University, Nantong, China

Dr. M. Dyer
University of California at Los Angeles (UCLA), Los Angeles, California, USA

Dr. M. Eisenberg
University of Colorado, Boulder, Colorado, USA

Professor G. Fischer
University of Colorado, Boulder, Colorado, USA

Professor J.S. Gero
University of North Carolina at Charlotte, Charlotte, North Carolina, USA

Professor F. Herrera
Universidad de Granada, Granada, Spain

Founding Editor

Professor E.A. Edmonds
De Montfort University, Leicester, England, UK

Professor M. Reformat
University of Alberta, Canada

Professor S. Wang
Jiangnan University, China

Professor X. Wang
Tianjin University, China

Associate Professor Q. Wei
Tsinghua University, China

Professor X. Yao
Southern University of Science and Technology, China
University of Birmingham, UK

Professor H. Ying
Wayne State University, USA

Professor U. Yun
Sejong University, Seoul, Republic of Korea

Associate Professor N. Zhang
University of the District of Columbia, USA

Professor Z. Zheng
Beihang University, China

Professor Z. Zhou
Nanjing University, China

Professor T. Li
Southwest Jiaotong University, Chengdu, China

Dr. H. Lieberman
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Dr. C.A. Miller
Smart Information Flow Technologies, Minneapolis, Minnesota, USA

Dr. D. Riecken
IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA

Professor A. Selamat
Universiti Teknologi Malaysia, Johor Bahru, Malaysia

Dr. K. Taylor
Australian National University, Canberra, Australian Capital Territory, Australia

Dr. L.G. Terveen
University of Minnesota, Minneapolis, USA

Dr. E. Tsui
The Hong Kong Polytechnic University, Kowloon, Hong Kong

Professor E. Tyugu
Institute of Cybernetics at Tallinn University of Technology, Tallinn, Estonia

Professor T. Wu
Shanghai Jiao Tong University, Shanghai, China

Dr. Y. Yao
University of Regina, Regina, Saskatchewan, Canada

ScienceDirect

Submit your article

Menu Search in this journal

Volume 190

29 February 2020

Download full issue

Previous vol/issue

Next vol/issue

Receive an update when the latest issues in this journal are published

Sign in to set up alerts

FEEDBACK

Knowledge-Based Systems
Supports *open access*
Submit your article
Menu

king considering experts'

Xuanhua Xu, Qianhui Zhang, Xiaohong Chen
Article 105108

Article preview

Research article ○ Abstract only

A BPSO-based method for high-utility itemset mining without minimum utility threshold

Ridowati Gunawan, Edi Winarko, Reza Pulungan
Article 105164

Article preview

Research article ○ Abstract only

An efficient active learning method for multi-task learning

Yanshan Xiao, Zheng Chang, Bo Liu
Article 105137

Article preview

Research article ● *Open access*

Word Sense Disambiguation: A comprehensive knowledge exploitation framework

Yinglin Wang, Ming Wang, Hamido Fujita
Article 105030

Download PDF Article preview

Software publication ○ Abstract only

Senpy: A framework for semantic sentiment and emotion analysis services

J. Fernando Sánchez-Rada, Oscar Araque, Carlos A. Iglesias
Article 105193

Article preview

Research article ○ Abstract only

adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection

Xuhong Wang, Ying Du, Shijie Lin, Ping Cui, ... Yupu Yang
Article 105187

Article preview

Research article ○ Abstract only

Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms

Hamdi Tolga Kahraman, Sefa Aras, Eyüp Gedikli
Article 105169

Article preview

FEEDBACK

Knowledge-Based Systems
Supports *open access*

Submit your article

Menu

Research article ○ Abstract only

Generative adversarial networks based on Wasserstein distance for knowledge graph embeddings

Yuanfei Dai, Shiping Wang, Xing Chen, Chaoyang Xu, Wenzhong Guo
Article 105165

Article preview

Research article ○ Abstract only

NearCount: Selecting critical instances based on the cited counts of nearest neighbors

Zonghai Zhu, Zhe Wang, Dongdong Li, Wenli Du
Article 105196

Article preview

Research article ○ Abstract only

Reinforcement learning approach for optimal control of multiple electric locomotives in a heavy-haul freight train:A Double-Switch-Q-network architecture

Huiyue Tang, Yuan Wang, Xiang Liu, Xiaoyun Feng
Article 105173

Article preview

Research article ○ Abstract only

Online transfer learning with multiple source domains for multi-class classification

Zhongfeng Kang, Bo Yang, Shantian Yang, Xiaomei Fang, Changjian Zhao
Article 105149

Article preview

Research article ○ Abstract only

Effective Bayesian-network-based missing value imputation enhanced by crowdsourcing

Chen Ye, Hongzhi Wang, Wenbo Lu, Jianzhong Li
Article 105199

Article preview

Research article ○ Abstract only

Semi-supervised multi-view clustering with Graph-regularized Partially Shared Non-negative Matrix Factorization

Naiyao Liang, Zuyuan Yang, Zhenni Li, Shengli Xie, Chun-Yi Su
Article 105185

Article preview

Research article ○ Abstract only

FEEDBACK

Knowledge-Based Systems
Supports *open access*
Submit your article
Menu

pplications

Research article ○ Abstract only

A context-aware embeddings supported method to extract a fuzzy sentiment polarity dictionary

J. Bernabé-Moreno, A. Tejeda-Lorente, J. Herce-Zelaya, C. Porcel, E. Herrera-Viedma
Article 105236

Article preview

Research article ○ Abstract only

Intelligent decision making for service providers selection in maintenance service network: An adaptive fuzzy-neuro approach

Weibo Ren, Kezhong Wu, Qiusheng Gu, Yaoguang Hu
Article 105263

Article preview

Research article ○ Abstract only

Fuzzy entropy clustering by searching local border points for the analysis of gene expression data

Yiping Zeng, Zeshui Xu, Yue He, Yu Rao
Article 105309

Article preview

Research article ○ Abstract only

An OWA-based hierarchical clustering approach to understanding users' lifestyles

Jennifer Nguyen, Albert Armisen, Germán Sánchez-Hernández, Mònica Casabayó, Núria Agell
Article 105308

Article preview

Research article ○ Abstract only

Automatic construction of multi-faceted user profiles using text clustering and its application to expert recommendation and filtering problems

Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, Luis Redondo-Expósito
Article 105337

Article preview

Research article ○ Abstract only

Adaptive early classification of temporal sequences using deep reinforcement learning

Coralie Martinez, Emmanuel Ramasso, Guillaume Perrin, Michèle Rombaut
Article 105290

Article preview

Research article ○ Abstract only

FEEDBACK

Knowledge-Based Systems
Supports *open access*
Submit your article
Menu

deling

Research article ○ Abstract only

Machine learning-based wear fault diagnosis for marine diesel engine by fusing multiple data-driven models

Xiaojian Xu, Zhuangzhuang Zhao, Xiaobin Xu, Jianbo Yang, ... Guodong Wang
Article 105324

Article preview

Research article ○ Abstract only

Predicting ICD-9 code groups with fuzzy similarity based supervised multi-label classification of unstructured clinical nursing notes

Tushaar Gangavarapu, Aditya Jayasimha, Gokul S. Krishnan, Sowmya Kamath S.
Article 105321

Article preview

Research article ○ Abstract only

Deep time–frequency representation and progressive decision fusion for ECG classification

Jing Zhang, Jing Tian, Yang Cao, Yuxiang Yang, Xiaobin Xu
Article 105402

Article preview

Research article ○ Abstract only

Machine learning based decision making for time varying systems: Parameter estimation and performance optimization

Yiyang Chen, Yingwei Zhou
Article 105479

Article preview

Research article ○ Abstract only

Detection of SQL injection based on artificial neural network

Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, Guozhen Liu
Article 105528

Article preview

Previous vol/issue

Next vol/issue

ISSN: 0950-7051

Copyright © 2022 Elsevier B.V. All rights reserved

Copyright © 2022 Elsevier B.V. or its licensors or contributors.
ScienceDirect® is a registered trademark of Elsevier B.V.

RELX™

FEEDBACK

Knowledge-Based Systems
Supports *open access*

Submit your article

Menu

FEEDBACK

Knowledge-Based Systems 190 (2020) 105164

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A BPSO-based method for high-utility itemset mining without minimum utility threshold☆

Ridowati Gunawan [a,b], Edi Winarko [a,*], Reza Pulungan [a]

[a] Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia
[b] Department of Informatics, Faculty of Science and Technology, Sanata Dharma University, Yogyakarta, Indonesia

ARTICLE INFO

Article history:
Received 20 March 2019
Received in revised form 27 October 2019
Accepted 29 October 2019
Available online 31 October 2019

Keywords:
High-utility itemset mining
Binary particle swarm optimization
Minimum utility threshold
Computational intelligence
Data mining

ABSTRACT

High-utility itemset mining is used to obtain high utility itemsets by taking into account both the quantity as well as the utility of each item, which have not been considered in frequent itemset mining. Many algorithms compute high utility itemsets by setting a minimum utility threshold in advance. However, determining the minimum utility threshold is not easy. Too high or too low a threshold may result in incorrect high utility itemsets. In this paper, we propose a method based on binary particle swarm optimization to optimize the search for high utility itemsets without setting the minimum utility threshold beforehand. Instead, the application of the minimum utility threshold is performed as a post-processing step. Experiments on five datasets indicate that the proposed method is better than existing methods in finding high utility itemsets, and the time to obtain those itemsets is faster than that with setting the minimum utility threshold first.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Data mining is a set of techniques used to obtain hidden, useful, and interesting knowledge from a large dataset. One of the techniques is association rule mining, whose purpose is to get transaction patterns, correlations, or associations among items in the dataset. Association rule mining begins with finding items that frequently appear together. Association rule mining, however, only pays attention to the appearances of the items, and the utility of those items are considered the same. This technique is called *frequent itemset mining*. Frequent itemset mining can be significantly improved by taking into account the quantity and the weight of the items. Quantity represents the number of appearances of the same item, while weight may model the profit of each item. In this technique, the quantity and the weight of all items are no longer considered the same. Itemset mining that considers both is known as *high-utility itemset mining* (HUIM). Itemsets obtained this way will be more acceptable to the user because they become more meaningful and fit the actual conditions.

Previous researches in high-utility itemset mining mostly focused on obtaining high utility itemsets in a large search space by considering two main factors, namely speed and memory usage. Determining the optimal minimum utility threshold to get those high utility itemsets is often overlooked. Too high a threshold may result in not getting any high utility itemset, while too low a threshold may result in getting too many itemsets that they become less meaningful. The user has to set the minimum utility threshold manually by trying exact utilities from the transaction data one by one. Furthermore, when the data changes, either because of addition or update, the minimum utility threshold that has been determined can no longer be used.

Determining the minimum utility threshold is analogous to determining the minimum support in frequent itemset mining. Obtaining the optimal frequent itemsets without first setting the minimum support has been attempted by combining frequent itemset mining with computational intelligence, such as Genetic Algorithm (GA) [1], Particle Swarm Optimization (PSO) [2,3], and Ant Colony Optimization (ACO) [4,5]. This combination can improve the performance of the resulting method: it does not require a minimum utility threshold [1,2], it is faster in association rule searches [1,2,4], and it is more compatible with real-world applications as well as with the user's expectation [1,4]. Previous researches also indicated that among computational intelligence methods, the PSO-based method has a better computation time and generates more meaningful rules compared to GA-based [3]. We expect that a similar approach of not using a minimum utility threshold can also be used in high-utility itemset mining.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105164.

* Corresponding author.
E-mail addresses: ridowati.gunawan@mail.ugm.ac.id, rido@usd.ac.id (R. Gunawan), ewinarko@ugm.ac.id (E. Winarko), pulungan@ugm.ac.id (R. Pulungan).

https://doi.org/10.1016/j.knosys.2019.105164
0950-7051/© 2019 Elsevier B.V. All rights reserved.

High-utility itemset mining has been combined with computational intelligence, including with GA [6], with PSO [7–9], and with ACO [10]. Kannimuthu and Premalatha [6] proposed two GA-based algorithms, HUPE$_{umu}$-GA (with minimum utility threshold) and HUPE$_{wumu}$-GA (without minimum utility threshold). Even though HUPE$_{wumu}$-GA does not use a minimum utility threshold it produces better results than HUPE$_{umu}$-GA. Those studies, however, still focused on speed and memory utilization. Ways for tuning the minimum utility threshold have not been the focus of those studies, and the user still has to try to determine the appropriate minimum utility threshold manually.

Tseng et al. [11] proposed two efficient algorithms—TKU (mining Top-K Utility itemsets) and TKO (mining Top-K utility itemsets in One phase)—for mining high utility itemsets without a minimum utility threshold, but only need to set the number of produced itemsets. Although the algorithms succeeded in producing itemsets without a minimum utility threshold, they suffer from long running time and high memory consumption.

In this paper, we propose a method based on binary particle swarm optimization (BPSO) to mine high utility itemsets without first setting a minimum utility threshold. The method generates itemsets according to the BPSO algorithm and outputs all of them as a list sorted by the utility. A minimum utility threshold can then be imposed on this list as a post-processing step. We then compare our proposed method with other methods without a minimum utility threshold, such as TKU and TKO, and also compare it with other computational intelligence algorithms (swarm and evolutionary algorithms), such as HUPE$_{umu}$-GRAM^{+}, HUIM-BPSO^{+}, HUIM-ACS, Bio-HUIF-GA, Bio-HUIF-PSO, and Bio-HUIF-BA methods for various minimum utility thresholds.

The main contribution of the paper is an algorithm based on binary particle swarm optimization to obtain high utility itemsets without the need to set the minimum utility threshold beforehand. Further contributions are the use of transactions with the highest utility as the initial population of the particles, and the determination of the right parameters to get the high utility itemsets.

The rest of this paper is organized as follows: Related work is briefly reviewed in Section 2. Section 3 provides the concepts, definitions, and notations necessary for the rest of the paper. The proposed algorithm for mining high utility itemsets is described in Section 4. Experimental design, result, and analysis are presented in Section 5. The paper is concluded in Section 6.

2. Related work

In this section, work related to methods for finding high utility itemsets as well as particle swarm optimization for frequent itemset mining and high-utility itemset mining is briefly reviewed.

2.1. High-utility itemset mining

Frequent itemset mining has two limitations: it assumes that each item appears in a transaction once, and the weights of all items or their profits in the transaction are assumed equal [12]. Both limitations lead to researches on high-utility itemset mining. Yao et al. [12] defined two types of utility: transaction utility (*i.e.*, quantity) and external utility (*i.e.*, profit).

The focus of researches in high-utility itemset mining is to obtain itemsets with utility more than or equal to a minimum utility threshold. Yao and Hamilton [13] proposed a high-utility itemset mining using Agrawal and Srikant's apriori concept [14], in which searching for high utility itemsets is carried out by first generating candidate itemsets, namely by forming the cross product of all items. Candidate itemsets with utility more than the minimum utility threshold are considered high utility itemsets. Although they used the apriori concept to generate candidate high utility itemsets, Yao and Hamilton [13] proved that the apriori property or the downward closure property does not apply to high utility itemsets. Downward closure property stipulates that any subset of a frequent itemset must also be a frequent itemset. The downward closure property is not satisfied by high utility itemsets because a subset of a high utility itemset is not necessarily of high utility. The large number of candidate itemsets and the inability to apply the downward closure property on high utility itemsets result in long computation times in getting high utility itemsets.

Liu et al. [15] developed a two-phase algorithm that uses the concepts of transaction utility and transaction weighted utility to prune the search space of the high utility itemsets. The proposed transaction weighted utility maintains the downward closure property as in frequent itemset mining, and this is referred to as the transaction-weighted downward closure (TWDC) property. Because transaction-weighted utility itemsets satisfy the downward closure property, any frequent-itemset mining algorithm can be modified to mine high utility itemsets, notably the apriori algorithm. Transaction-weighted downward closure property is used to prune high-utility itemset candidates in the first phase, and then the complete high utility itemsets are obtained in the second phase. In the first phase, items with utility less than the minimum utility threshold are discarded, and this reduces the number of items used to generate high-utility itemset candidates.

Searching for high utility itemsets using FP-Growth and tree structure has a better computation time compared with using the two-phase algorithm [16–18]. However, the three algorithms have a common weakness: their large memory consumption. This weakness arises because the tree structure built is not efficient in storing information on the itemsets with high utility. Searching for high utility itemsets without generating candidate itemsets first has better performance, such as found in HUI-Miner [19], d2HUP [20], and Fast High Utility Miner (FHM) [21]. Efficient high-utility itemset mining (EFIM) uses two types of utility calculations, namely sub-tree utility and local utility [22].

Empirical studies have been conducted on the classical algorithms for high-utility itemset mining. Correct algorithm selection is usually based on the type of the datasets processed, whether they are dense or sparse, or long or short. EFIM is generally considered an efficient algorithm in terms of memory usage and is appropriate for small yet dense datasets. Meanwhile, d2HUP has a better computation time compared to EFIM, especially for large sparse datasets [23].

Instead of using a minimum utility threshold, Tseng et al. [11] proposed two efficient algorithms, TKU and TKO, for mining a predetermined desired number of high utility itemsets. Although the algorithms succeeded in obtaining high utility itemsets without a minimum utility threshold, they require long running time and have high memory consumption. Duong et al. [24] proposed an algorithm called kHMC to discover the top-K high utility itemsets more effectively in a single phase. They used three strategies, namely RIU, CUD, and COV, to raise its internal minimum utility threshold and reduce the search space. kHMC outperforms TKO in terms of memory consumption and running time.

Fournier et al. [25] suggested the importance of examining periodic patterns in multiple sequences. Although it has only been applied to frequent itemset mining, the idea of periodic patterns can also be applied to high-utility itemset mining. Fournier et al. [26] proposed three methods, namely mining local high utility, mining peak high utility, and mining non-redundant peak high utility itemsets. Apart from the minimum utility threshold, the addition of time-period elements is state-of-the-art in high utility itemset mining. Yun et al. [27] suggested a high-average

utility pattern mining (HAUPM) method for data streams using damped windows. The data used is no longer in a static database, but generally continuous flow of data with no restrictions, such as data obtained from heartbeat sensors. The damped window model is used to get more meaningful patterns in the stream environment. Nguyen et al. [28] proposed a method for obtaining closed high utility itemsets (CHUIs). CHUI mining involves finding a representative set of high utility itemsets, which is usually much smaller than the whole high utility itemsets but can generate the entire itemsets without loss of information. A method called HMiner-Closed is an effective miner of CHUIs for both dense and sparse datasets.

Instead of the traditional high-utility itemset mining, Kannimuthu and Premalatha [6] designed a GA-based algorithm to mine high utility itemsets through mutation. They proposed two approaches: with a minimum utility threshold called HUPE$_{umu}$-GA and without a minimum utility threshold called HUPE$_{wumu}$-GA. The fitness function in both methods is the utility. For HUPE$_{umu}$-GA a high utility itemset is an itemset with utility more than the minimum utility threshold, while a HUPE$_{wumu}$-GA itemset is a top-K utility itemset population derived from the evolutionary process.

Wu et al. [10] used ant colony system (ACS), which is extended from ant colony optimization (ACO) for searching high utility itemsets (HUIM-ACS). The designed algorithm adopts the TWU concept to reduce the combinatorial problem. Through certain routing graphs, HUIM-ACS does not have to perform utility calculations for the same itemsets and has the ability to check all processes whether there are itemsets in the transaction or not. From their conducted experiments, the proposed HUIM-ACS algorithm produces better results compared to the GA-based or PSO-based algorithms in terms of running time and the ability to discover high utility itemsets. However, it obtains fewer high utility itemsets than the PSO-based algorithm in the connect dataset.

Existing high-utility itemset algorithms, however, mostly focus on time and memory consumption. The determination of the minimum utility threshold is not the focus of previous researches, even though the minimum utility threshold significantly affects the performance of the algorithm and the obtained itemsets. Determining the minimum utility threshold is not easy, and therefore, research is needed to find out how to get high utility itemsets without first setting a minimum utility threshold.

2.2. Particle swarm optimization

Particle swarm optimization (PSO) was proposed in [29], and many variations of PSO have been developed henceforth. A PSO algorithm is a stochastic-based optimization technique inspired by the social behavior of a flock of birds or a school of fish. A particle is analogous to an individual, while a swarm is defined as a population or a flock. Each particle has two characteristics: position and velocity; it changes position according to its velocity. A particle remembers the best position it has ever passed (*pbest*), which is adjusted according to the outcomes of learning and the environment (*gbest*). Change to its position or velocity is influenced by the swarm used to achieve a particular goal.

PSO has also been used to obtain high utility itemsets in [7, 30]. The first step is finding 1-itemsets that meet the transaction weighted utility. A 1-itemset with transaction weighted utility less than a minimum utility threshold will not be used for further processing. Next, itemsets are encoded as particles using the roulette wheel and randomly determined particle velocity. The utility is used as a fitness function, and the sigmoid function is used in updating the particles' velocity. The proposed PSO-based algorithm, which uses BPSO with sigmoid function and transaction-weighted utility strategy, was named HUIM-BPSO$_{sig}$. Evaluation results showed that this algorithm is more efficient and faster in finding high utility itemsets compared to the GA-based algorithm (HUPE$_{umu}$-GA). HUIM-BPSO$_{sig}$ was further improved by altering its particles' encoding to use OR/NOR trees, used only at the initial population and during the population update. The improvement was called HUIM-BPSO^{+} [9], and it produced better results than HUIM-BPSO$_{sig}$. Even though HUIM-BPSO^{+} can produce more itemsets than HUIM-BPSO$_{sig}$ and sometimes can get the same number of itemsets as the deterministic algorithms, in some datasets (mushroom, accident, and foodmart) the number of obtained itemsets is still fewer. The deterministic algorithm used in the comparison is the TWU model [19].

Song and Huang [31] stated that high-utility itemset mining based on bio-inspired algorithms is more efficient but is not guaranteed to get all high utility itemsets in the database. Instead of using the current optimal value like the standard algorithms in the bio-inspired approach, they proposed to use proportionally selected discovered high utility itemsets as the target values of the next population. Three bio-inspired algorithms were put forward, namely genetic (Bio-HUIF-GA), particle swarm optimization (Bio-HUIF-PSO), and bat (Bio-HUIF-BA) algorithms. Of the three algorithms, based on their experiments, Bio-HUIF-GA produces the same number of itemsets as the deterministic algorithms used in the comparison, namely IHUP [32] and UP-Growth [11]. Bio-HUIF-PSO and Bio-HUIF-BA, on the other hand, sometimes generate all itemsets, and sometimes miss some.

3. Preliminaries

In this section we describe several basic concepts [9,21,33] required in high-utility itemset mining. Given a finite set of items $I = \{i_1, i_2, \ldots, i_m\}$, an itemset X is a set of distinct items and $X \subseteq I$. A transaction database $D = \{T_1, T_2, \ldots, T_n\}$ is a set of transactions T_c and T_c has an identifier c, called *Tid*. Each item $i \in I$ is associated with a positive value $pr(i)$, called external utility (*i.e.*, the profit of the item). Each item i in a transaction T_c has a positive value $q(i, T_c)$, called internal utility or transaction utility (*i.e.*, quantity).

Definition 1. The following definitions are required:

1. The utility of item i in transaction T_c is defined by $u(i, T_c) = pr(i) \cdot q(i, T_c)$.
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 in database D is defined by $TU = \Sigma_{T_c \in D} TU(T_c)$.

Example 1. An example is presented in Tables 1 and 2. Table 1 contains a sample database involving 7 transactions and 5 distinct items. Each transaction consists of items, each with its own quantity. Transaction T_1 consists of 3 items represented by b, c, and d, and with quantity 1, 1, and 2, respectively. The profit (external utility) of each item is shown in Table 2. It consists of 5 items with profit 5, 1, 2, 2, and 3.

Table 1
An example of a transaction database D.

TId	Transaction (item: quantity)
T_1	$(b:1)(c:1)(d:2)$
T_2	$(a:1)(b:7)(c:3)(d:2)$
T_3	$(a:1)(c:2)(d:3)$
T_4	$(c:2)(d:2)(e:5)$
T_5	$(a:5)(b:4)(d:5)(e:6)$
T_6	$(a:3)(b:8)(c:1)$
T_7	$(d:5)(e:4)$

Table 2
Profit (utility) of each item.

Item	a	b	c	d	e
Profit	5	1	2	2	3

Based on the tables, we provide the following examples to illustrate the calculation for Definition 1:

1. The utility of item d in transaction T_5 is $u(d, T_5) = pr(d) \cdot q(d, T_5) = 2 \cdot 5 = 10$.
2. The utility of itemset $\{a, d\}$ in transaction T_5 is $u(\{a, d\}, T_5) = u(a, T_5) + u(d, T_5) = pr(a) \cdot q(a, T_5) + pr(d) \cdot q(d, T_5) = 5 \cdot 5 + 2 \cdot 5 = 25 + 10 = 35$.
3. The utility of itemset $\{a, d\}$ in database D is $u(\{a, d\}) = u(\{a, d\}, T_2) + u(\{a, d\}, T_3) + u(\{a, d\}, T_5) = 9 + 11 + 35 = 55$.
4. The transaction utility of transaction T_5 is $TU(T_5) = u(a, T_5) + u(b, T_5) + u(d, T_5) + u(e, T_5) = 25 + 4 + 10 + 18 = 57$.
5. The transaction weighted utility of item a is $TWU(a) = TU(T_2) + TU(T_3) + TU(T_5) + TU(T_6) = 25 + 15 + 57 + 33 = 130$.
6. The total utility of database D is $TU = TU(T_1) + TU(T_2) + TU(T_3) + TU(T_4) + TU(T_5) + TU(T_6) + TU(T_7) = 7 + 25 + 13 + 23 + 57 + 33 + 22 = 180$.

4. Proposed method

We propose a method for high-utility itemset mining without minimum utility threshold, called **HUIM-BPSO-nomut** (High-Utility Itemset Mining without minimum utility threshold based on Binary Particle Swarm Optimization). This method consists of four phases: pre-processing, initializing and encoding particles, fitness evaluation, and updating phases, similar to previous BPSO-based methods [9,29]. Each of these phases will be described below. The pseudocode of the proposed method is depicted in Algorithm 1. Inputs to function HUIM-BPSO-nomut() are a database D containing transactions with items and quantities, the size of the population *pop_size*, and the number of iterations *iterations*. The output of the function is a set of high utility itemsets *HUIs*.

4.1. Pre-processing

In the pre-processing phase, the utility of each item in all transactions and the total utility of each transaction are calculated from transaction and profit data using Definitions 1.1 and 1.4, respectively. As an illustration, the utility and the total utility of transaction data in Table 1 and profit data in Table 2 are given in Table 3. Each row in Table 3 is divided into three groups, separated by colon (:). The first group contains items in the transaction, the second is the total utility of the transaction, and the third contains the utilities of items in the transaction. Notice that the transactions in the table have been sorted in descending order based on their total utilities.

Algorithm 1 The proposed high-utility itemset mining algorithm

```
1: function HUIM-BPSO-NOMUT(D, 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:     end for
6:     Sort transactions in D descendingly based on total utility
7:     Population P ← ∅ ▷ Phase 2: initializing and encoding
8:     for j ← 1 to pop_size do
9:         T ← Dequeue(D)
10:        Encode T into particle p⃗
11:        Generate velocity v⃗
12:        Compute fitness of p⃗: fit(p⃗) ▷ Phase 3: fit evaluation
13:        P ← P ∪ { (p⃗, v⃗, fit(p⃗)) }
14:    end for
15:    HUIs ← Copy(P)
16:    P_(b) ← Copy(P)
17:    p⃗_(g) ← FindBestParticle(P_(b))
18:    Compute constriction factor k ▷ Eq. (4)
19:    for i ← 1 to iterations do ▷ Phase 4: updating
20:        for each (p⃗, v⃗, fit(p⃗)) ∈ P, (p⃗_(b), v⃗_(b), fit(p⃗_(b))) ∈ P_(b) do
21:            Update velocity v⃗ ▷ Eqs. (3) and (5)
22:            Update particle p⃗ ▷ Eq. (6)
23:            Compute fitness of p⃗: fit(p⃗) ▷ Eq. (2)
24:            HUIs ← HUIs ∪ { p⃗ }
25:            if fit(p⃗) > fit(p⃗_(b)) then
26:                p⃗_(b) ← Copy(p⃗)
27:                p⃗_(g) ← FindBestParticle(P_(b))
28:            end if
29:        end for
30:    end for
31:    return HUIs
32: end function
```

Table 3
Transactions, utilities and total utility.

TId	Transaction (items : total utility : utilities)
T_5	a b d e : 57 : 25 4 10 18
T_6	a b c : 25 : 15 8 2
T_4	c d e : 23 : 4 4 15
T_2	a b c d : 22 : 5 7 6 4
T_7	d e : 22 : 10 12
T_3	a c d : 15 : 5 4 6
T_1	b c d : 7 : 1 2 4

4.2. Initialization and particle encoding

In the second phase, the particle swarm optimization method is initialized by forming its population. The population consists of *pop_size* number of particles, each representing a transaction in the database. A particle is formally defined as a vector $\vec{p}$ of dimension d, where d is the number of distinct items in all transactions, such that for a transaction T:

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

Note that only *pop_size* transactions with the largest total utilities are being represented in the population. As an illustration, transactions in Table 3 only involve five distinct items. Assuming *pop_size* $= 5$, Table 4 shows the encoding of transactions T_5, T_6, T_4, T_2, and T_7 from Table 3 as particles $\vec{p}_{(1)}$, $\vec{p}_{(2)}$, $\vec{p}_{(3)}$, $\vec{p}_{(4)}$, and $\vec{p}_{(5)}$, respectively.

Table 4
Encoding transactions as particles.

	a	b	c	d	e
$\vec{p}_{(1)}$	1	1	0	1	1
$\vec{p}_{(2)}$	1	1	1	0	0
$\vec{p}_{(3)}$	0	0	1	1	1
$\vec{p}_{(4)}$	1	1	1	1	0
$\vec{p}_{(5)}$	0	0	0	1	1

Table 5
Initializing particles' velocities.

	a	b	c	d	e
$\vec{v}_{(1)}$	0.343757	0.640539	0.636980	0.724399	0.719616
$\vec{v}_{(2)}$	0.521030	0.409739	0.092495	0.572762	0.503367
$\vec{v}_{(3)}$	0.838405	0.369433	0.644313	0.647657	0.709206
$\vec{v}_{(4)}$	0.336895	0.611416	0.768125	0.923636	0.409901
$\vec{v}_{(5)}$	0.359932	0.072850	0.858039	0.868211	0.133234

In this phase, we also initialize the velocity of each particle randomly. Similar to a particle, its velocity is a vector $\vec{v}$ of dimension d such that for all $i \in T$, $-v_{\max} \leq v_i \leq v_{\max}$. Table 5 shows an example of the initialization of particles' velocities, where $\vec{v}_{(i)}$ is the velocity of particle $\vec{p}_{(i)}$.

After initializing the velocity of the encoded particle, its fitness is evaluated, which will be elaborated on in Section 4.3. The particle, together with its velocity and fitness, is then put into the population P. Three things must be noted here. First, all itemsets in the original population generated from *pop_size* transactions with the largest total utilities form the initial high utility itemsets. Other high utility itemsets are generated as "descendants" of these initial itemsets. Second, a new population $P_{(b)}$ is created, which in the beginning is just a copy of the original population. This population $P_{(b)}$ is used to record the best (*i.e.*, fittest) "descendants" of each particle in P at each iteration. Third, the current best of all particles $\vec{p}_{(g)}$ is sought among those in $P_{(b)}$. This best particle is used later on for updating subsequent particles in Section 4.4.

4.3. Fitness evaluation

To determine the best particle, the fitness of each particle $\vec{p}$ is evaluated, which is defined by:

$$fit(\vec{p}) = u(X), \tag{2}$$

where $u(X)$ is the utility according to Definition 1.3 and X is an itemset containing precisely all items in particle $\vec{p}$. This fitness function has also been used by Kannimuthu and Premalatha [6] and Lin et al. [9]. As an illustration, the fitness of particle $\vec{p}_{(5)}$ in Table 4 is $fit(\vec{p}_{(5)}) = u(\{d, e\}) = u(\{d, e\}, T_5) + u(\{d, e\}, T_4) + u(\{d, e\}, T_7) = 28 + 19 + 22 = 69$. The reader is invited to verify that $\vec{p}_{(5)}$ is the fittest among particles in Table 4.

4.4. Updating

In this phase, all particles in P and their corresponding best pairs in $P_{(b)}$ are repeatedly updated as many as *iterations* times. In each iteration, the following are carried out on each particle:

1. Updating velocity. Let $\vec{x}(t)$ be the vector $\vec{x}$ at discrete time (iteration) t. The velocity $\vec{v}$ associated with a particle $\vec{p}$ is updated as follows [34]:

$$v_i(t+1) = k \cdot [v_i(t) + c_1 \cdot r_1 \cdot (p_{(b)i} - p_i(t)) + c_2 \cdot r_2 \cdot (p_{(g)i} - p_i(t))], \tag{3}$$

where:

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4 \cdot \varphi}\right|}, \varphi = c_1 + c_2, \text{ and } \varphi \geq 4. \tag{4}$$

The constriction factor k plays a balancing role between the global best particle $\vec{p}_{(g)}$ and the local best particle $\vec{p}_{(b)}$, whose value is determined by the selection of the individual factor c_1 and the social factor c_2 according to Eq. (4). Two random numbers $r_1, r_2 \in (0, 1)$ are generated in each iteration, which means that the two random numbers are used in updating all components of all particles in a particular iteration.
It should be noted that each velocity component in Eq. (3) must fall within a particular bound; this is known as the "$v_{\max}$ method" [35]. In this paper, we use the $v_{\max}$ method together with a piece-wise linear function as defined in [36], as follows:

$$v_i(t+1) = \begin{cases} v_{\max}, & \text{if } v_i(t+1) > v_{\max}, \\ v_i(t+1), & \text{if } -v_{\max} \leq v_i(t+1) \leq v_{\max}, \\ -v_{\max}, & \text{if } v_i(t+1) < -v_{\max}. \end{cases} \tag{5}$$

Example 2. As an illustration, let us select the individual factor $c_1 = 2$ and the social factor $c_2 = 2$. Consequently, $\varphi = c_1 + c_2 = 2 + 2 = 4$ and $k = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\cdot\varphi}\right|} = \frac{2}{\left|2-4-\sqrt{4^2-4\cdot4}\right|} = 1$. Let $r_1 = 0.639415$ and $r_2 = 0.387346$ and assume that $\vec{p}_{(b)} = [1, 0, 1, 0, 1]^\top$ and $\vec{p}_{(g)} = [1, 1, 1, 0, 1]^\top$. Table 6 shows the detailed calculation involved for updating the velocity associated with particle $\vec{p}_{(1)}$ in Table 5.

2. Updating particle. Based on its updated velocity, the particle $\vec{p}$ is updated as follows [29]:

$$p_i(t+1) = \begin{cases} 1, & \text{if } r < S(v_i(t+1)) = \dfrac{1}{1 + e^{-v_i(t+1)}}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where r is a random number generated for each component of each particle, and $S(\cdot)$ is the sigmoid function. Note that this update may change the configuration of the particle, which in turn may result representing a completely different itemset with a different total utility and, therefore, representing a new high utility itemset.

3. Evaluating the particle's fitness. Once a particle is updated, its fitness is re-evaluated using Eq. (2). 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 the particles in $P_{(b)}$ by invoking function FindBestParticle(), which is depicted in Algorithm 2.

Algorithm 2 Finding the best particle

```
1: function FindBestParticle(P): p⃗
2:     fit ← 0.0
3:     for each (q⃗, v⃗, fit(q⃗)) ∈ P do
4:         if fit(q⃗) > fit then
5:             fit ← fit(q⃗)
6:             p⃗ ← Copy(q⃗)
7:         end if
8:     end for
9:     return p⃗
10: end function
```

4.5. Time and space complexity

By inspecting Algorithm 1, it is straightforward that computation time is mostly spent on the for-loop in lines 19–30. Assuming

Table 6
Updating velocity $\vec{v}_{(1)}$ from Table 5.

i	Calculation	Result
1	$1 \cdot [0.343757 + 2 \cdot 0.639415 \cdot (1-1) + 2 \cdot 0.387346 \cdot (1-1)]$	0.343757
2	$1 \cdot [0.640539 + 2 \cdot 0.639415 \cdot (0-1) + 2 \cdot 0.387346 \cdot (1-1)]$	−0.638291
3	$1 \cdot [0.636980 + 2 \cdot 0.639415 \cdot (1-0) + 2 \cdot 0.387346 \cdot (1-0)]$	2.690502
4	$1 \cdot [0.724399 + 2 \cdot 0.639415 \cdot (0-1) + 2 \cdot 0.387346 \cdot (0-1)]$	−1.329123
5	$1 \cdot [0.719616 + 2 \cdot 0.639415 \cdot (1-1) + 2 \cdot 0.387346 \cdot (1-1)]$	0.719616

that function FindBestParticle() has time complexity $\mathcal{O}(pop\_size)$, then the time complexity of the for-loop is $\mathcal{O}(iterations \cdot pop\_size^2)$. Most of the time, computation spent on this main for-loop is much larger than that spent on the for-loop in lines 2–5, which has time complexity $\mathcal{O}(|D|)$. If not, the overall time complexity of the algorithm is $\mathcal{O}(iterations \cdot pop\_size^2 + |D|)$.

The space complexity of the algorithm, on the other hand, depends on the data structures retained all the time. The algorithm needs to store database D with space complexity $\mathcal{O}(|D|)$. It also stores two instances of population, P and $P_{(b)}$, each with space complexity $\mathcal{O}(pop\_size \cdot d)$, where d is the number of distinct items in the transaction database. This is because apart from its binary representation, a particle is also determined by its d velocities. Finally, the algorithm must store the high utility itemsets *HUIs*, which is originally of size *pop_size* but grows to $\mathcal{O}(iteration \cdot pop\_size)$ in the worst case. Note that *HUIs* only needs to store the binary representations of the particles, without their velocities. Overall, the space complexity of Algorithm 1 is $\mathcal{O}(iteration \cdot pop\_size + pop\_size \cdot d + |D|)$.

4.6. The proposed method compared with HUIM-BPSO$^+$

Both the proposed method and HUIM-BPSO$^+$ use the BPSO approach. In this section, we emphasize the differences between them. The first is in the pre-processing phase. HUIM-BPSO$^+$ initializes its population by empty transactions (represented by zero vectors), while the proposed method uses *pop_size* transactions with the highest utility. Furthermore, in this phase, HUIM-BPSO$^+$ also performs 1-HTWUIs (high-transaction-weighted utilization 1-itemsets) process [15] to prune unpromising items. An item is unpromising if its transaction weighted utility is less than the minimum utility threshold. Such items are removed from all transactions. Only items in 1-HTWUIs are used in all transactions and in subsequent parts of the algorithm. The proposed method uses no such process.

The second is in updating velocity. The proposed method combines the $v_{\max}$ method with a piece-wise linear function, and uses the constriction factor k to balance between the global search result $\vec{p}_g$ and the local search result $\vec{p}_b$. HUIM-BPSO$^+$, on the other hand, does not make use of the $v_{\max}$ method with a piece-wise linear function, and the constriction factor k is a just constant.

The third and the most important is in the use of minimum utility threshold. The proposed method generates itemsets according to the BPSO algorithm (which, in the worst case, are as many as $iteration \cdot pop\_size$) and outputs them as a list sorted by the utility. The user may impose a minimum utility threshold on this list, but this step is a post-processing one, in the sense that this imposition does not change the output of the method. The application of minimum utility threshold, on the other hand, is an integral part of HUIM-BPSO$^+$. In each iteration, an itemset is stored or discarded based on whether its utility exceeds the minimum utility threshold or not. Once an itemset is discarded, all its "descendants", namely those obtained from subsequent updates of the itemset, cannot simply be ignored because these "descendants" may still be found during the iterative process by modifying other particles. In our experience, some of these "descendants" may have high utility.

Table 7
Characteristics of the datasets used in the experiments.

Dataset	Size	$\|I\|$	$\|D\|$	Avg	Total utility	Density (%)
Chess	335	75	3,192	37	1,484,915	49.33
Connect	9,309	129	67,557	43	32,782,315	33.33
Mushroom	558	119	8,124	23	2,217,907	19.33
Accident 10%	34,678	468	340,183	33	12,659,954	7.22
Foodmart	176	1,559	4,141	4	12,011,023	0.28

5. Experiments, result and analysis

To evaluate the proposed method, we perform several series of experiments. This section specifies the datasets we use in the experiments, presents their setup and results, and provides discussion and analysis.

5.1. Datasets

We use five datasets in our experiments, four from Frequent Itemset Mining Dataset (fimi) repository: chess, connect, mushroom, and accident; and one from Microsoft 1996: foodmart. The characteristics of the datasets used in the experiments are shown in Table 7. Size represents file size in KB, $|I|$ represents the number of distinct items, $|D|$ represents the total number of transactions, Avg represents the average length of transactions, and Density represents the ratio of the average number of items in all transactions to the number of distinct items.

Those datasets are a collection of datasets generally used in frequent itemset research. The internal utilities are determined using a uniform distribution in the range of $[1, \ldots, 15]$, which means the minimum quantity is one, and the maximum quantity is fifteen. The external utilities or profits, on the other hand, are generated using a half-normal distribution. The total utilities of all datasets are listed in the sixth column of Table 7. Because the number of transactions in the accident dataset is vast, only 10% of the transactions are used. The original datasets, together with the corresponding internal and external utilities we used in the experiments, can be found at http://www.github.com/ridowati/arhuim.

5.2. Experimental setup

The proposed method, HUIM-BPSO-nomut (BPSO-nomut for short), will be compared with ten previous algorithms. Firstly, it is compared with high-utility itemset mining algorithms without minimum utility threshold, but with a pre-determined number of itemsets, namely TKU (Top-K utility), TKO (Top-K utility in One phase) [11], and kHMC [24]. Throughout the paper, the TKO used is TKO$_{Base}$ without optimization, as found in the SPMF library [37]. The three algorithms are used as a basis of comparison because they are classic algorithms, where the user does not specify the minimum utility threshold before the search process is carried out. The number of highest utility itemsets selected, K, is determined by the user.

Secondly, the proposed method is compared with other computational intelligence methods that do not use a minimum utility threshold. They are HUPE$_{umu}$-GRAM$^+$ (GA-tree for short) [9],

which is based on GA with OR/NOR tree, and HUIM-BPSO$^+$ (BPSO-tree for short) [9], which is based on BPSO with OR/NOR tree. Each comparison with GA-tree and BPSO-tree involves five experiments, from which four quantities are measured, namely the number of high utility itemsets, total utility, running time, and memory consumption.

Thirdly, the proposed method is also compared with HUI-Miner [19], which is a deterministic algorithm with minimum utility threshold, as well as with several computational intelligence methods that use minimum utility threshold, namely GA-tree, BPSO-tree, and HUIM-ACS [10]. Lastly, the proposed method is compared with three newly proposed bio-inspired algorithms [31], namely, Bio-HUIF-GA, Bio-HUIF-PSO, and Bio-HUIF-BA.

The proposed method and the previous methods are implemented based on the open-source data mining library SPMF [37], except for HUIM-ACS and kHMC. All algorithms are implemented in Java and run 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. All implementations can be found at http://www.github.com/ridowati/arhuim.

5.3. Result and analysis

The proposed method vs TKU, TKO, and kHMC. In this part, the proposed method is compared with TKU, TKO, and kHMC to find out the strengths and weaknesses of computational intelligence and deterministic methods. Each dataset is experimented on with TKU, TKO, and kHMC by varying K, the number of highest utility itemsets produced, up to a maximum value still doable in our equipment. The proposed method proceeds as usual, producing high utility itemsets without restriction and is set with the population size 20 and the number of iterations 10,000. Once the proposed method finishes, the top (highest utility) K itemsets it produces are collected.

For chess, connect, and accident datasets, TKU produces the top K high utility itemsets as desired. For the mushroom dataset, K can be set up to 70,005, while for the foodmart dataset, it can be set up to 8002. For these two datasets, setting larger values of K results in the same numbers of high utility itemsets, namely 70,005 for the mushroom dataset and 8002 for the foodmart dataset. For TKO and kHMC, itemsets can be generated for all datasets with any value of K. To assess the strengths and weaknesses of the four methods, we compare the "quality" of the top K high utility itemsets they produced, measured by the total utility of those itemsets. The first row of Fig. 1 depicts the total utility obtained by TKU, TKO, kHMC, and the proposed method for each dataset.

For the chess dataset, of the three algorithms, TKO has the best performance. TKU requires large storage space for candidates and the top K itemsets, especially for K greater than 10. kHMC has a weakness for large K, namely a significant performance reduction in time and memory consumption, which has also been observed by [38]. In this case, the percentages of total utility obtained by the proposed method compared with TKU, TKO, and kHMC for $K = 1$, 10, 100, 1000, and 3000 are 100%, 100%, 99.93%, 99.02%, and 96.43%, respectively. For the connect dataset, the percentages are 100%, 100%, 99.97%, 99.13%, and 96.08%, respectively; for the mushroom dataset, those percentages are 100%, 100%, 98.05%, 84.28%, and 73.62%; for the accident dataset, they are 100%, 100%, 98.05%, 84.28%, and 73.65%; while for the foodmart dataset, the percentages are 96.41%, 96.33%, 79.93%, 80.21%, and 54.44%.

The running time of the four methods for each dataset is shown in the second row of Fig. 1. The proposed method is much faster than TKU but much slower than TKO. The proposed method is slower than TKO for all datasets except for the connect dataset. The running time of kHMC is faster than others almost for all datasets and all K values except for mushroom when $K = 3000$. Population size and the number of iterations affect the running time of the proposed method, while the number of desired itemsets holds only negligible influence.

The proposed method consumes less memory than others except for the foodmart dataset, as shown in the third row of Fig. 1. TKU and TKO consume much memory, except for the foodmart dataset, where the proposed method needs more memory than the others. kHMC consumes more memory than others for connect, mushroom, and accident datasets, and sometimes even consumes more memory than is available; for instance, when processing chess and connect datasets when $K = 3000$.

Our experiments indicate that for the same values of K, TKO is better than TKU and kHMC. TKU requires more time to obtain high utility itemsets compared to TKO and kHMC, while kHMC is faster than the others but requires more memory. The strength of the proposed method compared with TKU is that it can produce more itemsets with relatively less memory and less running time. Compared with TKO, the strength of the proposed method is that it can produce the same number of itemsets, especially for small values of K with less memory. Compared with kHMC, the strength of the proposed method is that it can produce the same number of itemsets, and usually with less memory. In conclusion, although based on computational intelligence approach, the proposed method can be used as an alternative in obtaining high utility itemsets, especially for datasets with high density, like the connect dataset, with less memory.

The proposed method vs BPSO-tree and GA-tree—without minimum utility threshold. In this part, three computational intelligence methods are compared by fixing them to produce the same number of high utility itemsets for each dataset. For this purpose, GA-tree and BPSO-tree are adjusted by setting their minimum utility threshold to 0 and then collecting a pre-determined number of highest utility itemsets they produce. For all three methods, the desired number of high utility itemsets is 100, the population size is 20, and the number of iterations is 10,000. To assess the strengths and weaknesses of the three methods, we compare the "quality" of the high utility itemsets they produced, measured by the total utility of those itemsets.

From the 100 highest utility itemsets, the total utility of GA-tree, BPSO-tree, and the proposed method for each dataset is depicted in the first row of Fig. 2. It can be observed that the proposed method always produces itemsets with the largest total utility for all datasets. For accident and connect datasets, the total utility produced is even significantly larger. The running time of the three methods can be seen in the second row of Fig. 2. Except for the connect dataset, the proposed method is faster than others and for some of them significantly so. The third row of the figure depicts the memory consumption of the three methods. The proposed method mostly has a better memory consumption than BPSO-tree, but worse than GA-tree. Nevertheless, since all three algorithms are optimization-based, their memory consumption is quite low and depends mostly on the population size and the number of iterations.

It is easier in practice for the user to determine the number of desired high utility itemsets than the minimum utility threshold. The three algorithms can always produce the number of itemsets with the highest utility. This is different if the acquisition of the high utility itemsets is based on the minimum utility threshold. If the pre-determined minimum utility threshold is not correct, the desired itemsets may not be obtained, or too many itemsets are obtained.

In conclusion, the proposed method always produces better quality itemsets than GA-tree or BPSO-tree, within relatively

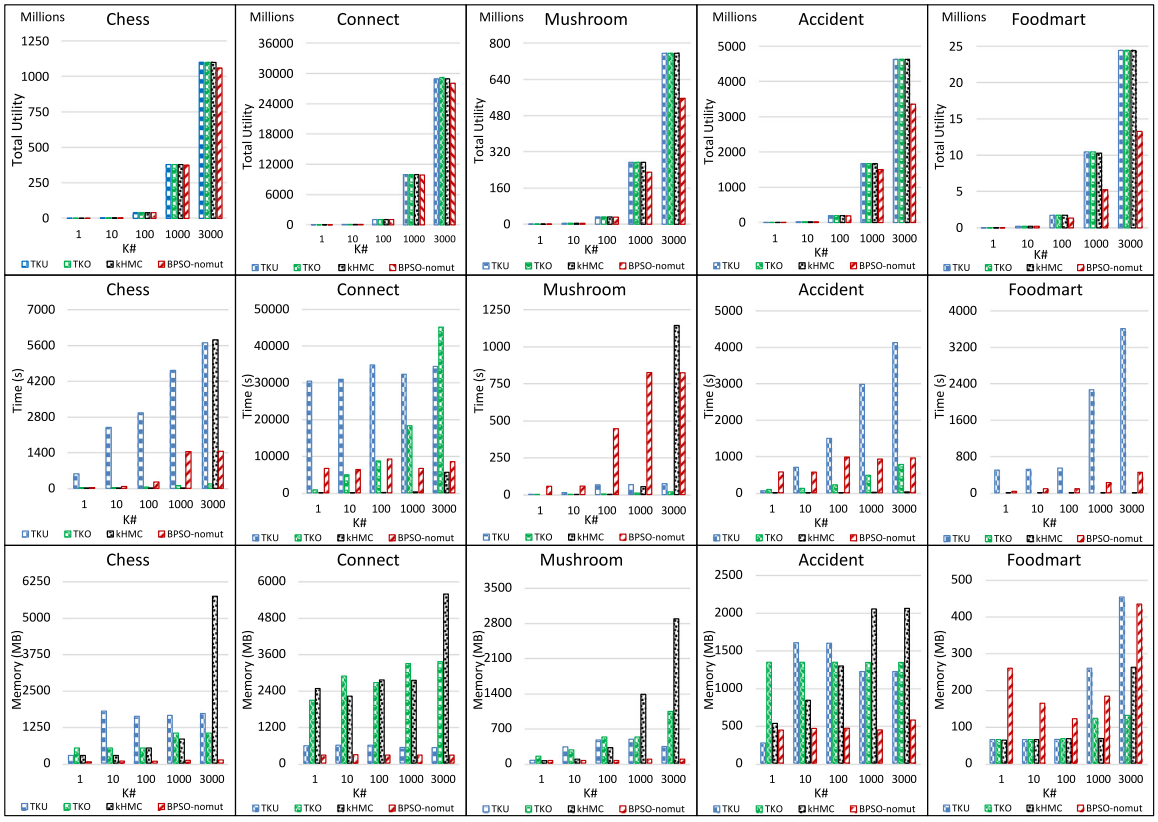


Fig. 1. Comparing TKU, TKO, kHMC, and the proposed method showing the total utility (first row), the running time (second row), and the memory consumption (third row).

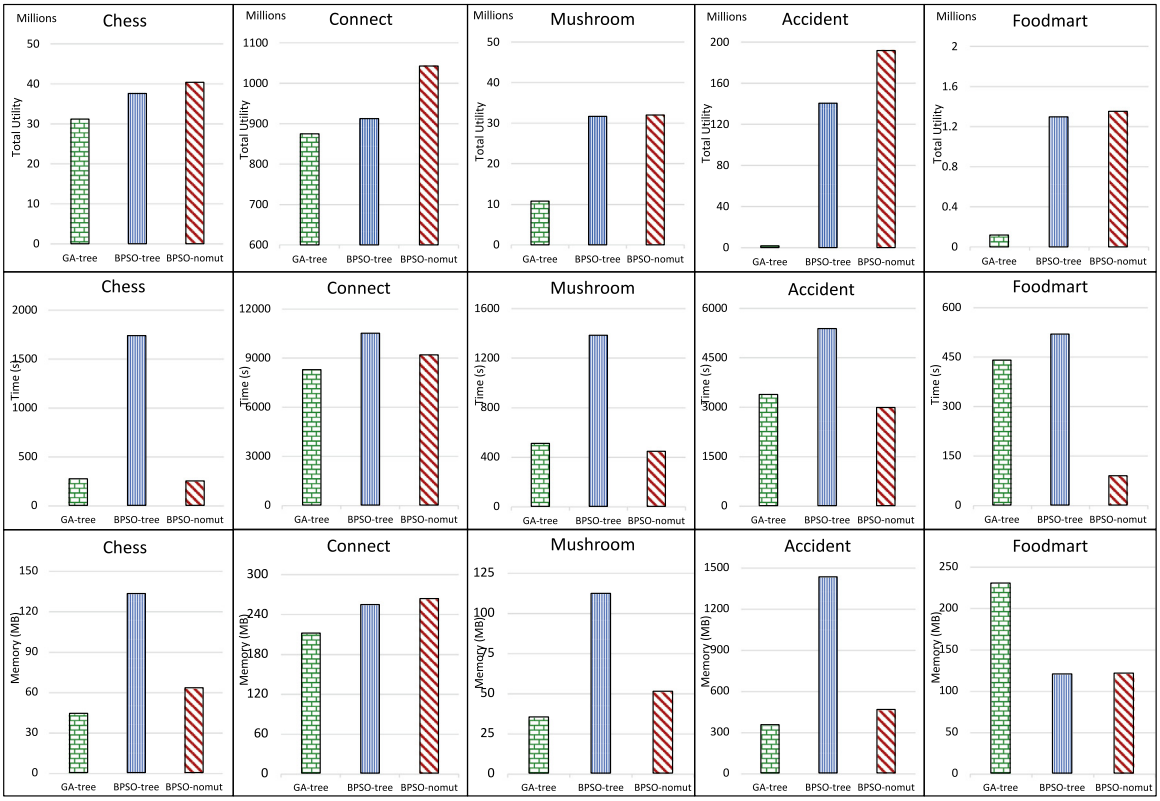


Fig. 2. Comparing GA-tree, BPSO-tree and the proposed method without minimum utility threshold showing the total utility (first row), the running time (second row), and the memory consumption (third row).

Fig. 3. Comparing GA-tree, BPSO-tree, HUIM-ACS, and the proposed method with minimum utility threshold showing the number of itemsets (first row), the total utility (second row), the running time (third row), and the memory consumption (fourth row). In the fifth row, the four algorithms are put in the context of deterministic algorithm HUI-Miner, which produces all itemsets.

better running time and comparatively similar memory consumption. The 100 highest utility itemsets produced by the proposed method consistently and sometimes significantly have more total utility.

The proposed method vs BPSO-tree, GA-tree, and HUIM-ACS—with minimum utility threshold. In this part, four computational intelligence methods are compared by fixing them to produce high utility itemsets using a minimum utility threshold. For this purpose, once the proposed method finishes generating high utility itemsets, a post-processing step is added to collect all itemsets whose utility is more than a pre-determined minimum utility threshold. The minimum utility threshold is expressed as a percentage (δ) of the total utility (TU) in each dataset, *i.e.*, $\delta \cdot TU$. To assess the strengths and weaknesses of the four methods, we compare the number of high utility itemsets they produced as well as the total utility of those itemsets.

Fig. 3 depicts the results of experiments for this comparison, showing the number of itemsets, total utility, running time, and memory consumption of the four methods. Note that the minimum utility threshold used for each dataset is different, reflected

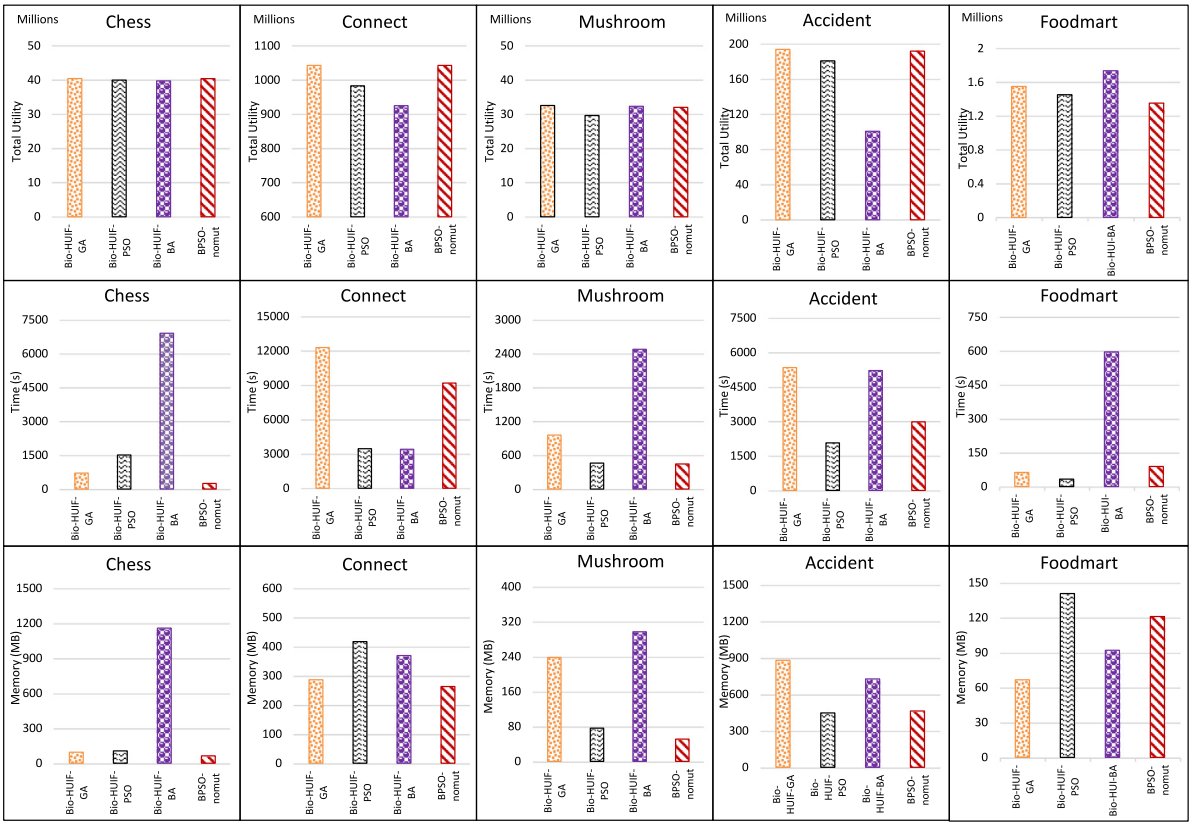


Fig. 4. Comparing Bio-HUIF-GA, Bio-HUIF-PSO, Bio-HUIF-BA, and the proposed method when they are fixed to produce the top $K = 100$ high utility itemsets, showing the total utility (first row), the running time (second row), and the memory consumption (third row).

by the different δ, which we adopt from previous studies [7–9]. It is evident from the first and second rows of the figure that the proposed method consistently produces more itemsets and with more total utility than GA-tree as well as BPSO-tree, even though the proposed method does not use a minimum utility threshold. For foodmart and mushroom datasets, GA-tree cannot even generate any high utility itemset for some of the minimum utility thresholds specified. HUIM-ACS can produce more itemsets than GA-tree and BPSO-tree, but not always more than the proposed method. For the foodmart dataset, HUIM-ACS produces more itemsets, while for the connect dataset, the proposed method produces more. As for the other datasets, at certain minimum utility thresholds, HUIM-ACS produces the same number of itemsets as the proposed method.

The running time and the memory consumption of the four methods are depicted in the third and fourth rows of Fig. 3. The proposed method is relatively faster than BPSO-tree except for the chess dataset. GA-tree is almost always faster than the proposed method, but always worst in terms of the number of itemsets and the total utility produced. HUIM-ACS requires more time than the others for almost all datasets except for the mushroom dataset, where it performs similarly to the proposed method. However, apart from the mushroom dataset, the running time difference for other datasets is very long. For the connect dataset, with a minimum utility threshold of 29.5% and 29.7%, the process must be terminated because the time needed is very long. The proposed method requires relatively more memory than GA-tree or BPSO-tree, especially for dense datasets.

The proposed method does not perform a search for high-transaction-weighted utilization 1-itemsets (1-HTWUIs), as is done in BPSO-tree or GA-tree [15]. BPSO-tree and GA-tree limit themselves to use only items in 1-HTWUIs, thus eliminating items that have transaction-weighted-utility below the minimum utility threshold. Without restricting to 1-HTWUIs, the proposed method can produce more itemsets and with more total utility for various minimum utility thresholds.

To get an idea of how many high utility itemsets are supposed to be produced, a traditional and deterministic algorithm HUI-Miner [19] is used as a basis of comparison. The fifth row of Fig. 3 shows the percentage of high utility itemsets produced by the four methods out of those produced by HUI-Miner. The proposed method is always better than the GA-tree and BPSO-tree for all datasets and can reach more than 90%, except for connect and foodmart datasets. As the minimum utility threshold is set larger, the proposed method gets closer to HUI-Miner. HUIM-ACS can produce the same number of itemsets as HUI-Miner for the mushroom dataset. The proposed method, on the other hand, can produce more itemsets than HUIM-ACS for the connect dataset.

The proposed method vs bio-inspired algorithms. In this part, we compare the proposed method with three newly proposed bio-inspired algorithms, Bio-HUIF-GA, Bio-HUIF-PSO, and Bio-HUIF-BA [31]. These algorithms require a pre-determined minimum utility threshold, and, for comparison, is set to 0. The four methods are compared by inspecting the number of itemsets and total utility they produced based on top-K (where $K = 100$) and the minimum utility threshold. The minimum utility thresholds used are the same as the previous experiments.

Fig. 4 compares the four algorithms when they are set to produce the top $K = 100$ high utility itemsets. The first row shows the total utility of those top 100 itemsets. Out of the five datasets, the proposed method obtains the same results as Bio-HUIF-GA for four datasets, and the results are always better than the other two bio-inspired algorithms. However, the proposed method obtains the lowest result for the foodmart dataset.

The running time of the algorithms can be seen in the second row of Fig. 4. The proposed method is the fastest for chess and

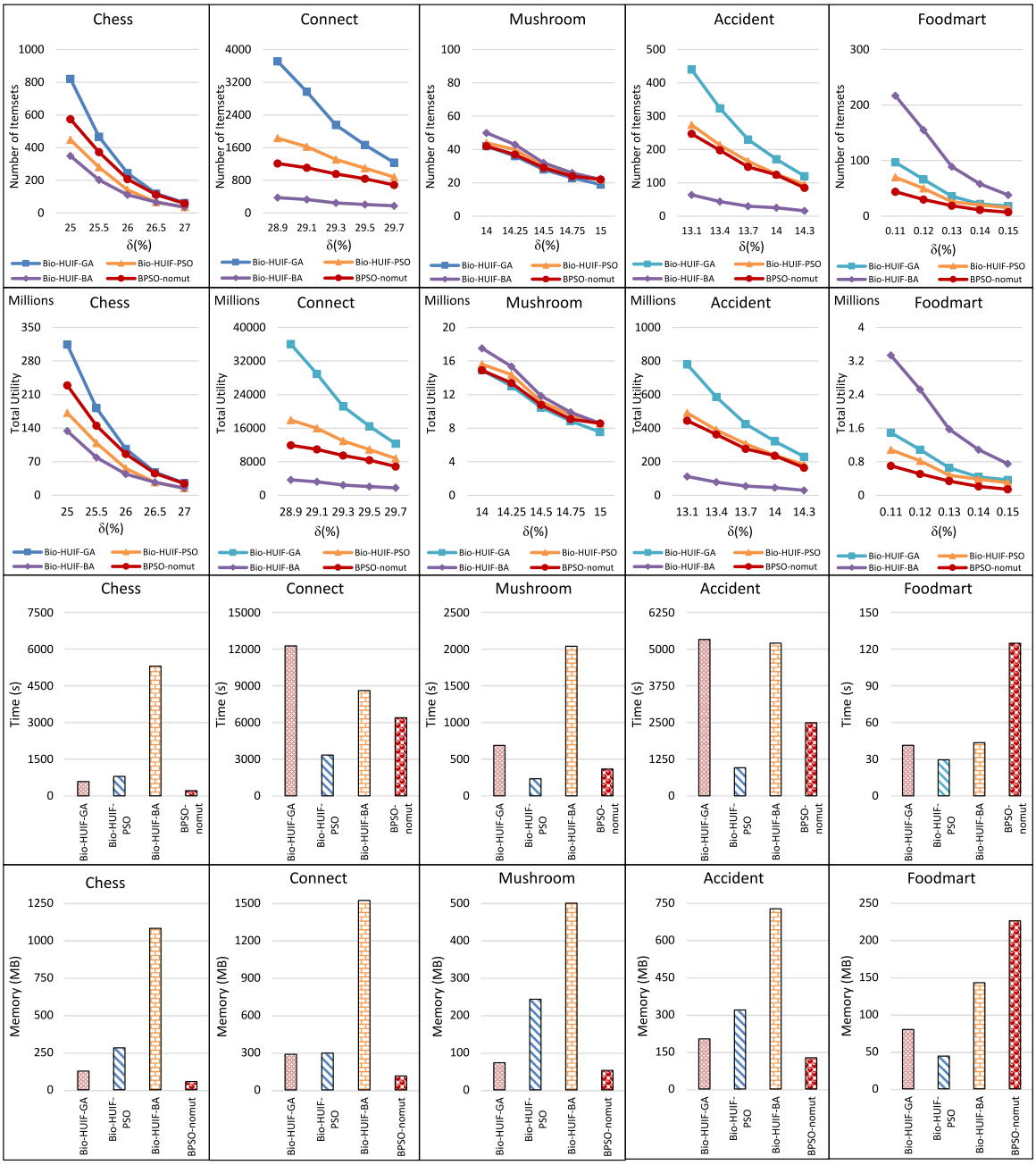


Fig. 5. Comparing Bio-HUIF-GA, Bio-HUIF-PSO, Bio-HUIF-BA, and the proposed method when they are set to produce all high utility itemsets and then a minimum utility threshold is imposed, showing the number of itemsets (first row), the total utility (second row), the running time (third row), and the memory consumption (fourth row).

mushroom datasets, is faster than Bio-HUIF-GA for the connect dataset, is faster than Bio-HUIF-GA and Bio-HUI-BA for the accident dataset, and is faster than Bio-HUIF-BA for the foodmart dataset. The proposed method requires less memory than the three bio-inspired algorithms, except for the foodmart dataset, in which it requires more than Bio-HUIF-GA and Bio-HUIF-BA.

Fig. 5, on the other hand, compares the four algorithms when they are set to produce all high utility itemsets, and then a minimum utility threshold is imposed. For the bio-inspired algorithms, this is done by setting the minimum utility threshold to 0. The number of itemsets and their total utility are depicted in the first and second rows of Fig. 5. Although the proposed method cannot produce the highest number of itemsets or total utility compared to all three bio-inspired algorithms, it is higher than one or two of them for some datasets. For the chess dataset, the proposed method produces more than Bio-HUIF-PSO and Bio-HUIF-BA. For the connect dataset, it produces more than Bio-HUIF-BA, while for the mushroom dataset, all four produce almost the same number. For the accident dataset, the proposed method obtains more than Bio-HUIF-BA and is close to Bio-HUIF-PSO.

The running time of the four algorithms is depicted in the third row of Fig. 5. The proposed method has the longest running time for the foodmart dataset. For connect and accident datasets, it is slower than Bio-HUIF-PSO but faster than Bio-HUIF-GA and Bio-HUIF-BA, and for chess and mushroom datasets, it is the fastest. The memory consumption of the four algorithms is depicted in the fourth row. The memory consumption of the proposed method is the least for four datasets: chess, connect, accident, and mushroom; but the most for the foodmart dataset.

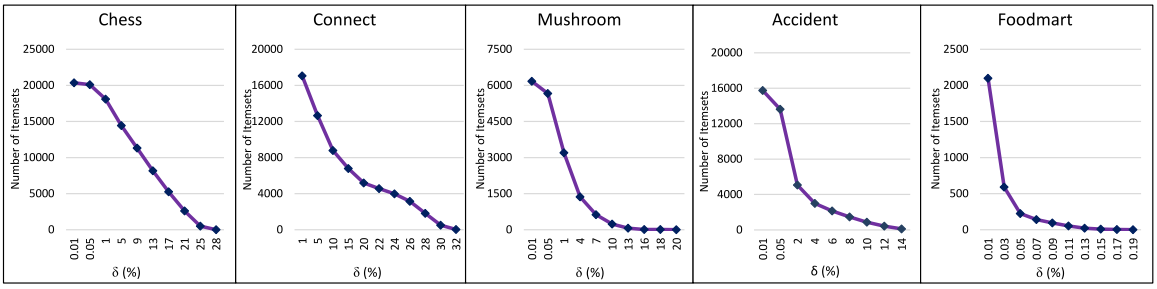


Fig. 6. The effect of varying the minimum utility threshold on the number of itemsets produced in each dataset.

In conclusion, the proposed method can produce similar numbers of itemsets to the bio-inspired algorithms. The memory usage of the proposed method is less compared to the three algorithms, and its running time is also comparatively faster. The weakness of the proposed method is evident in the case of the foodmart dataset, which is sparse. In this case, the proposed method is the worst in terms of the number of itemsets, total utility, running time, and memory consumption.

Discussion. To determine the "correct" minimum utility threshold, detailed knowledge about the characteristics of the data being processed must be obtained, and this is not easy. Different datasets have different minimum utility thresholds. Experiments on the five datasets show that the range of the minimum utility threshold widely varies. The user must, therefore, perform trial and error to determine this minimum utility threshold to obtain the desired number of itemsets. Fig. 6 shows a range of used minimum utility thresholds along with the number of itemsets obtained for each dataset. For the chess dataset, for instance, the user may determine the minimum utility threshold value between 0.01% until 28% and observe how the produced number of itemsets changes. In general, the higher the minimum utility threshold, the less the number of itemsets produced. Nevertheless, those itemsets have higher utility and are likely more useful to the user compared to those with lower utility.

There are many ways to assess the quality of HUIM algorithms, and we believe that the total utility obtained is as, if not more, important, as the number of itemsets produced. The total utility corresponds to the profit accrued if a collection of itemsets is selected or purchased. The user cannot determine the minimum utility threshold arbitrarily since it must be in the range shown in Fig. 6 to be meaningful. For very sparse datasets, such as the foodmart dataset, the minimum utility threshold must be very small. Among the five datasets, there is no dataset with a minimum utility threshold above 50%. These indicate that it is not easy to determine the minimum utility threshold; it cannot be made up arbitrarily in advance since it depends on the user's knowledge of the dataset. Therefore, the proposed method has a vital role in overcoming the problem of determining the minimum utility threshold. Furthermore, the determination of the initial population is an essential factor for optimization-based algorithms, such as genetic algorithm and binary particle swarm optimization. It is evident that restricting only to 1-HTWUIs does not make the produced itemsets of better quality.

Our experiments have also shown that:

1. For some datasets, the proposed method can generate itemsets with the same total utility as TKU for a specified number of itemsets. For other datasets, however, TKU produces itemsets with higher total utility. These indicate that the proposed method does not always manage to generate the highest utility itemsets. However, it should be noted that the proposed method completes in better or comparable running time and memory consumption in all datasets and that for some datasets, TKU may take unreasonable running time to complete.
2. In our experiments, TKO can produce itemsets according to the desired number of top-K; when the desired number K is reached, the itemsets and the total utility produced by both TKO and TKU are the same. The proposed method has not been able to achieve this. As with TKU, TKO also can produce the K highest utility itemsets for the connect dataset but with a very long running time.
3. kHMC has an advantage in speed, but the required memory is enormous. The proposed method, on the other hand, requires much less memory and produce almost similar itemsets.
4. The proposed method is always better than the other three computational intelligence methods when used with or without minimum utility threshold within relatively better running time and comparatively similar memory consumption. We suggest that this is because the proposed method is not restricted to only use items in 1-HTWUIs and applies the $v_{\max}$ method together with a piece-wise linear function when updating particles.
5. The proposed method can produce similar numbers of itemsets as the bio-inspired algorithms, but not for all datasets and minimum utility thresholds. The proposed method also has less running time and less memory consumption compared to the bio-inspired algorithms. However, the proposed method has a weakness when dealing with sparse datasets, especially the foodmart dataset.

6. Conclusion

We have proposed a method based on binary particle swarm optimization to obtain high utility itemsets without a minimum utility threshold. The advantage of the proposed method is that the process of obtaining high utility itemsets becomes more optimal because it does not need to set the minimum utility threshold in advance. Besides, if there are changes due to additions or updates to the dataset, the proposed method can be used immediately without adjusting the minimum utility threshold. We have also shown that the proposed method is superior to other computational intelligence methods in terms of the number of itemsets and the total utility produced.

On the other hand, for datasets with small average transaction length or with small number of transactions, even though the proposed method produces the same number of itemsets, it only produces itemsets of lower total utility compared to TKU, TKO, and kHMC. To overcome this problem, we expect that the velocity update in the BPSO-based method can be tuned differently, for instance, by using a probabilistic velocity approach. Using this approach, updating velocity for bit '0' will be treated differently from that for bit '1'. We plan to investigate other computational intelligence methods based on swarm intelligence, such as ant or bee colony, to address this weakness of the proposed method.

Acknowledgments

This research was supported by the doctoral dissertation grant of the Ministry of Research, Technology and Higher Education of the Republic of Indonesia, grant no. 109/SP2H/LT/DRPM/2018.

References

[1] X. Yan, C. Zhang, S. Zhang, Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support, Expert Syst. Appl. 36 (2009) 3066–3076, http://dx.doi.org/10.1016/j.eswa.2008.01.028.
[2] R.J. Kuo, C.M. Chao, Y.T. Chiu, Application of particle swarm optimization to association rule mining, Appl. Soft Comput. 11 (1) (2011) 326–336, http://dx.doi.org/10.1016/j.asoc.2009.11.023.
[3] M. Gupta, Application of weighted particle swarm optimization in association rule mining, Int. J. Comput. Sci. Inform. 1 (3) (2012) 69–74.
[4] R.J. Kuo, C.W. Shih, Association rule mining through the ant colony system for national health insurance research database in Taiwan, Comput. Math. Appl. 54 (11–12) (2006) 1303–1318, http://dx.doi.org/10.1016/j.camwa.2006.03.043.
[5] V. Mangat, Swarm intelligence based technique for rule mining in the medical domain, Int. J. Comput. Appl. 4 (1) (2010) 19–24.
[6] S. Kannimuthu, K. Premalatha, Discovery of high utility itemsets using genetic algorithm, Int. J. Eng. Technol. 5 (6) (2013) 4866–4880, http://dx.doi.org/10.1080/08839514.2014.891839.
[7] J.C.-W. Lin, L. Yang, P. Fournier-Viger, J. Frnda, L. Sevcik, M. Voznak, An evolutionary algorithm to mine high-utility itemsets, Adv. Electr. Electr. Eng. 13 (4) (2015) 392–398, http://dx.doi.org/10.15598/aeee.v13i4.1474.
[8] J.C.-W. Lin, L. Yang, P. Fournier-Viger, J.M.T. Wu, T.P. Hong, L.S.L. Wang, J. Zhan, Mining high-utility itemsets based on particle swarm optimization, Eng. Appl. Artif. Intell. 55 (2016) 320–330, http://dx.doi.org/10.1016/j.engappai.2016.07.006.
[9] J.C.W. Lin, L. Yang, P. Fournier-Viger, T.P. Hong, M. Voznak, A binary PSO approach to mine high-utility itemsets, Soft Comput. 21 (17) (2017) 5103–5121, http://dx.doi.org/10.1007/s00500-016-2106-1.
[10] J.M.-T. Wu, J. Zhan, J.C.-W. Lin, An ACO-based approach to mine high-utility itemsets, Knowl.-Based Syst. 116 (2016) 102–113, http://dx.doi.org/10.1016/j.knosys.2016.10.027.
[11] V.S. Tseng, C.W. Wu, P. Fournier-Viger, P.S. Yu, Efficient algorithms for mining Top-K high utility itemsets, IEEE Trans. Knowl. Data Eng. 28 (1) (2016) 54–67, http://dx.doi.org/10.1109/TKDE.2015.2458860.
[12] H. Yao, H.J. Hamilton, C.J. Butz, 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, 2004, pp. 482–486, http://dx.doi.org/10.1137/1.9781611972740.51.
[13] H. Yao, H.J. Hamilton, Mining itemset utilities from transaction databases, Data Knowl. Eng. 59 (2006) 603–626, http://dx.doi.org/10.1016/j.datak.2005.10.004.
[14] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the 20th International Conference on Very Large Data Bases, Vol. 1215, 1994, pp. 487–499.
[15] Y. Liu, W.-k. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, in: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'05), Springer-Verlag Berlin, Heidelberg, 2005, pp. 689–695, http://dx.doi.org/10.1007/11430919_79.
[16] A. Erwin, R.P. Gopalan, N.R. Achuthan, CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach, in: 7th IEEE International Conference on Computer and Information Technology (CIT 2007), 2007, pp. 71–76, http://dx.doi.org/10.1109/CIT.2007.120.
[17] V.S. Tseng, C.W. Wu, B.E. Shie, P.S. Yu, UP-Growth: an efficient algorithm for high utility itemset mining, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 253–262, http://dx.doi.org/10.1145/1835804.1835839.
[18] C.-W. Lin, T.-P. Hong, W.-H. Lu, An effective tree structure for mining high utility itemsets, Expert Syst. Appl. 38 (6) (2011) 7419–7424, http://dx.doi.org/10.1016/j.eswa.2010.12.082.
[19] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM), 2012, pp. 55–64, http://dx.doi.org/10.1145/2396761.2396773.
[20] J. Liu, K. Wang, B.C.M. Fung, Mining high utility patterns in one phase without generating candidates, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1245–1257, http://dx.doi.org/10.1109/TKDE.2015.2510012.
[21] P. Fournier-Viger, C.W. Wu, S. Zida, V.S. Tseng, FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8502 LNAI, Springer International Publishing, 2014, pp. 83–92, http://dx.doi.org/10.1007/978-3-319-08326-1_9.
[22] S. Zida, P. Fournier-Viger, J.C.W. Lin, C.W. Wu, V.S. Tseng, EFIM: a fast and memory efficient algorithm for high-utility itemset mining, Knowl. Inf. Syst. 51 (2) (2017) 595–625, http://dx.doi.org/10.1007/s10115-016-0986-0.
[23] C. Zhang, G. Almpanidis, W. Wang, C. Liu, An empirical evaluation of high utility itemset mining algorithms, Expert Syst. Appl. 101 (2018) 91–115, http://dx.doi.org/10.1016/j.eswa.2018.02.008.
[24] Q.H. Duong, B. Liao, P. Fournier-Viger, T.L. Dam, An efficient algorithm for mining the Top-K high utility itemsets, using novel threshold raising and pruning strategies, Knowl.-Based Syst. 104 (2016) 106–122, http://dx.doi.org/10.1016/j.knosys.2016.04.016.
[25] P. Fournier-Viger, Z. Li, J.C.W. Lin, R.U. Kiran, H. Fujita, Efficient algorithms to identify periodic patterns in multiple sequences, Inform. Sci. 489 (2019) 205–226, http://dx.doi.org/10.1016/j.ins.2019.03.050.
[26] P. Fournier-Viger, Y. Zhang, J.C.W. Lin, H. Fujita, Y.S. Koh, Mining local and peak high utility itemsets, Inform. Sci. 481 (2019) 344–367, http://dx.doi.org/10.1016/j.ins.2018.12.070.
[27] U. Yun, D. Kim, E. Yoon, H. Fujita, Damped window based high average utility pattern mining over data streams, Knowl.-Based Syst. 144 (2018) 188–205, http://dx.doi.org/10.1016/j.knosys.2017.12.029.
[28] L.T.T. Nguyen, V.V. Vu, M.T.H. Lam, T.T.M. Duong, L.T. Manh, T.T.T. Nguyen, B. Vo, H. Fujita, An efficient method for mining high utility closed itemsets, Inform. Sci. 495 (2019) 78–99, http://dx.doi.org/10.1016/j.ins.2019.05.006.
[29] J. Kennedy, R.C. Eberhart, 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, 1997, pp. 4–8, http://dx.doi.org/10.1109/ICSMC.1997.637339.
[30] J.C.-W. Lin, W. Gan, P. Fournier-Viger, T.P. Hong, V.S. Tseng, Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowl.-Based Syst. 96 (2015) 171–187, http://dx.doi.org/10.1016/j.knosys.2015.12.019.
[31] W. Song, C. Huang, Mining high utility itemsets using bio-inspired algorithms: A diverse optimal value framework, IEEE Access 6 (2018) 19568–19582, http://dx.doi.org/10.1109/ACCESS.2018.2819162.
[32] C.F. Ahmed, S.K. Tanbeer, B. Jeong, Y. Lee, Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans. Knowl. Data Eng. 21 (12) (2009) 1708–1721, http://dx.doi.org/10.1109/TKDE.2009.46.
[33] S. Zida, P. Fournier-Viger, J.C.W. Lin, C.W. Wu, V.S. Tseng, EFIM: A highly efficient algorithm for high-utility itemset mining, in: LNCS, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9413, 2015, pp. 530–546, http://dx.doi.org/10.1007/978-3-319-27060-9_44.
[34] M. Clerc, The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3, CEC 1999, 1999, pp. 1951–1957, http://dx.doi.org/10.1109/CEC.1999.785513.
[35] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS'95), 1998, pp. 39–43, http://dx.doi.org/10.1109/MHS.1995.494215.
[36] M.F. Tasgetiren, Y.-C. Liang, A binary particle swarm optimization algorithm for lot sizing problem, J. Econ. Soc. Res. 5 (2) (2004) 1–20.
[37] P. Fournier-Viger, C.W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, H.T. Lam, The SPMF open-source data mining library version 2, in: Proceeding 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III, in: LNCS, vol. 9853, Springer, 2016, pp. 36–44, http://dx.doi.org/10.1007/978-3-319-46131-1_8.
[38] S. Krishnamoorthy, A comparative study of Top-K high utility itemset mining methods, in: P. Fournier-Viger, J.C.W. Lin, R. Nkambou, B. Vo, V.S. Tseng (Eds.), High-Utility Pattern Mining: Theory, Algorithms and Applications, Springer, 2019, pp. 47–74, http://dx.doi.org/10.1007/978-3-030-04921-8_2.