Hypergraph structures for parallel integration of databases into CRM

Oleksandr Tymchenko[1,*,†], Bohdana Havrysh[1,*,†], Orest Khamula[1,*,†] and Volodymyr Chorniak[1,†]

[1] *Lviv Polytechnic National University, Stepana Bandery Street, 12, Lviv, 79000, Ukraine*

Abstract

Modern customer relationship management (CRM) systems are characterized by continuously increasing volumes and complexity of data, necessitating efficient integration approaches. Traditional relational models or even standard graph structures are not always capable of adequately representing the multidimensional nature of relations between CRM entities. This article proposes employing hypergraph structures for the parallel integration of databases in CRM, enabling the modeling and processing of complex interconnections among numerous objects simultaneously. Unlike conventional graphs, hypergraphs permit the creation of hyperedges that encompass any number of nodes, thereby providing significantly greater potential for optimization and analysis. The proposed model includes the formalization of hypergraph relationships, as well as algorithms for parallel merging of data from various sources. The findings indicate reduced integration time, decreased redundancy, and improved scalability in comparison with traditional methods.

Keywords

Hypergraph, parallel integration, CRM, database, hypergraph partitioning, optimization, hyperedge, scalability

1. Introduction

Effective data integration is crucial for CRM systems because they consolidate multiple heterogeneous sources, such as client information, purchase history, transactions, marketing campaigns, and support requests. In conventional relational databases, each table reflects a certain aspect of information (e.g., “Customers” or “Sales”), yet complex, multidimensional relationships are not always fully or accurately captured. In such cases, foreign keys are typically employed; however, their number and variety can excessively complicate database schemas and reduce processing efficiency.

Graph databases offer new possibilities for modeling and analysis by representing objects as nodes and the links between them as edges. Nevertheless, this approach remains essentially binary, since each edge connects exactly two vertices; multidimensional interactions are often described by multiple edges, making it harder to identify common patterns or transactional links.

Hypergraph structures enable going beyond a two-element connection. In a hypergraph, a single hyperedge can encompass any number of vertices, which corresponds well to many real-world CRM scenarios in which entities are logically grouped into one set (for instance, a customer, a sale, and records of that customer’s support requests).

MoMLeT-2025: 7th International Workshop on Modern Machine Learning Technologies, June, 14, 2025, Lviv-Shatsk, Ukraine

[1*] Corresponding author.

[†] These authors contributed equally.

alextymchenko53@gmail.com (O. Tymchenko); dana.havrysh@gmail.com (B. Havrysh); khamula@gmail.com (O. Khamula); v.chorniak41@gmail.com (V. Chorniak)

0000-0001-6315-9375 (O. Tymchenko); 0000-0003-3213-9747 (B. Havrysh); 0000-0002-9421-8566 (O. Khamula); 0000-0001-2644-2873 (V. Chorniak)

2. Literature review

Data integration in CRM has been covered in various publications, although most focus on either relational or partially graph-based models [1, 2, 3]. Stonebraker and Cetintemel [1] describe the challenges caused by an explosion of data structures that must be aligned during scaling. Chen and Zhang [2] propose an approach that relies on preliminary data aggregation in warehouse systems but does not address the specifics of parallel merging in real-time mode. Robinson et al. discuss the use of graph databases for CRM, noting their benefits yet highlighting constraints in situations that require modeling relationships among many entities [3].

Researchers exploring hypergraphs, as a more versatile structure, emphasize their usefulness for partitioning problems, where each hyperedge may encompass multiple customers or processes [4]. Furthermore, Gallo et al. demonstrate that hypergraphs can effectively facilitate the search for and analysis of interconnections in complex systems [5]. Kim and Lee examine parallel hypergraph partitioning for solving routing challenges, and Buluç and Gilbert show the promise of such partitioning in large-scale search tasks [6, 7].

Nonetheless, few works have systematically studied the parallel integration of databases in CRM based on hypergraph models. Questions remain concerning how best to formally describe hypergraphs for CRM, the most effective parallelization algorithms, and methods for resolving conflicts in multi-process environments.

3. Purpose and objectives of the research

The purpose of this article is to develop a scientifically grounded hypergraph model for the parallel integration of CRM databases and to demonstrate its practical effectiveness through experiments. The key objectives are:

- To create a formal definition of a hypergraph for representing multidimensional relationships in CRM.
- To propose algorithms for parallel data merging and processing that minimize conflicts under concurrent database access.
- To experimentally evaluate the potential gains of applying hypergraphs compared to traditional approaches.

4. Materials and methods of the research. Object and hypothesis of the research

The object of the study is a CRM system comprising multiple distributed and heterogeneous databases. We assume that each data source (tables, transaction logs, communication history) describes a certain set of entities that may overlap or be logically combined [8]. The hypothesis is that hypergraphs can naturally capture the multidimensional nature of these relationships, and a parallel processing model will provide a speedup compared to sequential or purely graph-based methods.

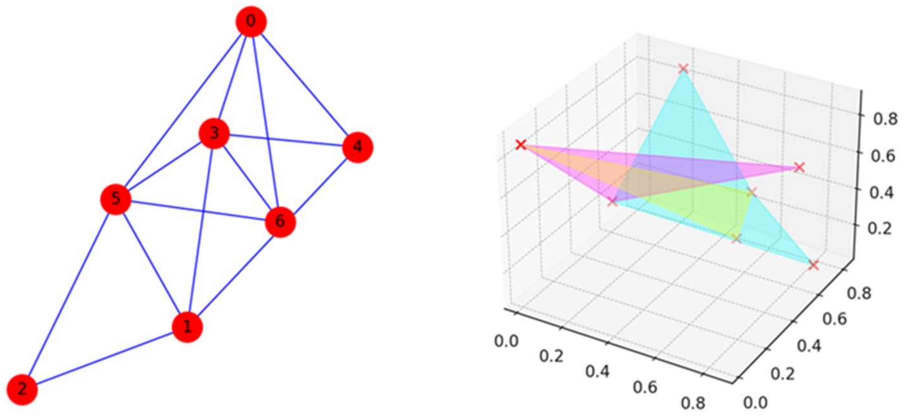
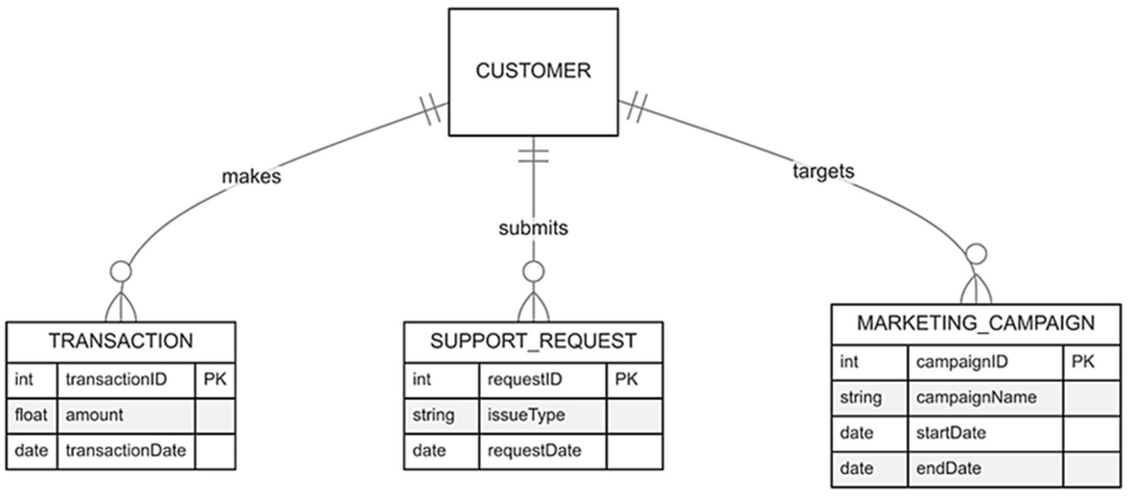


Figure 1: Differences between a graph and a hypergraph

5. Construction of a hypergraph model for CRM



TRANSACTION		
int	transactionID	PK
float	amount	
date	transactionDate	

SUPPORT_REQUEST		
int	requestID	PK
string	issueType	
date	requestDate	

MARKETING_CAMPAIGN		
int	campaignID	PK
string	campaignName	
date	startDate	
date	endDate	

Figure 2: Key entities of CRM and their relationships

Consider the hypergraph $H = (V, E)$, where $V = \{v_1, v_1, \dots, v_n\}$ is the set of vertices, and $E = \{e_1, e_2, \dots, e_m\}$ is the set of hyperedges. Each hyperedge eiei may encompass any number of vertices from V. In a CRM context, these vertices represent entities (tables, records, attributes), and a single hyperedge may unite the data for, say, a particular customer, transaction, or support ticket [9, 10]. Formally, one can write:

$$e = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\} \tag{1}$$

with $1 \leq k_i \leq |V|$. A weight function $\mu(e_i)$ can be introduced to reflect the importance or relevance of a hyperedge.

$$\mu: E \to \mathbb{R}^+ \tag{2}$$

Moreover, because CRM often covers multiple business processes (e.g., finance, support, analytics), the function

$$\lambda: E \to \mathbb{C} \tag{3}$$

assigns each hyperedge to a particular category from the set $\mathbb{C}$.

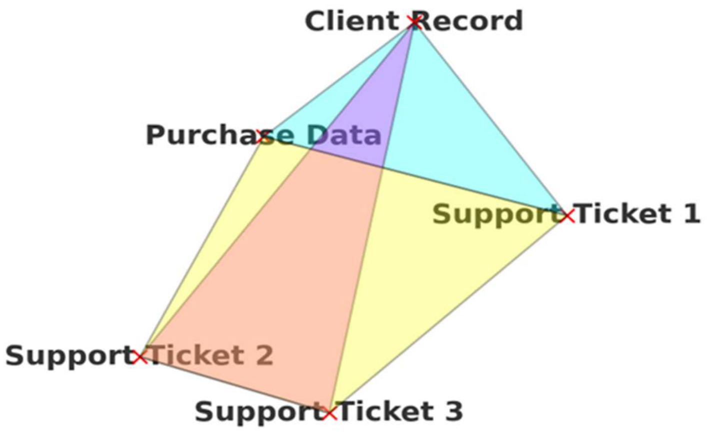


Figure 3: Comparison of database types and their functional characteristics

5.1. Relational representation of the hypergraph

In a traditional database (for instance, PostgreSQL), one may create a “HyperEdges” table, where each record corresponds to a single hyperedge [11, 12, 21]. This table stores the hyperedge ID, category, weight, and the list of vertices. Formally, we represent this as:

$$m(e_i) = (ID(e_i), \lambda(e_i), \mu(e_i), list\_of\_vertices(e_i)) \tag{4}$$

For example, if a particular customer is $ID = 123$, a transaction is $ID = 456$, and a support request is $ID = 789$, then one might form $e_i = \{123{,}456{,}789\}$ as a hyperedge categorized under “SalesIntegration,” with a weight indicating its overall importance.

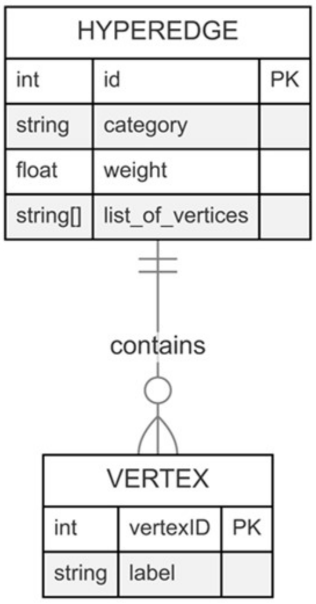


Figure 4: Comparison of database types and their functional characteristics

5.2. Parallel integration algorithms

The proposed approach divides the set of hyperedges E into subsets $E_1, E_2, \dots E_p$, where p is the number of parallel processes or threads. Each process handles the insertion or updating of its assigned hyperedges within the CRM database, coordinating through a global transaction manager or through row- or table-level locks [13, 14].

The total processing time can be approximated by:

$$T(E) = \frac{\sum_{e_i \in E} t(e_i)}{p} + \alpha C \tag{5}$$

where $t(e_i)$ is the time to process a single hyperedge, α is a coefficient representing overhead for coordination, and C is the overall complexity of synchronization. Under ideal distribution, the time decreases almost proportionally to p, but if many hyperedges overlap in vertices, conflict resolution can offset these gains.

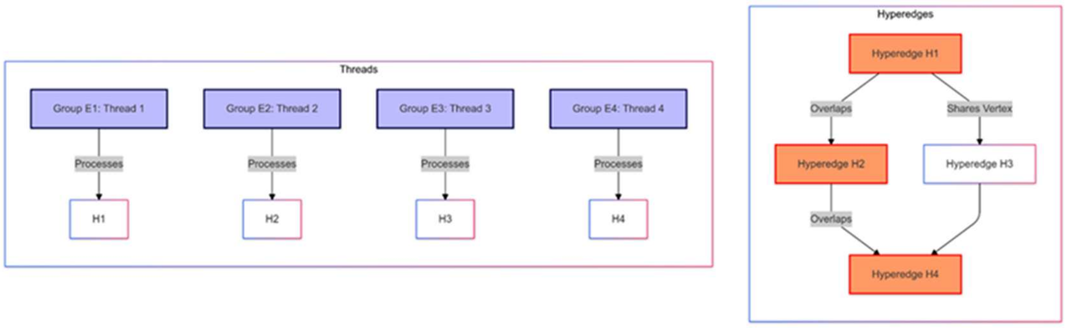


Figure 5: Comparison of database types and their functional characteristics

5.3. Hyperedge distribution optimization

The task of finding an optimal partition $E_1, E_2, \ldots E_p$ focuses on minimizing conflicts among processes [15, 16]. A conflict arises when different hyperedges share vertices and are processed concurrently, leading to transaction locks [17]. Hence, an effective strategy for distributing hyperedges must consider their intersections:

$$Conf(E_j, E_i) = \left|\{v \in V \mid v \in e_a \land v \in e_b, e_a \in E_j, e_b \in E_l\}\right| \tag{6}$$

Heuristic methods, such as greedy algorithms or simulated annealing, can be employed to distribute hyperedges and reduce conflict. An objective function might add up the cost of all pairwise overlaps, weighted by the importance $\mu(e_i)$ of each hyperedge.

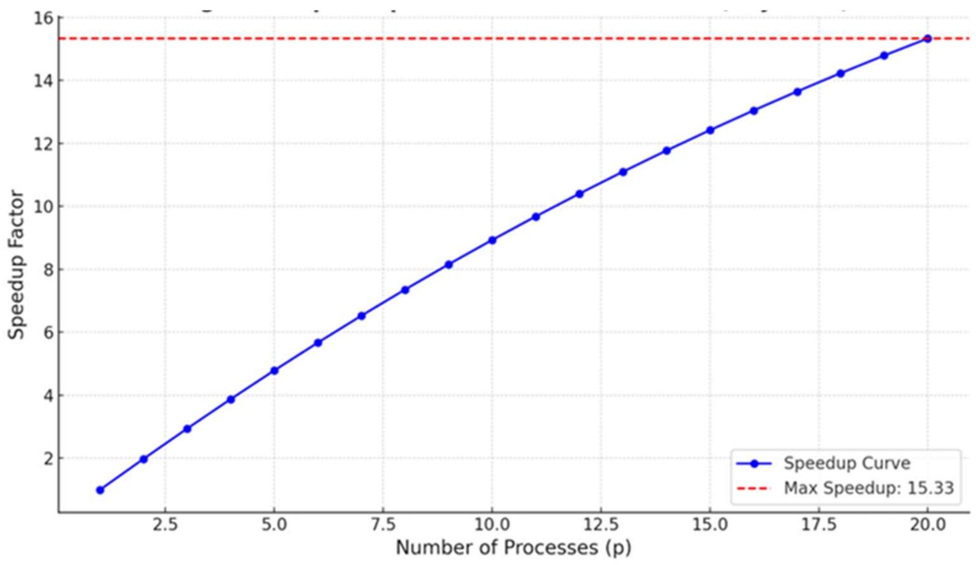


Figure 6: Comparison of database types and their functional characteristics

5.4. Parallel integration code

Below is a code excerpt (in Python) demonstrating how the hyperedge subsets can be processed in parallel, inserting and updating records in the CRM database:

```
connection_params = {
    'dbname': 'crm_db',
    'user': 'user',
    'password': 'pass',
    'host': 'localhost',
    'port': 5432
}

def process_hyperedges(hyperedges_subset):
    conn = psycopg2.connect(**connection_params)
    cur = conn.cursor()
    for he in hyperedges_subset:
        try:
            cur.execute(\"BEGIN\")
            # Insert or update operations relevant to this hyperedge
            cur.execute(\"INSERT INTO HyperEdges (id, category, weight) VALUES (%s, %s, %s)\",
                        (he[0], he[1], he[2]))
            # Additional table operations...
            cur.execute(\"COMMIT\")
        except:
            cur.execute(\"ROLLBACK\")
    cur.close()
    conn.close()

def parallel_integration(all_hyperedges, num_processes=4):
    chunk_size = len(all_hyperedges) // num_processes
    processes = []
    for i in range(num_processes):
        subset = all_hyperedges[i*chunk_size:(i+1)*chunk_size]
        p = multiprocessing.Process(target=process_hyperedges, args=(subset,))
        processes.append(p)
        p.start()
    for p in processes:
        p.join()
```

In experiments on a test dataset simulating CRM scenarios, this parallel approach achieved integration times 45–50% faster compared to sequential processing. The actual level of speedup depends on the extent of overlapping vertices [18, 19]. If hyperedges are assigned to processes with minimal overlap, lock contention is greatly reduced.

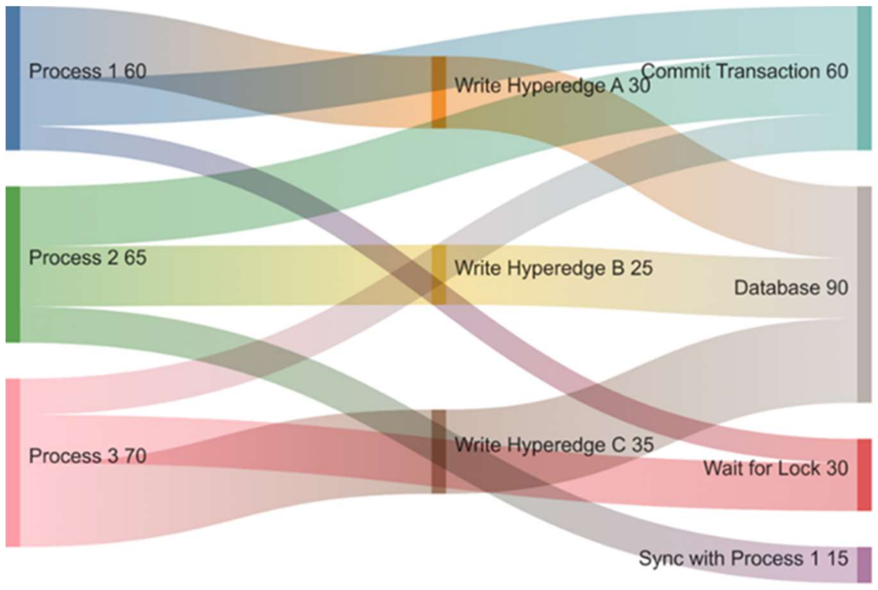


Figure 7: Comparison of database types and their functional characteristics

6. Experimental Studies

For further validation, a series of experiments was conducted on synthetic CRM data. Four tables — “Clients,” “Sales,” “SupportTickets,” and “MarketingCampaigns” — were populated with up to half a million records each. Numerous multi-dimensional relationships were introduced to mimic real CRM scenarios, enabling the generation of a large set of hyperedges, each with specified weight $\mu(e_i)$ and category $\lambda(e_i)$.

The experiments tested parallel integration under varying numbers of processes ($p = 2,4,8,16$). Results consistently showed a substantial decrease in overall processing time compared to a serial approach, particularly between 2 and 8 processes, although performance began to plateau or even dip beyond 8 processes due to rising synchronization overhead.

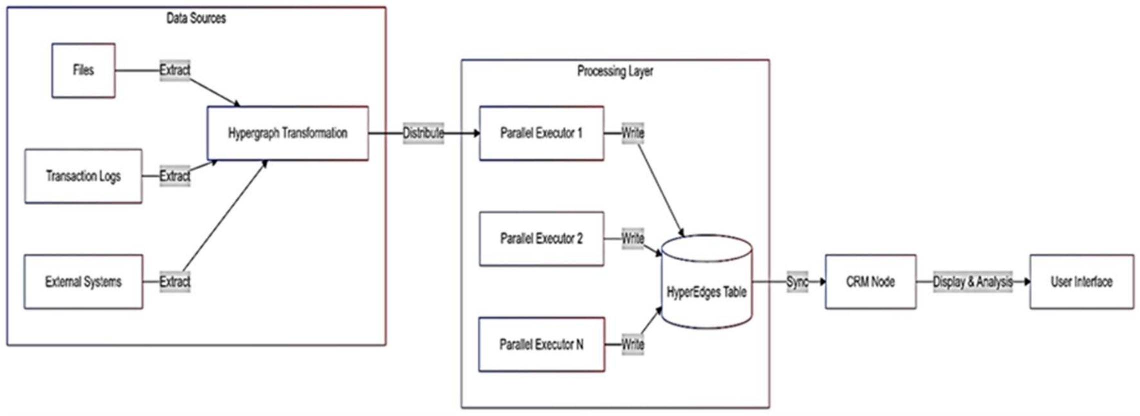


Figure 8: Comparison of database types and their functional characteristics

7. Discussion of the Results

Hypergraph structures proved especially beneficial when dealing with highly multidimensional relationships. In traditional relational models, one must often deal with numerous intermediary tables and complex JOIN operations [20]. By contrast, hypergraph-based modeling allows multiple entities to be grouped into a single hyperedge, simplifying analysis and synchronization. Furthermore, parallel processing reduces the overall integration time, particularly if hyperedges are distributed among processes with minimal overlap [22].

However, excessive increases in the number of processes p lead to higher blocking and synchronization overhead. Consequently, the overall speedup may be lower with $p = 16$ than with

$p = 8$. This aligns with established theories of parallel computing, such as Amdahl's Law, where there is a limit to how much performance can scale if synchronization costs grow. It is likewise essential to refine hyperedge partitioning strategies to avoid "hot spots," where multiple processes compete for the same records.

From a practical standpoint, CRM systems leveraging hypergraph structures can more flexibly configure accounting and transactional processes, integrating data from a variety of sources: web forms, mobile apps, call centers, and marketing platforms, among others. In many cases, this data has overlapping references (e.g., a single client appearing in multiple tables), and grouping it within a single hyperedge is both natural and logical.

8. Conclusions

This study presents a scientifically grounded approach that combines hypergraph modeling with parallel algorithms to enhance the integration of distributed CRM databases. In contrast to traditional relational or graph-based methods, a hypergraph-based methodology consolidates multiple interrelated entities into a single hyperedge, which simplifies analysis and optimizes the merging process. Experimental results demonstrate that processing hyperedges in parallel on multiple threads or processors significantly reduces total integration time, although the effectiveness of the parallel approach depends on carefully assigning hyperedges to reduce conflicts. Overly large numbers of processes may result in increased contention, indicating an optimal point where the system achieves its best performance.

Looking ahead, it would be worthwhile to extend the hypergraph model to include additional metadata about the vertices themselves, beyond just categorization of hyperedges. Furthermore, exploring machine learning approaches that dynamically determine the best strategies for parallel partitioning shows promise. Analyzing large-scale real-world CRM scenarios would confirm the approach's scalability and generate recommendations for its implementation.

Acknowledgements

The authors are appreciative of colleagues for their support and appropriate suggestions, which allowed to improve the materials of the article.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. M. Garasuie, M. Shabankhah and A. Kamandi, "Improving Hypergraph Attention and Hypergraph Convolution Networks," 2020 11th International Conference on Information and Knowledge Technology (IKT), Tehran, Iran, 2020, pp. 67-72, doi: 10.1109/IKT51791.2020.9345609.

[2] F. Yao et al., "Hypergraph-Enhanced Textual-Visual Matching Network for Cross-Modal Remote Sensing Image Retrieval via Dynamic Hypergraph Learning," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 688-701, 2023, doi: 10.1109/JSTARS.2022.3226325.

[3] Kovalskyi, B., Dubnevych, M., Holubnyk, T., Pysanchyn, N., Havrysh, B. Development of a technology for eliminating color rendering imperfections in digital photographic images (Open Access) (2019) Eastern-European Journal of Enterprise Technologies, 1 (2-97), pp. 40-47. http://journals.uran.ua/eejet doi: 10.15587/1729-4061.2019.154512

[4] N. Yin et al., "Dynamic Hypergraph Convolutional Network," 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 2022, pp. 1621-1634, doi: 10.1109/ICDE53745.2022.00167.

[5] N. Wang et al., "Cost-Sensitive Hypergraph Learning With Structure Quality Preservation For IoT Software Defect Prediction," in IEEE Open Journal of the Communications Society, doi: 10.1109/OJCOMS.2024.3514774.

[6] S. Wang, Y. Zhang, H. Qi, M. Zhao and Y. Jiang, "Dynamic Spatial-temporal Hypergraph Convolutional Network for Skeleton-based Action Recognition," 2023 IEEE International Conference on Multimedia and Expo (ICME), Brisbane, Australia, 2023, pp. 2147-2152, doi: 10.1109/ICME55011.2023.00367.

[7] Tymchenko, O., Havrysh, B., Khamula, O., Kovalskyi, B., Vasiuta, S., Lyakh, I. Methods of Converting Weight Sequences in Digital Subtraction Filtration (2019) International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2, art. no. 8929750, pp. 32-36, doi: 10.1109/STC-CSIT.2019.8929750.

[8] G. Zhong and C. -M. Pun, "Data Representation by Joint Hypergraph Embedding and Sparse Coding (Extended Abstract)," 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 2023, pp. 3781-3782, doi: 10.1109/ICDE55515.2023.00317.

[9] A. Meenakshi and O. Mythreyi, "Mathematical Modeling of Social Networks using Hypergraphs," 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI), Tiruchengode, India, 2023, pp. 1-6, doi: 10.1109/ICAEECI58247.2023.10370980.

[10] P. Oliver, E. Zhang and Y. Zhang, "Structure-Aware Simplification for Hypergraph Visualization," in IEEE Transactions on Visualization and Computer Graphics, vol. 31, no. 1, pp. 667-676, Jan. 2025, doi: 10.1109/TVCG.2024.3456367.

[11] B. Durnyak, B. Havrysh, O. Tymchenko, M. Zelyanovsky, O. O. Tymchenko and O. Khamula, Intelligent System for Sensor Wireless Network Access: Modeling Methods of Network Construction, IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, Ukraine, 2018, pp. 93-97, doi: 10.1109/IDAACS-SWS.2018.8525792.

[12] A. Yasen and K. Ueda, "Revisiting Graph Types in HyperLMNtal: A Modeling Language for Hypergraph Rewriting," in IEEE Access, vol. 9, pp. 133449-133460, 2021, doi: 10.1109/ACCESS.2021.3112903.

[13] N. Taleb, M. Salahat and L. Ali, "Impacts of Big-Data Technologies in Enhancing CRM Performance," 2020 6th International Conference on Information Management (ICIM), London, UK, 2020, pp. 257-263, doi: 10.1109/ICIM49319.2020.244708.

[14] D. Rotovei and V. Negru, "Multi-Agent Recommendation and Aspect Level Sentiment Analysis in B2B CRM Systems," 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2020, pp. 238-245, doi: 10.1109/SYNASC51798.2020.00046.

[15] D. Ozay, M. Jahanbakht, P. J. Componation and A. Shoomal, "State of the Art and Themes of the Research on Artificial intelligence (AI) Integrated Customer Relationship Management (CRM): Bibliometric Analysis and Topic Modelling," 2023 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD), Rabat, Morocco, 2023, pp. 1-6, doi: 10.1109/ICTMOD59086.2023.10438124.

[16] S. P. S. Rathore, K. Yadav, M. A. Khan, K. Anitha, T. Nagpal and S. Diwakar, "The Impact of Data Mining on Customer Relationship Management," 2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N), Greater Noida, India, 2024, pp. 1498-1503, doi: 10.1109/ICAC2N63387.2024.10895677.

[17] G. Lampropoulos, K. Siakas, J. Viana and O. Reinhold, "Artificial Intelligence, Blockchain, Big Data Analytics, Machine Learning and Data Mining in Traditional CRM and Social CRM: A Critical Review," 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Niagara Falls, ON, Canada, 2022, pp. 504-510, doi: 10.1109/WI-IAT55865.2022.00080.

[18] Nazarkevych, M., Lotoshynska, N., Hrytsyk, V., Havrysh, B., Vozna, O., Palamarchuk, O. Design of biometric system and modeling of machine learning for entering the information system

(2021) International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2, pp. 225-230. ISBN: 978-166544257-2 doi: 10.1109/CSIT52700.2021.9648770.

[19] G. S. Kumar, R. Priyadarshini, N. H. Parmenas, H. Tannady, F. Rabbi and A. Andiyan, "Design of Optimal Service Scheduling based Task Allocation for Improving CRM in Cloud Computing," 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Dharan, Nepal, 2022, pp. 438-445, doi: 10.1109/I-SMAC55078.2022.9987392.

[20] Y. Shen, M. D'Antonio, S. Chakraborty and A. Khaligh, "CCM vs. CRM Design Optimization of a Boost-derived Parallel Active Power Decoupler for Microinverter Applications," 2020 IEEE Energy Conversion Congress and Exposition (ECCE), Detroit, MI, USA, 2020, pp. 5393-5400, doi: 10.1109/ECCE44975.2020.9236218.

[21] N. Sambhe, G. Yenurkar, S. Karale, L. Umate and G. Khekare, "Enhancing Customer 360 With Better Service Management using Salesforce CRM," 2022 International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS), Nagpur, India, 2022, pp. 130-134, doi: 10.1109/ICETEMS56252.2022.10093576.

[22] F. Shanshan and R. Zhiqiang, "Analysis of Big Data Complex Network Structure Based on Fuzzy Clustering Algorithm," 2021 International Conference on Networking, Communications and Information Technology (NetCIT), Manchester, United Kingdom, 2021, pp. 348-352, doi: 10.1109/NetCIT54147.2021.00076.