An improved coverage pattern based ad-slot allocation framework for display advertising

Thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science
in
Computer Science and Engineering by Research

by

Sathineni Preetham Reddy
20161063
preethamreddy.s@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
April, 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled ”An improved coverage pattern based ad-slot allocation framework for display advertising” by Sathineni Preetham Reddy, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. P. Krishna Reddy,
Data Sciences and Analytics Center,
Kohli Center on Intelligent Systems,
IIIT Hyderabad.

To my parents

Acknowledgments

I would like to express my eternal gratitude my guide – Prof. P. Krishna Reddy for introducing me to research and for his invaluable supervision, motivation and support during the course of my research. His constant guidance has helped me a lot in overcoming a lot of barriers in my journey. I would also like to extend my heartfelt thanks to the late Dr. Anirban Mondal, whose rigorous review process of research papers before submission to conferences greatly enhanced my understanding of how to articulate ideas effectively. I would like to acknowledge and convey my gratitude for the tremendous assistance of Dr. Srinivas Annappalli, who has guided me as a mentor and co-authored in the publications I have authored for this thesis.

I would like to thank my friend Subramanyam varma for helping me in brainstorming and formulating ideas for this journal. I would like to thank my friend Harshini Mandadapu for her continuous motivation and moral support throughout. I would also like to thank my friends Vashist Madiraju, Anirudh Reddy, Y.V.S Harish, Harsha Vardhen, Sathvik Sanjeev and Divija Palleti for making my college life a wonderful memory that I will cherish forever. I would also like to thank my colleagues Saideep Chennupati, Kaushik Komalapalli and Pradeep pallakila for their support.

I am very grateful to my family for putting their faith in me and constantly supporting me in everything throughout the journey.

Abstract

Advertising is a part of marketing activity and brings awareness to potential viewers about the products/services. It helps businesses to increase sales and create a brand image of the products. Online advertising is a form of advertising which uses the Internet to promote products/services to viewers. With the proliferation of eCommerce, online advertising has become a popular mode of advertising. It enables a higher reach of users at reduced costs. Search engine advertising, display advertising, social media advertising, native advertising, video advertising, and email advertising are some of the modes of online advertising. In this thesis, we have made an effort to improve display advertising.

Web banners or display advertising are important among the types of online advertising. Display advertising mainly consists of visitors, advertisers, and the publisher. The publisher owns the website (a collection of web pages). A web page contains ad slots. Advertisers are interested in placing the banners in the ad slots to expose banners to potential users. Normally, given a website, several advertisers bid for slots. Given the website, click stream data of user visits, and the demands of multiple advertisers for user visits, the issue is to develop an approach to allocating ad slots so that the maximum number of advertisers' demands can be satisfied and the publisher's revenue can be improved. The naive approach suffers from the issues of ad repeatability and underutilization of slots. Existing works addressed this problem by modeling it as an optimization problem by employing reinforcement learning and data mining techniques. Also, in the literature, there is an effort to solve the problem by exploiting the knowledge of coverage patterns and assuming a single slot per web page. However, there are multiple slots per page in practice, and the existing approach can not be extended to a practical scenario.

In this thesis, we have tried to improve the existing coverage pattern-based approach, which has been proposed by assuming a single slot per page by considering a practical scenario, i.e., by considering multiple ad slots for each web page. As part of this effort, it was noted that the existing coverage pattern algorithms are not scalable to very large-size click-stream transactions. So, in this thesis, to improve the scalability of the existing approach, we have developed a distributed approach to mine coverage patterns; next, we have proposed an ad slot allocation approach by considering multiple ad slots per page.

In the proposed distributed coverage pattern mining (DCPM) approach, we employ a notion of the summarized form of Inverse Transactional Database (ITD) and replicate it at every node. We also employ an efficient clustering-based method to distribute the computational load among the Worker nodes. We have performed extensive experiments using two real-world datasets and one synthetic dataset. The

results show that the proposed approach significantly improves the performance over the state-of-the-art approaches in terms of execution time.

We have also proposed a framework for display advertising by considering multiple slots per page. In this framework, we employ the DCPM approach and compute the ad-slot patterns *ASPs*. We then calculate the impressions of each ASP and propose an efficient allocation approach to meet the advertiser demand in the form of impressions. Our extensive performance evaluation using two real-time click-stream datasets demonstrates the efficiency of the proposed framework in terms of improved publisher revenue and reduced ad repeatability.

Ad revenue is vital for several e-commerce companies and revenue from display advertising is one of the opportunities. We have proposed a scalable pattern mining approach to improve the publisher's revenue through display advertising. We hope that the proposed framework will facilitate the improvement of the approaches that are being followed by display advertisement companies.

Contents

List of Figures

List of Tables

Important notations and abbreviations

Notation	Description
TDB	Transactional Data Base
CPs	Coverage Patterns
OR	Overlap Ratio
CS	Coverage Support
$NOPs$	Non-overlap patterns
minRF	Minimum relative frequency
minCS	Minimum coverage support
maxOR	Maximum overlap ratio
ASPs	Ad Slot Patterns

Chapter 1

Introduction

Advertising plays a pivotal role in today's business landscape, serving as an essential tool for organizations to promote their products, services, and brand identities. It helps businesses connect with target audiences, influence consumer behavior, and drive sales and business growth. The advertising industry has witnessed a transformative journey, shifting from conventional physical media, such as billboards and print publications, to embrace new technological horizons. As a medium, the internet has redefined advertising over the past couple of decades through online advertising. In this thesis, we propose an improved approach to improve display advertising, which is a type of online advertising.

In the rest of this chapter, we first briefly explain the different types of online advertising along with research issues. Next, we explain display advertising and briefly describe the related work. Subsequently, we summarize the contributions and present the thesis organization.

1.1 Types of online advertising

The advertisers run campaigns on any one or simultaneously across all these types of online advertising. We discuss some of the important modes of online advertising: Search Engine Advertising (SEA), Display Advertising, Social Media Advertising, Native Advertising, Video Advertising, and Email Advertising.

Search Engine Advertising (SEA): It represents a specialized subset of Search Engine Marketing (SEM), wherein advertisers bid to place their ads on search engine results pages (SERPs) in response to specific user queries [19] as shown in Figure 1.1. Predominantly utilizing a Pay-Per-Click (PPC) model, SEA capitalizes on immediate visibility, ensuring that brands appear prominently during potential customers' active searches. The essence of SEA lies in its targeted nature, ensuring high relevance by matching ads to user-intent-driven queries.

The dynamic sphere of SEA invites numerous research issues. A prominent challenge revolves around understanding and optimizing bid strategies in an auction environment, ensuring cost-effectiveness while maximizing visibility [49]. Furthermore, deciphering the nuanced role of ad copy, landing page relevance, and user behavior metrics in influencing the 'quality score'—a pivotal determinant in ad

placements—remains a focal area of investigation. With the ubiquity of multi-device usage, understanding cross-device conversions and attribution modeling further complicates SEA's research spectrum.

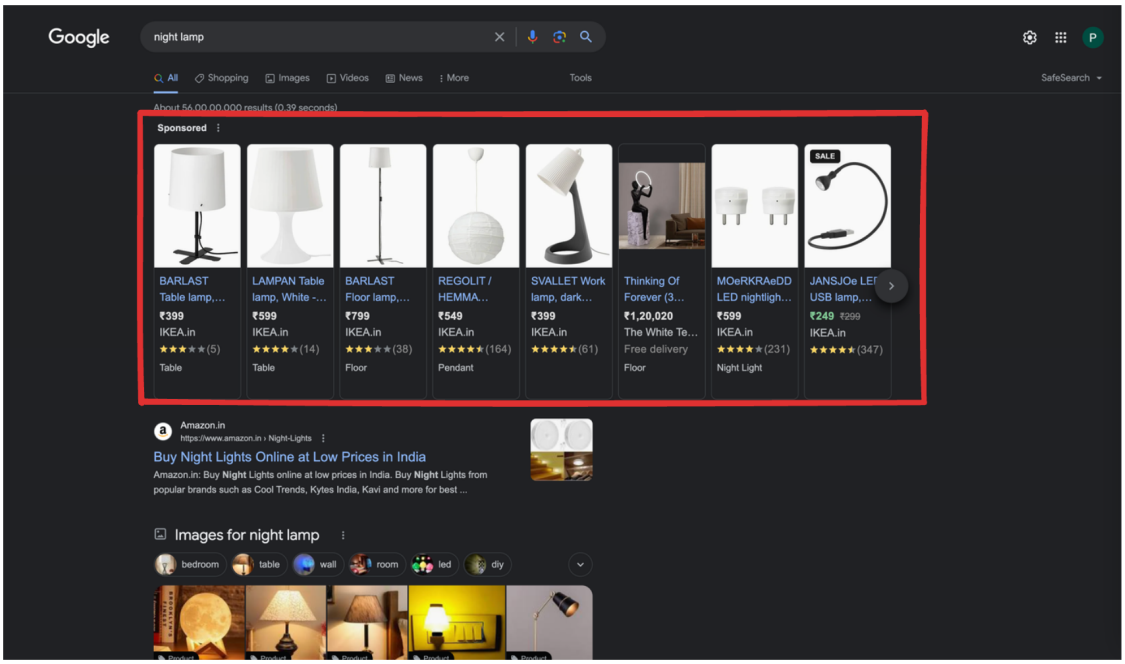


Figure 1.1: Search Engine Advertising

Display Advertising: It employs visual graphics to promote brands on websites, applications, and various digital platforms, encompassing formats such as banners, interstitials, and rich media [26] as shown in Figure 1.2. Unlike search ads that cater to explicit user queries, display ads aim to capture passive attention, creating brand awareness, and fostering recall. With the integration of multimedia elements and interactivity, display ads strive to provide an informative and engaging user experience across many digital landscapes.

The intricate matrix of display advertising poses numerous research conundrums. One primary area of exploration is the efficacy of targeted versus non-targeted ads and how the incorporation of user data influences ad performance [33]. Ad fatigue, resulting from repetitive exposure, challenges researchers to discern optimal frequency caps. Additionally, with the increasing adoption of ad blockers, understanding the underlying causes of user aversion to display ads and devising strategies to circumvent this issue are subjects of pressing academic interest.

Social Media Advertising: It has firmly established its place in the contemporary digital marketing paradigm, leveraging vast platforms to disseminate tailored messages to targeted audiences [29] as shown in Figure 1.3. Through platforms like Facebook, Instagram, Twitter, and LinkedIn, brands can engage consumers in interactive and personalized ways, building communities around products or ideologies. Given social media's two-way communication channel, advertisers can cultivate brand loyalty, foster real-time engagement, and drive awareness and conversion with unparalleled granularity.

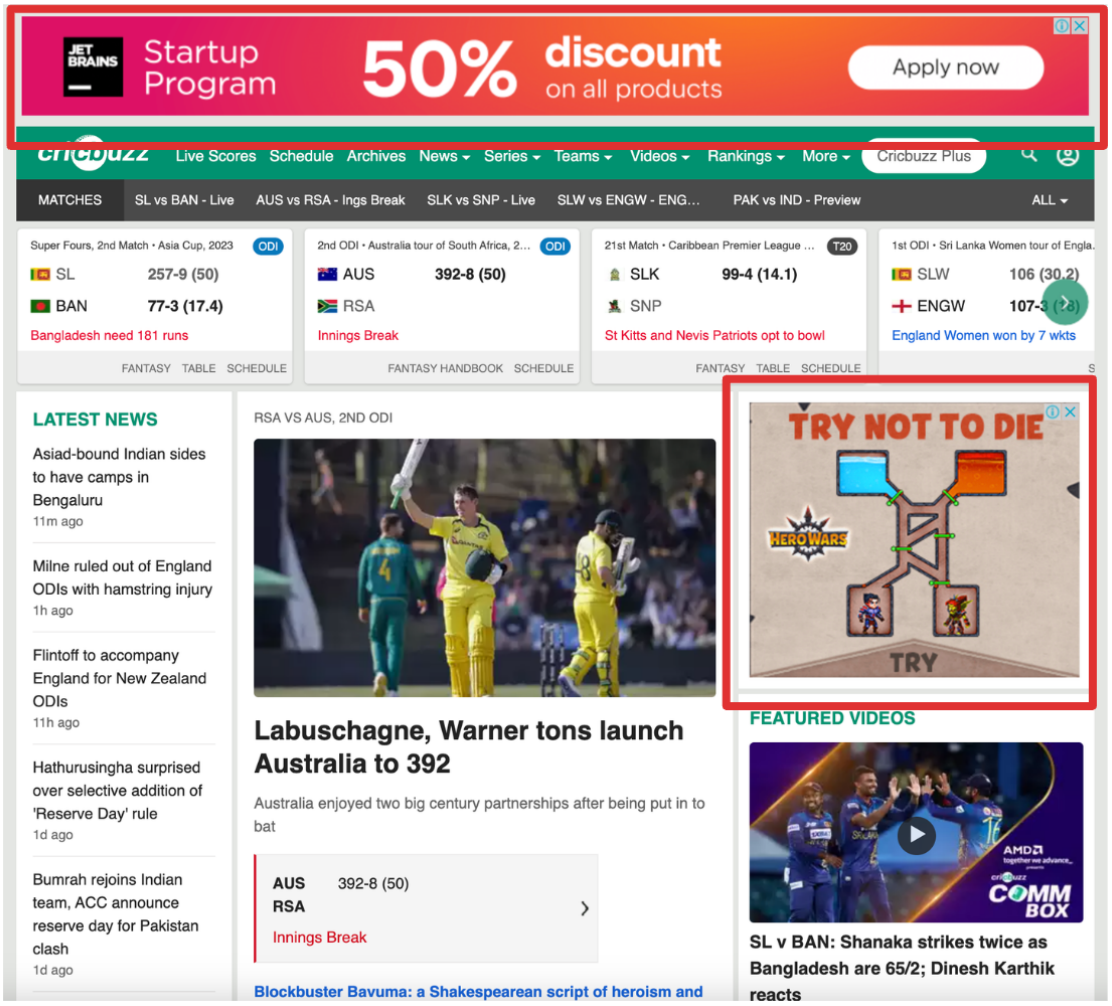


Figure 1.2: Display Engine Advertising

Despite its ubiquity and success, social media advertising presents a gamut of research challenges. Central to academic discourse is understanding the determinants of user engagement and the efficacy of paid versus organic reach [58]. With algorithmic changes in platforms influencing content visibility, the intricacies of platform-specific advertising strategies and their alignment with user behavior and preferences remain under the microscope. Furthermore, the continuous evolution of new formats, such as ephemeral content and influencer collaborations, mandates research into their long-term viability and impact on consumer perceptions.

Native Advertising An innovative approach within the digital marketing sphere, seamlessly integrates promotional content into media platforms in a manner that aligns with the user experience, often masquerading as editorial content [63] as shown in Figure 1.4. This subtle blend ensures that native ads are less intrusive than traditional advertising formats, capitalizing on the user's trust and engagement

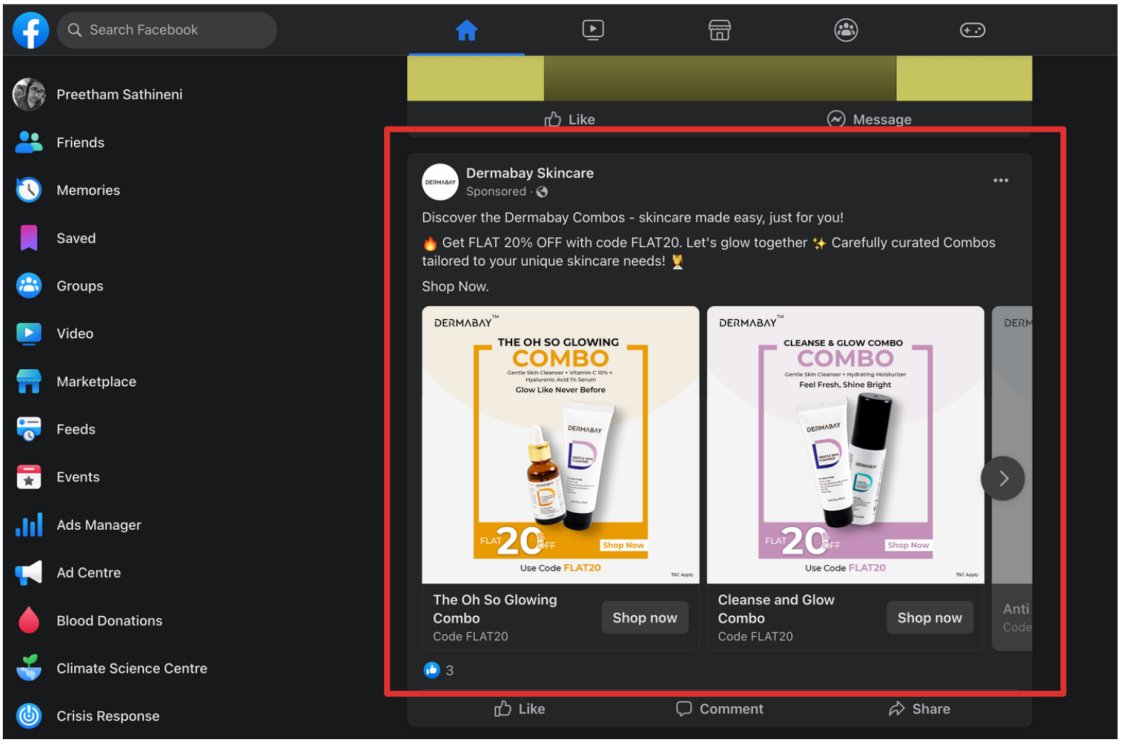


Figure 1.3: Social Media Advertising

with the platform. By matching the platform's form, feel, and function, native advertising offers brands an avenue to communicate with consumers without disrupting their online journey.

Delving into native advertising unfolds a suite of research challenges. One pivotal area of investigation concerns consumer recognition and discernment, exploring the potential for audience misinterpretation and the ethical implications therein [11]. With the rise of programmatic buying, understanding the algorithms that place native ads and ensuring content relevancy becomes essential. Moreover, measuring the efficacy of native ads, given their non-traditional format, and quantifying metrics such as engagement and trust in comparison to standard advertising methods, remain subjects of rigorous academic exploration.

Video Advertising It has emerged as a dominant and compelling medium within the digital marketing ecosystem, harnessing dynamic visual content to engage audiences on platforms ranging from social media to streaming services [1] as shown in Figure 1.5. Given its capacity to tell compelling stories and create immersive experiences, video ads have proven effective in fostering brand awareness, generating leads, and driving sales. This format leverages the synergy of audiovisual elements, positioning itself as a powerful tool in a marketer's arsenal, especially in an age where user attention spans are increasingly fragmented.

The expanding terrain of video advertising presents several research challenges. Key areas of academic investigation encompass optimal video length in relation to user engagement, the efficacy of

Figure 1.4: Native Advertising

in-stream versus out-stream ads, and the emergent role of interactive video ads in enhancing the user experience [59]. With the proliferation of mobile devices, ensuring video content is optimized for varying screen sizes and resolutions becomes paramount. Moreover, understanding the intricate balance between storytelling and direct promotion, and its impact on conversion rates, is an area of continuous exploration.

Email Advertising Email advertising, often termed email marketing, remains an enduring and potent instrument within the digital marketing domain, leveraging personalized content to communicate directly with consumers in their inboxes [32] as shown in Figure 1.6. This channel allows brands to curate targeted messages, fostering sustained engagement, and cultivating brand loyalty. Given its inherent opt-in nature, email advertising holds the advantage of addressing an audience that has already expressed interest in the brand's offerings, rendering it a valuable tool for nurturing leads and driving conversions.

The realm of email advertising is not devoid of its research intricacies. Scholars are actively exploring the optimal frequency and timing of email dispatch to maximize open rates and avoid recipient fatigue [40]. The increasing prevalence of mobile email access necessitates investigations into mobile-

Figure 1.5: Video Advertising

optimized designs and the impact of interactive elements within email content. Furthermore, as consumers become more privacy-conscious, research on the balance between personalization and privacy, and the evolving regulations surrounding data usage in email marketing, gains prominence.

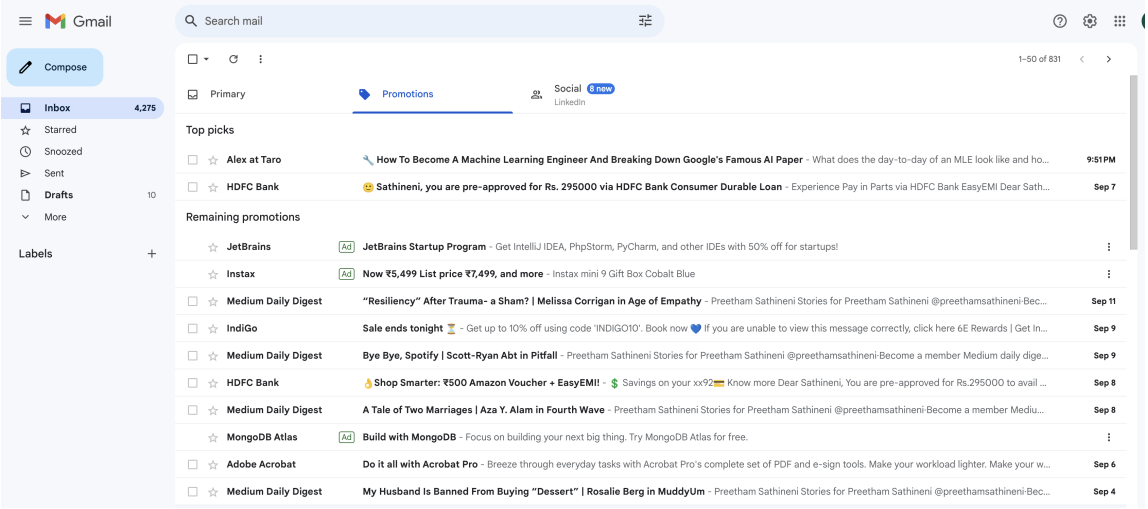


Figure 1.6: Email Advertising

Among these, Display advertising is one of the active and predominantly used mode of advertising. We now present the details of display advertising.

1.2 About Display advertising

Display advertising is a mode of online advertising in which the advertisements are displayed on the web page. A web page contains a set of ad slots, where the advertisements can be placed. Basically, display advertisement consists of the following stakeholders: (i) visitors, (ii) advertisers, and (iii) publishers. We now explain the role of each stakeholder along with their interests.

(i) Visitor: A visitor is a user who visits web pages. We assume that all the ad slots present on the web page are viewed by the visitor when the visitor visits the web page. Therefore, the number of views associated with an ad slot equals the number of visitors visiting the web page. The main aim of the visitor would be to have a pleasant browsing experience in a given session and would get annoyed if there is a repeated display of the same advertisements.

(ii) Advertiser: An advertiser is a person or company that advertises a product, service, or event as an advertisement. Visitors visit the advertisement (ad slots) on the web page and the number of distinct visits of the advertisement is referred to as *impressions*. The advertiser places one's demand in the form of impressions (number of distinct views).

(iii) Publisher: A publisher is a user who manages ad slots on web pages. The publisher allocates ad slots to advertisements based on the demand for impressions. The publisher makes revenue by allocating the ad slots to the advertiser based on the demand as per guaranteed contract norms. The guaranteed contract norms for display advertising guarantee that the publisher meets the advertisers' demand and thus guarantees the corresponding views demanded. In this paper, we are considering the *Cost Per Impression* (*CPI*) model, which is the most commonly used revenue model in display advertising.

One of the research issues in display advertising is efficient ad slot allocation. The problem of ad slot allocation is as follows: Given a click-stream data set and a set of advertisers along with their demand, the task of the publisher is to allocate a set of web pages to each advertiser such that the advertiser's demand is met and the publisher's revenue is maximized. A rudimentary approach to solve this problem includes (i) identification of all unique ad slots on the web pages and (ii) random allocation of ad slots to each advertiser until one's demand is met. However, such a rudimentary approach is less efficient for the following reasons. Firstly, it degrades the visitor experience because there will be many repeating advertisements on the web pages viewed by the visitor. For example, if a visitor is visiting three pages, he/she may see the same ad on all pages. (Note that an ideal allocation approach should display a different ad to the user on each page.) Secondly, it reduces the publisher's revenue because one cannot meet the required demand of advertisers due to the wastage of views caused by the repetition of user visits. From this discussion, it is evident that the main aim of the publisher is to find the set of ad slots such that the repeated user views of the advertisement are minimized, and distinct user views of the advertisements/ad slots are maximized.

In the literature, the problem of display ad-slot allocation has been tackled through various approaches. The ad-slot allocation problem was modeled as an optimization problem and proposed optimal or near-optimal solutions for the same[20, 21, 61]. Moreover, reinforcement learning-based techniques were proposed in [2, 71, 35], which train the model through user-activity-based reinforcements. In

addition, the authors in [31] proposed a data-mining-based framework to address the display ad-slot allocation. However, the work proposed in [31] is limited to a single ad-slot per web page. In this thesis, we propose an improved allocation approach by considering multiple ad-slots per web page.

1.3 Research gap and motivation

We explain the research gap in an elaborative manner.

- An effort has been made in [31] to address the ad-slot allocation problem in display advertising scenarios. This approach proposes a coverage pattern-based approach to improve the ad-slot allocation. However, the approach in [31] has been proposed by considering only one ad slot per web page. The limitation is that, as we have multiple ad slots per page in practical scenarios, the approach cannot be applied in practical scenarios.
- Another issue is that the existing coverage pattern mining is not scalable for transactional datasets in which the number of items is large. In the clickstream datasets of a typical website, each record contains the IDs of webpages a user visits. Overall, the number of items is relatively large, i.e., equal to the number of webpages on the website. The algorithm used in [31] is not scalable. In another context, an effort is to investigate a scalable approach based on the MapReduce framework in [47]. However, as the size of the transactional database (TDB) and the total number of items in the display ad scenario are huge, the approach in [47] has not worked. Under MapReduce, it has been observed that there is high data shuffling of a large number of candidate patterns at each level among various machines in distributed systems.

Overall, we have proposed an approach to improve the scalability of the coverage pattern mining approach and proposed a workable and practical ad allocation framework by considering multiple ad slots per page.

1.4 Overview of the proposed approaches

As already mentioned, in this thesis, we propose a scalable and efficient distributed implementation of the coverage pattern mining technique, referred to as the Distributed Coverage Pattern Mining *DCPM* approach. Further, we investigate and introduce an Ad-slot allocation framework in display advertising with multiple ad-slots per web page, which is quite scalable for large click-stream datasets by leveraging the *DCPM* approach.

1.4.1 DCPM approach

Given a Transactional DataBase (TDB) over a set of items, a coverage pattern (CP) [53] is a set of items or a pattern whose items cover a certain percentage of transactions in TDB. A coverage pattern

is associated with Relative Frequency (RF), Coverage Support (CS), and Overlap Ratio (OR) measures (the details of these measures will be provided in Section 3). Given TDB and user-specified CS, OR, and RF constraints, the problem is to extract all CPs from TDB. In the literature, a level-wise candidate generation approach, which we designate as Coverage Pattern Mining (*CPM*) algorithm, was proposed in [53].

The existing *CPM* algorithm was proposed by considering a single machine using a level-wise candidate generation approach to extract CPs from the given TDB. Typically, the number of candidate patterns at each level is very large. Given limited main memory and CPU power, computing and validating such a huge number of candidate patterns on a single machine is very time-consuming. We propose a Distributed Coverage Patterns Mining (*DCPM*) algorithm to improve the performance.

In this proposed approach, we employ the notion of an Inverse Transactional Database (ITD), a vertical form of TDB, and replicate it at every machine in the distributed system. We then apply a distributed implementation of the level-wise (*CPM*) algorithm. Moreover, we propose a clustering-based method to distribute the task of extracting CPs among the worker nodes to validate candidate patterns in an efficient manner. We have conducted a performance evaluation of the new proposed approach using click-stream and real-time datasets. We observed that the *DCPM* approach performs significantly better than the existing state-of-the-art approaches, both in terms of data shuffled across machines and total execution times.

1.4.2 An efficient ad-slot allocation framework

Given a *publisher*, set W of web pages, set Φ_s of ad-slots, web-page to ad-slots mapping data M and a set $\mathbb{A}$ of advertisers along with their demand in terms of impressions, the framework proposes an efficient ad-slot allocation approach such that the total revenue generated for the publisher is maximized.

The proposed framework comprises three steps. First, we extract all the sets of ad-slot patterns $ASPs$ (a set of ad-slots) with an overlap of user views of an advertisement less than a certain threshold value using the *DCPM* approach. In addition to the ad-slot patterns, we extract all the low-frequency ad-slots (with a frequency less than a particular threshold, i.e., minRF). Second, we compute the number of impressions associated with each ad-slot pattern. Third, we propose an efficient ad-slot allocation strategy using binary search to allocate ad-slot patterns and low-frequency ad-slots to advertisers based on demand.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

- In Chapter 2, we discuss the literature review.
- In Chapter 3, we explain the background of coverage pattern mining.

- In Chapter 4, we present the most efficient coverage pattern mining algorithm, i.e., the DCPM algorithm, and show the enhancement in performance through experiments.
- In Chapter 5, we present the new ad-slot allocation approach for display advertising with multiple ad slots per web page and present the experiments for the same
- In Chapter 6, we provide the summary of the thesis and discuss future research directions.

Chapter 2

Related Work

In this chapter, we also discuss the related work on various parallel approaches in the field of pattern mining. Further, we discuss the related work on ad-slot allocation in banner advertising. Lastly, we present the related work on coverage patterns and various mining techniques and applications.

2.1 Parallel approaches in pattern mining

In the work [66], a method to optimize the Apriori algorithm using the Hadoop-MapReduce framework is presented. By partitioning the dataset and distributing it across multiple nodes, the algorithm achieves parallel and scalable processing. It employs a two-phase approach: local processing to identify frequent itemsets and global merging, then pruning using a hash-based data structure.

The work in [28] presents an enhanced version of the Apriori algorithm that incorporates coding techniques and the MapReduce framework. The objective is to optimize the performance of the algorithm while addressing scalability and efficiency concerns. The proposed approach introduces coding mechanisms to reduce intermediate data size and minimize the amount of data transferred during the MapReduce process. Additionally, it utilizes the parallel processing capabilities of MapReduce for efficient computation.

The work [13] focuses on addressing the challenges of mining frequent itemsets in large-scale datasets. It proposes an efficient approach that leverages parallel processing and distributed computing techniques to handle big data. The authors propose an algorithm that partitions and distributes the dataset across multiple nodes in a parallel computing environment. They utilize an optimized data structure to store and process the itemsets, enabling efficient mining of frequent itemsets.

The authors in [16] propose an efficient approach called BIGMiner for mining frequent patterns in large-scale datasets. It utilizes distributed computing to parallelize the mining process and introduces optimizations such as data partitioning, pruning strategies, and memory management.

A method for efficiently mining fuzzy high-utility patterns in large datasets using the Hadoop framework is proposed in [65]. It introduces a parallel algorithm that distributes the dataset across multiple nodes in a Hadoop cluster, allowing for scalable processing. The method incorporates fuzzy extensions

to high-utility pattern mining and employs pruning techniques and optimizations to enhance performance.

The authors in [36] introduce a method for mining linguistic frequent patterns using a compressed structure. The proposed approach aims to efficiently extract frequently occurring patterns in textual data. It employs a compressed data structure to store linguistic patterns, reducing memory consumption and improving processing speed. The method utilizes pattern growth techniques to mine frequent patterns and incorporates pruning strategies to optimize the mining process.

The work in [44] introduces YAFIM, a parallel algorithm for frequent itemset mining utilizing Apache Spark. YAFIM is designed to efficiently handle large-scale datasets by leveraging the distributed computing capabilities of Spark. The algorithm divides the dataset into partitions and performs local mining on each partition, followed by combining the results to obtain the global frequent itemsets. YAFIM incorporates optimizations such as pruning techniques and memory management strategies to improve performance.

The authors of [67] present a parallel and improved version of the Apriori algorithm using the Spark framework. The proposed algorithm, implemented on a distributed computing platform, enhances the efficiency and scalability of frequent itemset mining. It utilizes Spark's parallel processing capabilities to divide the dataset into partitions, perform local mining on each partition, and combine the results to obtain global frequent itemsets. The algorithm incorporates optimizations, such as candidate pruning and memory management techniques, to improve performance.

In [27], authors introduce a distributed algorithm for frequent itemset mining using the Spark framework. The proposed algorithm is designed to efficiently mine frequent itemsets from large-scale datasets by leveraging Spark's distributed computing capabilities. It divides the dataset into partitions and performs parallel mining on each partition using the Apriori algorithm. The results from each partition are combined to obtain the global frequent itemsets. To improve performance, the algorithm incorporates optimizations such as pruning strategies and memory management techniques.

The work in [48] presents Adaptive-Miner, an efficient distributed algorithm for association rule mining on the Spark framework. Adaptive-Miner is designed to handle large-scale datasets and leverage the parallel processing capabilities of Spark. The algorithm partitions the dataset and performs distributed mining on each partition using an adaptive strategy that dynamically adjusts the mining process based on the characteristics of the data. It incorporates optimizations such as pruning techniques and memory management strategies to enhance performance.

The work in [51] introduces HFIM, a hybrid frequent itemset mining algorithm designed for big data processing using the Spark framework. HFIM leverages Spark's parallel processing capabilities to efficiently mine frequent item sets from large-scale datasets. The algorithm combines two mining techniques, Apriori and FP-Growth, to balance memory usage and computational efficiency. It partitions the dataset and performs parallel mining using a combination of the two techniques to extract frequent itemsets. HFIM incorporates optimizations such as data compression and pruning strategies to improve performance.

The work in [47] introduces a MapReduce-based algorithm for efficient coverage pattern mining. The algorithm is specifically designed to address the challenges posed by large-scale datasets. It leverages the parallel processing capabilities of MapReduce to distribute the mining process across multiple compute nodes, enabling faster computation and scalability. The algorithm partitions the dataset into smaller segments, where each segment is processed independently by a MapReduce task. During the Map phase, frequent itemsets are extracted from each segment, and in the Reduce phase, the results are combined to generate the final coverage patterns. This parallel processing approach significantly reduces the execution time compared to traditional sequential algorithms. The experimental evaluation demonstrates the algorithm's scalability and performance in handling large transaction databases.

2.2 Ad-slot allocation in banner advertising

Banner advertising is an effective way to generate revenue for website owners. However, the challenge arises when it comes to deciding which ad to display in which slot on a webpage. Ad-slot allocation is an important problem that has been addressed by several research papers over the past couple of decades. This section summarizes some of the approaches proposed in the literature.

In the work [39], the researchers formulated the ad-slot allocation problem as an integer linear program (ILP) and used a branch-and-price algorithm to solve it. Their approach can handle multiple ad formats and constraints, such as frequency capping and targeting. They also proposed a heuristic algorithm to solve the problem with large input size.

Authors in [12] explore the use of greedy bidding strategies in keyword auctions. The authors propose a mathematical framework for analyzing the auction dynamics and optimizing bidding decisions. They consider keyword value, budget constraints, and ad rankings to develop novel greedy algorithms. By leveraging these algorithms, advertisers can aim to maximize their utility in the auction. Through mathematical analysis and empirical evaluations, the paper provides insights into the effectiveness of the proposed bidding strategies in keyword auctions.

A game-theoretic approach was proposed to model the ad-slot allocation problem in [43]. They formulated the problem as a non-cooperative game and used a Nash equilibrium solution concept to allocate the ad slots. Their approach can handle dynamic bidding and changing market conditions. They also proposed a simulation-based algorithm to test their approach and showed that it performs well in practice.

The work in [24] proposes a budget-constrained ad-slot allocation algorithm that considers both the expected revenue from displaying an ad in a slot and the cost of reserving the slot for the ad. They modeled the problem as a mixed-integer linear program and used a column generation approach to solve it. Their approach can handle multiple ad formats and constraints, such as budget constraints and ad relevance. They also showed that their approach performs well in practice.

In [41], authors proposed a model that considers the relationship between the advertiser's bid, the click-through rate (CTR), and the resulting cost per click. They analyze the optimal bidding strategy for

advertisers to maximize their utility and provide insights into the factors that influence CPC pricing. By examining the mathematical formulation and logical reasoning behind CPC pricing, the paper offers a comprehensive understanding of this pricing model in the context of display advertising.

A reinforcement learning algorithm to learn the optimal ad-slot allocation policy is proposed in [64]. They used a deep Q-network to learn the optimal policy based on historical ad performance and website visitor behavior. Their approach can handle multiple ad formats and dynamic changes in the ad market. They also showed that their approach performs better than existing heuristics regarding revenue generation.

In the work [72], researchers proposed a machine learning algorithm to predict the optimal ad-slot allocation. They used a support vector regression algorithm to predict the click-through rate of each ad in each slot and then allocated the slots based on the predicted click-through rates. Their approach can handle multiple ad formats and dynamic changes in the ad market. They also showed that their approach outperforms existing heuristics regarding revenue generation.

The work in [69] provides a comprehensive measurement and analysis of RTB systems. They conduct empirical studies and data analysis to examine various aspects of RTB, including bid landscape, bidding strategies, and ad performance. Their findings offer valuable insights into the effectiveness and dynamics of real-time bidding, contributing to a deeper understanding of this important aspect of online advertising.

The work [6] addresses the problem of budget optimization in sponsored search. They propose a censored learning approach within Markov Decision Processes (MDPs) to allocate budgets and effectively maximize performance in sponsored search advertising.

The authors in [10] explored the application of reinforcement learning in real-time bidding for display advertising. They propose a framework that leverages reinforcement learning techniques to optimize bidding strategies in an online auction environment. Their approach considers factors such as user profiles, ad characteristics, and historical bidding data to dynamically adapt bidding decisions. Experimental results demonstrate the effectiveness of their approach in maximizing the expected utility and improving performance in display advertising.

The work [73] introduced the idea of utilizing deep reinforcement learning to optimize online advertising impressions within recommender systems. The model can effectively capture complex patterns and representations from the available data by employing deep neural networks. The reinforcement learning component enables the system to learn and adapt its advertising impression strategies based on feedback and rewards received in real-time. This combination of deep reinforcement learning allows for a dynamic and data-driven approach to improve the recommendation process and maximize the impact of online advertising impressions.

The authors in [15] proposed an approach for ad-slot allocation in sponsored search that considers both the ad's relevance to the query and the willingness of the advertiser to pay for the ad. They modeled the problem as a Markov decision process and used dynamic programming to find the optimal ad-slot

allocation policy. Their approach is able to handle multiple ad formats and constraints, such as budget constraints and ad relevance. They also showed that their approach performs well in practice.

The work in [62] proposed a hybrid genetic algorithm for ad-slot allocation that combines a local search algorithm with a genetic algorithm. They used the local search algorithm to generate initial solutions and the genetic algorithm to refine the solutions. Their approach is able to handle multiple ad formats and constraints, such as budget constraints and ad relevance. They also showed that their approach performs better than existing heuristics regarding revenue generation.

Work in [34] proposed a budget-constrained approach for ad-slot allocation that maximizes the expected revenue. They formulated the problem as an integer linear program and used a branch-and-bound algorithm to solve it. Their approach takes into account the budget constraints of the advertisers and is able to handle multiple ad formats.

The authors in [14] proposed a novel idea that involves modeling the bidding behavior of advertisers as a reinforcement learning problem. Specifically, they formulate the bidding process as a Markov Decision Process (MDP) and use value iteration to solve for the optimal bidding strategy. By considering factors such as historical bidding data, budget constraints, and expected rewards, they demonstrate that reinforcement learning can effectively guide advertisers in making optimal bidding decisions in sponsored searches. The mathematical formulation and solution approach of MDP and value iteration provide a rigorous framework for understanding and optimizing advertiser behaviors in sponsored search auctions.

In work [38], the authors investigate the challenge of designing auction mechanisms that consider advertisers' dual objectives and preferences for both visibility (views) and engagement (clicks). They propose a novel auction framework that incorporates advertisers' sensitivity to views and clicks, allowing for a fair and efficient ad space allocation. Through theoretical analysis and simulations, the paper sheds light on the trade-offs and complexities involved in designing auctions that cater to advertisers' dual sensitivity, contributing valuable insights to the field of online advertising auctions.

The work in [17] introduces an approach for ad-slot allocation that uses deep reinforcement learning to learn the optimal ad-slot allocation policy. They used a deep Q-network to learn the policy based on historical ad performance and website visitor behavior. Their approach is able to handle multiple ad formats and dynamic changes in the ad market. They also showed that their approach performs better than existing heuristics in terms of revenue generation.

The authors in [68] proposed an approach for ad-slot allocation that uses a multi-objective genetic algorithm to optimize both the revenue and the click-through rate of the ads. They modeled the problem as a multi-objective optimization problem and used a genetic algorithm to find the Pareto-optimal solutions. Their approach is able to handle multiple ad formats and constraints, such as budget constraints and ad relevance. They also showed that their approach performs better than existing heuristics in terms of revenue and click-through rate.

2.3 Coverage Patterns

The notion of coverage was first introduced to mine interesting patterns called coverage patterns in the literature by [53, 57]. The authors proposed an iterative algorithm called Coverage Pattern Mining (CPM) to extract coverage patterns by leveraging the sorted closure property [37].

To address the limitations of the CPM approach, a novel technique known as Coverage Pattern with Pattern Growth (CPPG) was proposed [55]. This approach introduces the concept of a "non-overlap projected database (NOPD)" to overcome the need for multiple scans of the database. By utilizing the NOPD, CPPG recursively extracts CPs for each item. Additionally, the authors introduced the notion of minimal CPs [56] and introduced the Enhanced Coverage Pattern Projected Growth (ECPPG) algorithm to extract a minimal set of CPs.

In another study [60], researchers proposed a model for content-specific coverage patterns and an associated extraction algorithm. Their approach focuses on filtering out non-potential coverage patterns based on the semantics of the items in the database.

Exploiting the notion of overlap and coverage in coverage patterns, a framework for allocating advertisement slots in banner advertising (or display advertising) was proposed in [31]. The framework involves extraction, ranking and allocation of web pages for advertising and has been shown to enhance publisher's revenue compared to other rudimentary approaches.

In the realm of sponsored search advertising, researchers explored a mechanism to enhance the utilization of ad space for tail query keywords [9]. This approach creates an opportunity for advertisers to bid on "concepts" instead of keywords by utilizing level-wise coverage patterns to allocate incoming search queries efficiently. Furthermore, an improved allocation framework incorporating coverage and concept taxonomy was proposed in [7, 8]. Advertisers can bid on potential concepts in this framework, leading to improved ad space utilization for tail queries.

Incremental approaches for efficiently mining coverage patterns when new databases are added have been proposed [46]. In [30], a Comprehensive Coverage Pattern Mining (CCPM) approach for efficiently extracting Coverage Patterns incrementally for more actions like set of transactions are deleted only, and both sets of transactions are added and deleted from the original database. Additionally, a parallel algorithm for extracting coverage patterns within the MapReduce framework was introduced [47].

In the field of visibility mining, researchers introduced the notion of Multi-Location Visibility (MLV) queries [23]. These queries aim to identify the top-k query locations that maximize the visibility of a given target object. The proposed approach incorporates a portion-based transactional framework and a coverage pattern-based mining algorithm to process MLV queries.

2.4 Summary

In this chapter, we have explained all the existing work related to coverage pattern mining, parallel pattern mining approaches, and ad-slot allocation approaches in banner advertising.

The proposed approaches in this thesis are different from the related work in the following manner. In the field of coverage pattern mining, the idea of parallel implementation has only been explored in [47]. The proposed approach is an improved approach over [47]. Also, no ad allocation framework has been proposed in the literature by considering multiple slots per page.

In the next section, as a part of the background, we explain coverage pattern mining.

Chapter 3

Background

In this section, we provide details on the required background on coverage patterns for our proposed approaches. We first start by presenting the model of coverage patterns, and related terminologies and talk about the Coverage Pattern Mining (CPM) algorithm in detail.

3.1 Coverage patterns

3.1.1 Model of coverage patterns

Given a transactional database *TDB*, where each transaction is a subset of a set *I* of *m* items $\{i_1, i_2, i_3, \ldots, i_m\}$. Each transaction $T \in TDB$ has a unique identifier TID. Let T^{i_k} denotes the set of $TIDs$ in which item i_k is present. Coverage patterns are characterized by the notions of relative frequency, coverage support, and overlap ratio. We now define these parameters as follows.

Definition 1. Relative frequency The relative frequency of an item i_k is the fraction of transactions containing item i_k to the total number of transactions in TDB. Formally, defined as follows:

$$RF(i_k) = \frac{\mathsf{T}^{i_k}}{\mathsf{TDB}}$$

Here, $0 \leq RF(i_k) \leq 1$. An item i_k is considered as *frequent* if its $RF(i_k) \geq minRF$, where $minRF$ is the user-specified minimum Relative Frequency threshold. A pattern *P* is a subset of items in *I* i.e., $P \subseteq I$, $P=\{i_p, i_q, \ldots, i_r\}$, where $1 \leq p, q, r \leq m$. The *Coverage Set* (*CSet(P)*) of a pattern P is the set of all the transactions that contain at least one item from the pattern *P* i.e., *CSet(P)*=$T^{i_p} \cup T^{i_q} \cup \ldots \cup T^{i_r}$.

A pattern P is interesting when it covers more number of transactions in TDB. The measure of coverage of TDB is represented by coverage support.

Definition 2. Coverage support The *Coverage Support* of a pattern P (*CS(P)*) is the ratio of the size of *CSet(P)* to the size of TDB. Formally, defined as follows:

$$CS(P) = \frac{\mathsf{CSet(P)}}{\mathsf{TDB}}$$

A pattern with high coverage may not be interesting if there is a large overlap among the transactions covered by items in a pattern. In the literature, the concept of overlap between sets is most often described using Euler diagrams [5]. Consider two sets A and B in a universe of objects. The overlap of A and B is computed by $\frac{A \cap B}{A \cup B}$. However, this overlap does not satisfy *downward closure property*. Therefore, the measure of overlap among the transactions covered by items in a pattern is captured by *overlap ratio* metric. The *overlap ratio* helps in pruning the candidate patterns and identifying items to be added to a pattern such that the coverage support increases significantly. Given a pattern *P*, the notion of *overlap ratio* of *P* satisfies the sorted closure property [37] when the items in *P* are sorted in decreasing order of their relative frequencies i.e., $RF(i_p) \geq RF(i_q) \geq \ldots \geq RF(i_r)$.

Definition 3. Overlap ratio The *Overlap Ratio* of a pattern *P* (*OR(P)*) is the ratio of the number of transactions that are common between *CSet(P*$-i_r$*)* and T^{i_r} to the number of transactions in T^{i_r}. Formally, defined as follows:

$$OR(P) = \frac{\mathsf{CSet(P\text{-}i_r)} \cap T^{i_r}}{\mathsf{T}^{i_r}}$$

The computation of the overlap ratio of a pattern is an iterative process. To compute the overlap ratio of a pattern X with l items, first, we need to compute the overlap ratio of pattern X with $(l-1)$ items (without the last item).

A high value of coverage support indicates more number of transactions and a low value of overlap ratio means less repetitions among the transactions covered. A pattern is *interesting* if its coverage support is greater than or equal to the user-specified minimum Coverage Support threshold value (*minCS*) and its overlap ratio is less than or equal to the user-specified maximum Overlap Ratio threshold value (*maxOR*). Given the values of *minRF*, *minCS* and *maxOR*, a pattern P=$\{i_p, i_q, \ldots, i_r\}$ is considered as a *coverage pattern* if *RF(*i_k*)* $\geq$ *minRF* $\forall\ i_k \in P$, *CS(P)* $\geq$ *minCS* and *OR(P)* $\leq$ *maxOR*. We now explain the method to compute coverage support and overlap ratio as follows.

Consider a pattern X=$\{a, d, e\}$ sorted in decreasing order of their RF values. From dataset shown in Figure 3.1, the RF values of a, d, and e are 0.5, 0.4 and 0.4 respectively. The cover set of X, $CSet(X)$=$\{1,2,3,4,5,6,7,8,9,10\}$. The coverage support of X, $CS(X)$=$\frac{\mathsf{CSet(X)}}{\mathsf{TDB}}$=$\frac{10}{10}$=1.0. The computation of the overlap ratio of a pattern is an iterative process. To compute the overlap ratio of X=$\{a, d, e\}$, first, we need to compute the overlap ratio of its sub-pattern X'=$\{a, d\}$. The overlap ratio of a X', $OR(X')$=$\frac{\mathsf{CSet(a)} \cap T^d}{\mathsf{T}^d}$ = $\frac{1}{4}$ = 0.25. Next, the overlap ratio of X, $OR(X)$= $\frac{(\mathsf{CSet(a)} \cup CSet(d)) \cap T^e}{\mathsf{T}^e}$ = $\frac{\mathsf{CSet(X\text{-}\ \{e\})} \cap T^e}{\mathsf{T}^e}$=$\frac{2}{4}$ = 0.5. Given the values of $minRF$ = 0.4, $minCS$ = 0.7 and $maxOR$=0.5, the pattern X = $\{a, d, e\}$ is an *coverage pattern*.

TID	1	2	3	4	5	6	7	8	9	10
Items	a,b,c	a,c,e	a,c,e	a,c,d	b,d,f	b,d	b,d	b,e	b,e	a,b

Figure 3.1: Sample transactional database (TDB).

3.1.2 Coverage pattern mining algorithm CPM

We now present a level-wise apriori-based coverage pattern mining algorithm proposed in [54], which exploits the sorted closure property of *overlap ratio* for efficient pruning of candidate patterns.

Given a transactional database TDB with 10 transactions containing items set I={a,b,c,d,e,f} as depicted in Figure 3.1. Let the user-specified parameters be *minRF*=0.4, *minCS*=0.7 and *maxOR*=0.5. We present the coverage pattern mining algorithm with a working example.

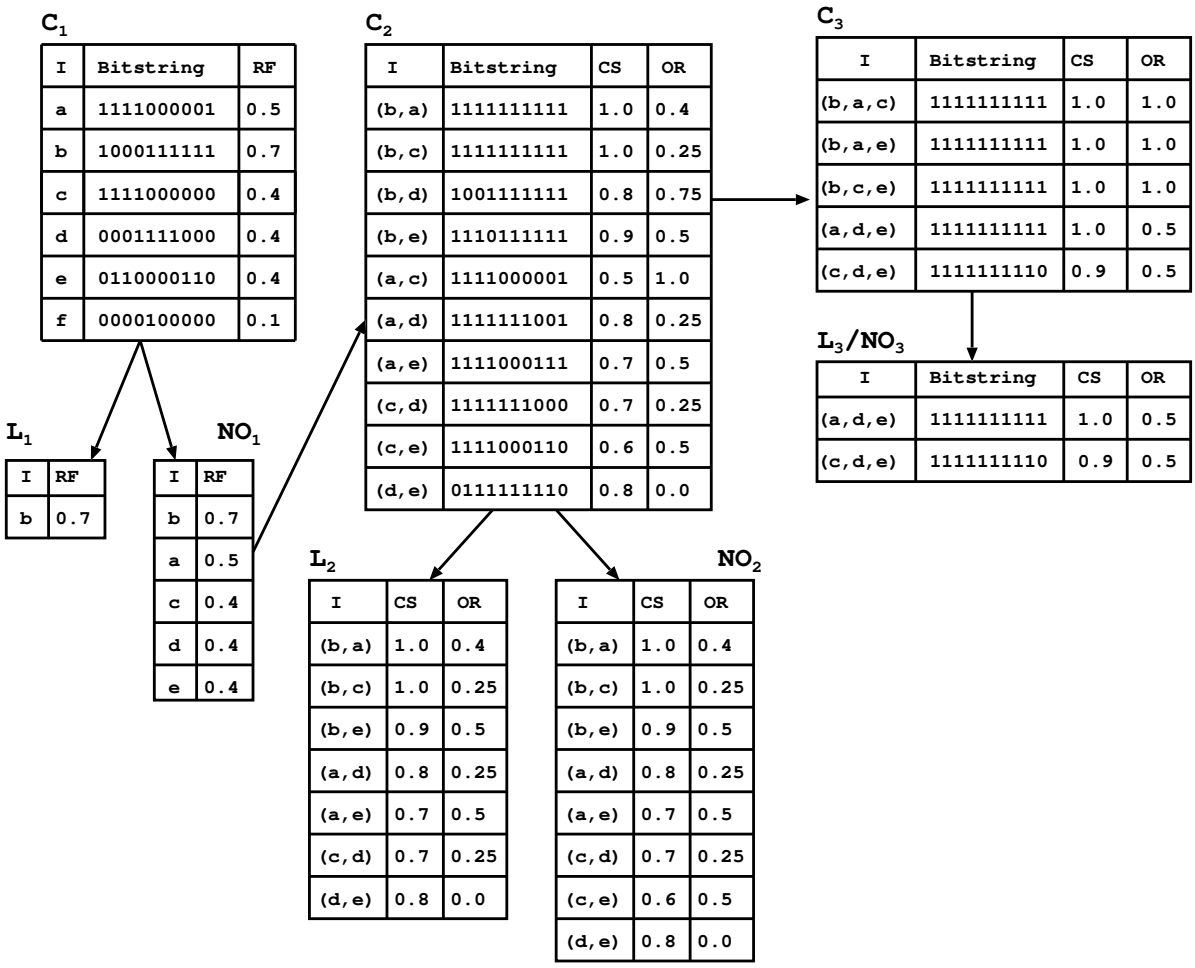


Figure 3.2: Coverage pattern mining algorithm working example for *minRF*=0.4, *minCS*=0.7 and *maxOR*=0.5.

The coverage pattern mining algorithm is an apriori kind of level-wise approach, where patterns of size k patterns were generated from the non-overlap patterns of size k-1. Let C_k, NO_k, and L_k denote the k-size candidate set, non-overlap pattern set, and coverage pattern set respectively. The algorithm begins by scanning the transactional database and extracts a set of all candidate items C_1 along with their relative frequencies. The set of items that satisfy *minRF* is termed as non-overlap patterns of size 1 (NO_1). The set of items that satisfy the *minCS* constraint are designated as coverage patterns of size 1 (L_1) as depicted in Figure 3.2 Table C_1, NO_1 and L_1. In the next step, NO_1 as a seed set, the algorithm generates the candidate pattern set C_2 by executing a self-join operation over NO_1, i.e C_2=$NO_1 \bowtie NO_1$. From C_2, the patterns that satisfy *maxOR* constraint are considered as 2-size non-overlap pattern set NO_2 and that satisfy *maxOR* and *minCS* are considered as 2-size coverage pattern set L_2 as depicted in Figure 3.2 Table C_2, NO_2 and L_2. Next, the sets C_3, NO_3 and L_3 are generated in a similar method as depicted in Figure 3.2 Table C_3, NO_3 and L_3. Here, NO_3 and L_3 are the same because there is no further generation of the candidate set and the algorithm stops. Normally, at each k-stage, there are two phases consisting of *join phase* and *prune phase*. In the joining phase, the candidate set was generated and in the pruning phase, the set of candidate patterns that do not satisfy *maxOR* threshold was pruned. The coverage pattern mining algorithm terminates when there are no non-overlap patterns or no candidate patterns generated. This algorithm employs bit strings representation of patterns, which replaces expensive set-based computation of coverage support and overlap ratio with simple and relatively faster bit-wise AND and OR operations.

In addition to that, a pattern growth-based coverage pattern mining technique, the Coverage Pattern Projected Growth (*CPPG*) algorithm [55] was developed, which is an improved method that leverages the concept of non-overlap pattern projection. Lastly, researchers have also proposed the Enhanced Coverage Pattern Projected Growth Method (*ECPPG*) in [56] to enhance further the performance of *CPPG* by utilizing minimal coverage pattern and exploiting the upward closure property to avoid the computation of unnecessary projections.

3.2 Summary

In this chapter we have presented the background information related to coverage patterns and ad-slot allocation in banner advertising. We also presented the problem of coverage patterns and coverage algorithms for efficient extraction of coverage patterns from flat transactional datasets. In the next chapter, we present an efficient distributed coverage pattern mining algorithm

Chapter 4

Distributed coverage pattern mining

This chapter presents an efficient distributed pattern mining approach *DCPM* to extract coverage patterns from transactional data sets. After the introduction, we discuss the basic idea of the approach and then present our proposed approach DCPM. We then run an illustration to explain the working of the approach. Lastly, we present the performance evaluation of our approach with existing methods on both synthetic and real click-stream data sets.

4.1 Introduction

Notably, the existing *CPM* algorithm [53] was proposed by considering a single machine, using a level-wise candidate generation approach to extract CPs from the given TDB. Typically, the number of candidate patterns at each level is enormous. Given limited main memory and CPU power, computing and validating such a massive number of candidate patterns on a single machine is very time-consuming.

In [47], an effort has been made to employ a MapReduce paradigm to compute CPs. In this approach, the TDB is partitioned among the participant sites. As the size of the TDB and the number of items increase, this approach would encounter performance problems because a large number of candidate patterns are to be transmitted at each level. In the literature, the notion of ITD is employed for extracting frequent patterns by the ECLAT [42] approach, which extracts frequent item sets in a depth-first search manner. An effort has been made to propose a distributed ECLAT [45, 52] based on the Spark framework. The notion of ITD is employed in ECLAT to improve the performance of extracting frequent item sets by computing and validating the frequent item sets in a distributed manner. So far, the notion of ITD has not been extended to extract CPs.

Considering the challenges mentioned above in the existing coverage pattern mining approaches, we propose an efficient Distributed Coverage Patterns Mining (*DCPM*) algorithm. In this approach, we employ the notion of Inverse Transactional Database (ITD), a vertical form of TDB, and replicate it at every node. Moreover, we propose a clustering-based method to distribute the task of extracting CPs among the worker nodes. Notably, the performance could be significantly improved by replicating TDB using the notion of ITD and employing a clustering-based approach to compute and validate candidate

patterns in a distributed manner. The results of the extensive performance study on both real-world and synthetic datasets show that the performance of the proposed approach significantly improves over the state-of-the-art approaches in terms of execution time and data shuffled across nodes.

4.2 Basic Idea

In the centralized *CPM* algorithm, candidate patterns are generated at each level. These candidate patterns are validated by accessing the entire TDB. To improve the performance, we incorporate the following ideas in the proposed Distributed *CPM* (*DCPM*) approach.

First, we efficiently distribute the TDB by computing the Inverted Transactional Database (ITD), which is a vertical form of the database, where each item is associated with a set of transaction identifiers to which it belongs. The ITD constitutes the database of all items, which satisfies the $minRF$ threshold. As a result, a reduced ITD is replicated to each node. Second, at each iteration, we distribute the $NOPs$ in a balanced manner. While distributing, we group $NOPs$, which differ only in the last item, and send the group to each node. With this, the computation of coverage support of a given candidate pattern can exploit the reuse at the local node. For example, if {a,b,c} and {a,b,d} are candidate patterns, the coverage set of {a,b}, which is being calculated while computing the coverage of {a,b,c} can be re-used to calculate the coverage of {a,b,d}. Moreover, significant savings in the transmission cost is achieved due to sending a group of candidate patterns in a clustered form instead of individual candidate patterns.

The overview of the proposed *DCPM* approach is as follows. We model the given transactional dataset into ITD and broadcast it to all Worker nodes. Each item in the ITD that satisfies the $minRF$ constraint forms the first level $NOPs$, and items that meet the $minCS$ constraint form the coverage patterns. For the second level iteration, we distribute the first level $NOPs$ and compute the second level $NOPs$ by joining the first level $NOPs$ with corresponding lesser frequent items in the ITD. From the next level onwards, the NOP clusters from the previous level are distributed, and next-level $NOPs$ and CPs are computed by utilizing the ITD.

4.3 Proposed Approach

We first present the terminologies used and then introduce the cluster data structure to explain the proposed approach.

4.3.1 Terminologies

Definition 4. ***Inverted Transactional Database (ITD)*** Given a TDB over a set I of items, ITD is a list of the form $< x,\ L(x) >$, where $x \in I$ and $L(x)$ is a set of TIDs in which the item x belongs.

Definition 5. ***Penultimate Prefix (PP)*** Consider a pattern X=$\{i_p, i_q, \ldots, i_r\}$, where $\|T^{i_p}\| \geq \|T^{i_q}\| \geq \ldots \geq \|T^{i_r}\|$. The penultimate prefix of a pattern X, which we refer to as $PP(X)$, is equal to the set of items excluding the last item i_r, i.e., $PP(X) = X - \{i_r\}$.

Definition 6. ***Penultimate Prefix Cluster (PPC)*** Consider a set S of patterns of equal size and each pattern $X \in S$ is of the form X=$\{i_p, i_q, \ldots, i_r\}$, where $\|T^{i_p}\| \geq \|T^{i_q}\| \geq \ldots \geq \|T^{i_r}\|$. Consider that all patterns in S have a common PP. Then, the Penultimate Prefix Cluster of S, which we refer to as PPC(X), is equal to $< CPP, LLI >$, where set CPP is the common penultimate prefix, which contains the set of items in PP and set LLI is the list of last items, which contains the last item of each pattern in S.

4.3.2 DCPM approach

Given a transactional dataset TDB, number of machines N, $minRF$, $minCS$, and $maxOR$, the proposed *DCPM* approach extracts all coverage patterns from TDB in a distributed environment. We adopt a Master-Slave architecture for the distributed environment. We designate one of the nodes as the Master node and the remaining nodes as Worker nodes. The Master node receives the input, distributes it among the Workers, and coordinates the computation. The computation is carried out in an iterative manner starting from the first level (k = 1). The Master terminates the computation when there are no more $NOPs$ or candidate patterns at the given level.

We explain the procedures followed by both Master and Worker for the first iteration, second iteration, and other iterations. The steps of *DCPM* at level k = 1, 2, and beyond 2 for Master and Worker are provided in Algorithm 1 and Algorithm 2 respectively.

(i) First level (k=1): This level involves computation of inverse transactional database ITD, NOP_1 and CP_1 (1-sized $NOPs$ and CPs). (At k = 1, the algorithm for Master is given in Algorithm 1 and the algorithm for Worker is given in Algorithm 2)

Master: Master distributes the TDB among N Worker nodes equally, i.e., each Worker receives a total number of $\frac{\mathsf{TDB}}{N}$ transactions to compute ITD.

Worker: Worker receives a TDB partition and computes the partial ITD of the TDB partition received, and returns it to the Master.

Master: The Master receives all the partial ITDs from the Workers and merges them to compute the final ITD. Next, NOP_1 and CP_1 are computed by comparing the RF of the items in the ITD with $minRF$ and $minCS$ respectively. The items in the ITD, which have lesser frequency values than $minRF$ are pruned. The resultant ITD is then sorted in decreasing order of RF and broadcasted to all the Workers for further levels.

Worker: It receives the final ITD and saves it.

Algorithm 1 Master()

Variables: N: Number of Worker nodes, NOP_k: Set of k-size non-overlap patterns, CP_k: Set of k-size coverage patterns, PPC: Penultimate Prefix Cluster, ITD: Inverse Transactional Database, CPP: Common Penultimate Prefix, LLI: List of Last Items.

k=1 (First Level)

1: Partition the dataset into N equal parts
2: Distribute the N partitions among N Workers
3: Collect Partial ITD from all the Workers
4: Merge all the partial ITDs and sort it in decreasing order based on the item RF.
5: Prune items in the ITD based on ***minRF*** constraint
6: Compute NOP_1, CP_1 from ITD
7: Broadcast ITD to all nodes

k=2 (Second Level)

1: Distribute NOP_1 among N Workers
2: Collect partial NOP_2 from each Worker ▷ Items in NOP_2 are modeled in the form of PPC
3: Merge all partial NOP_2 to form the complete NOP_2
4: Compute CP_2 by filtering patterns in NOP_2 with ***minCS*** threshold.

k>2 (Beyond Second Level)

1: Distribute the $PPCs$ in NOP_{k-1} among N Workers
2: Recieve partial NOP_k from all the Workers ▷ Items in NOP_k are modeled in the form of set of PPC
3: Merge all the partial NOP_k to form the complete NOP_k
4: Compute CP_k by filtering the patterns in NOP_k that satisfy the ***minCS*** threshold to obtain CP_k

Termination condition: The Master terminates the procedure at level k when either $NOP_k = \{\phi\}$ or the total number of $(k+1)$ size candidate patterns = 0.

(ii) Second level (k=2): In this level, we compute 2-size $NOPs$ and CPs from NOP_1 received from the first level. (At k = 2, the algorithm for Master is given in Algorithm 1, and the algorithm for Worker is given in Algorithm 2.)

Master: Master distributes NOP_1 among Workers such that each Worker receives only one item in NOP_1. The Master continues to assign the remaining items in the NOP_1 to Workers upon completion of the assigned task. The assignment process continues till all of the items in NOP_1 are exhausted.

Worker: The Worker receives one item from NOP_1 say x. The Worker joins x with an item in ITD, say y, such that $RF(x) \geq RF(y)$ and forms a 2-size candidate pattern $\{x, y\}$. The generated candidate pattern $\{x, y\}$ is said to be a 2-size NOP when it satisfies the $maxOR$ constraint. Note that all the lesser frequent items w.r.t. item x will be after the position of x in ITD as the ITD is sorted in decreasing order of RF. The Worker utilizes the ITD to compute the OR and CS for each candidate pattern. The computed $NOPs$ forms a set of $PPCs$ (Refer to the definition of PPC from Definition 6). The computed CS value of a pattern is attached with each NOP. The final set of $NOPs$, along with their respective CS value, is sent to the Master.

Algorithm 2 Worker()

Variables: N: Number of Worker nodes, NOP_k: Set of k-size non-overlap patterns, CP_k: Set of k-size coverage patterns, PPC: Penultimate Prefix Cluster, ITD: Inverse Transactional Database, CPP: Common Penultimate Prefix, LLI: List of Last Items.

k=1 (First Level)

1: Receive partial TDB from the Master
2: Compute partial ITD from the partial TDB
3: Send the partial ITD to the Master
4: Receive final merged ITD from Master
5: Store the ITD for further levels

k=2 (Second Level)

1: Receive partial NOP_1 from the Master
2: **for all** items i in partial NOP_1 **do**
3: Join item i with items in partial NOP_1 which have lesser frequencies to form new candidate patterns
4: Model the candidate patterns in the form of PPC: ($\{CPP, LLI\}$) ▷ Refer Level 2 in Illustrative Example
5: Compute CS and OR using ITD for each candidate pattern
6: Prunes all candidate patterns that do not satisfy ***maxOR*** threshold and forms partial NOP_2
7: **end for**
8: Send partial NOP_2 to Master

k>2 (Beyond Second Level)

1: Receive a PPC: $\{CPP, LLI\}$ in NOP_{k-1} from the Master
2: Compute the Coverage Set (CSet) of CPP.
3: **for all** items i in LLI **do**
4: Join CPP and i and calculate the new coverage set of $\{CPP, i\}$, $CSet$
5: **for all** item j in LLI where $RF(i) \geq RF(j)$ **do**
6: Compute CS and OR of the set $\{CPP, i, j\}$ using CSet
7: If the combination $\{CPP, i, j\}$ satisfies the $maxOR$ constraint, then add j into LLI of the PPC with new CPP :$\{CPP, i\}$.
8: **end for**
9: Update and add all the newly formed $PPCs$ into partial NOP_k
10: **end for**
11: Send partial NOP_k to Master

Master: It receives the partial NOP_2 along with their respective CS value and merges them to form NOP_2. The CP_2 (2-sized CPs) are computed by the Master by filtering out NOP_2 with ***minCS*** constraint.

(iii) k^{th} level (k>2): This level computes NOP_k and CP_k from NOP_{k-1}. (At k>2 (Beyond Second Level), the algorithm for Master is given in Algorithm 1 and the algorithm for Worker is given in Algorithm 2.)

Master: The Master distributes the $PPCs$ in NOP_{k-1} among the Workers such that each Worker receives one PPC. The Master continues to assign the remaining clusters to Workers upon completion of the assigned task. The assignment process continues till NOP_{k-1} is exhausted.

Worker: The Worker receives a PPC from the Master. The Worker generates candidate patterns by joining item x with item y such that $RF(x) \geq RF(y)$ and forms a candidate pattern $\{A,x,y\}$. Here, $x, y \in LLI$ and A is the CPP of the PPC. (Refer the definitions of LLI and CPP from the Definition 6). Note that all the items after item x in LLI have lesser frequencies than x because LLI is already sorted in decreasing order of relative frequency values. Here, LLI in any PPC is already sorted in decreasing order because the candidate patterns generation is done in an ordered manner. Further, the Worker computes OR and CS of the pattern $\{A,x,y\}$. The set of candidate patterns that satisfy the *maxOR* constraint will form the partial NOP_k and are sent to Master along with their individual CS values. The Worker computes the OR and CS in an efficient manner by avoiding the redundant set operations through reuse (Refer to lines 4, 5, 6 under Worker in Algorithm 2).

Master: The Master receives all the partial NOP_k along with CS values from Workers and merges all the partial NOP_k to compute NOP_k. Further, the Master extracts CP_k from the NOP_k subject to *minCS* constraint. The master terminates the process at a level when there is no possibility of generating candidate patterns for the next level.

4.3.3 An Illustrative Example

Table 4.1: Transactional Database

TID	1	2	3	4	5	6
Items	a, d	a, b, c	a, d	b, c	c	a, b, e

We demonstrate the working of *DCPM* through an example. The sample TDB is shown in Table 4.1. Consider one Master node and 3 Worker nodes $\{W_1, W_2, W_3\}$. Let the values of $minRF$, $minCS$, and $maxOR$ be 0.2, 0.8, and 0.8, respectively.

Level 1. Figure 4.1 depicts the first level of *DCPM*. The Master distributes the transactions with TIDs $\{1, 2\}$ to W_1, $\{3, 4\}$ to W_2 and $\{5, 6\}$ to W_3. Each Worker computes partial ITD for the transactions it has received and sends the partial ITD to the Master as shown in the Figure 4.1. The Master merges partial ITDs that are received from the three Workers and filters out item e as $RF(e) = \mathsf{ITD(e)/TDB} = 0.16$, which does not satisfy the *minRF* constraint. The items in the resultant ITD are sorted in decreasing order of their RF values. The resultant ITD is $\{a$:{1,2,3,6}, b:{2,4,6}, c:{2,4,5}, d:{1,3}$\}$ and it is sent to all Worker nodes. All the items in the filtered ITD form NOP_1 i.e., NOP_1 = $\{\{a\}, \{b\}, \{c\}, \{d\}\}$. For 1-size patterns, the CS of any item, say a is the same as RF, i.e., $CS(a) = RF(a) = \mathsf{ITD(a)/TDB} = 0.66$. Since no item in NOP_1 satisfy the $minCS$ constraint, CP_1 = $\{\phi\}$.

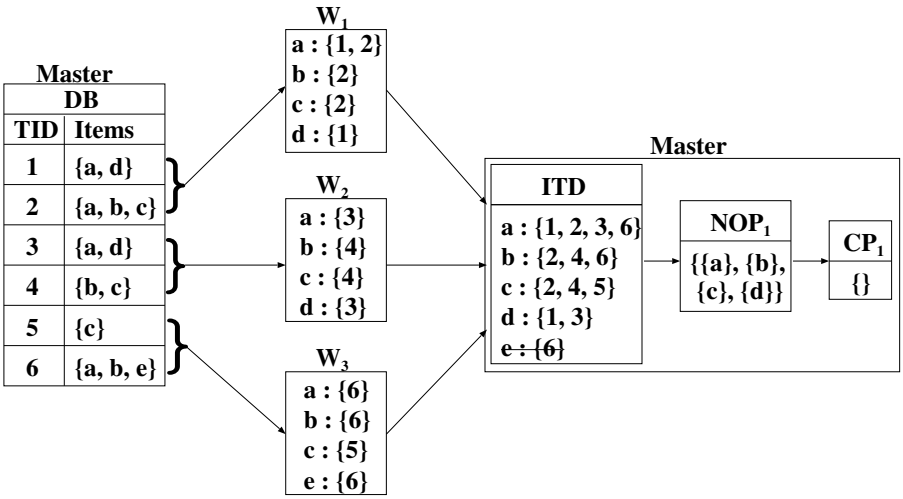


Figure 4.1: First level iteration

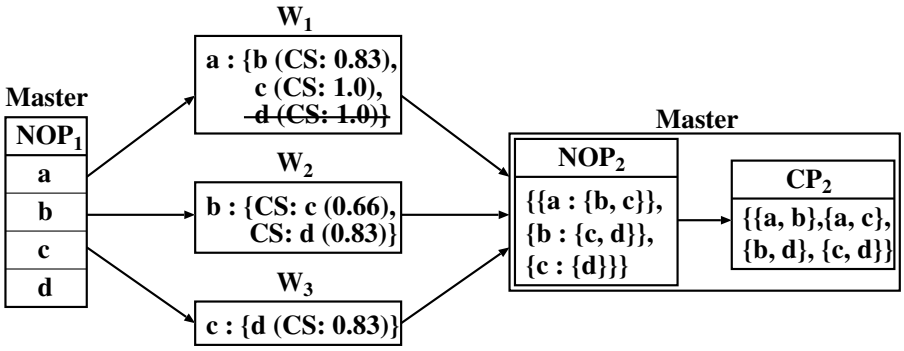


Figure 4.2: Second level iteration

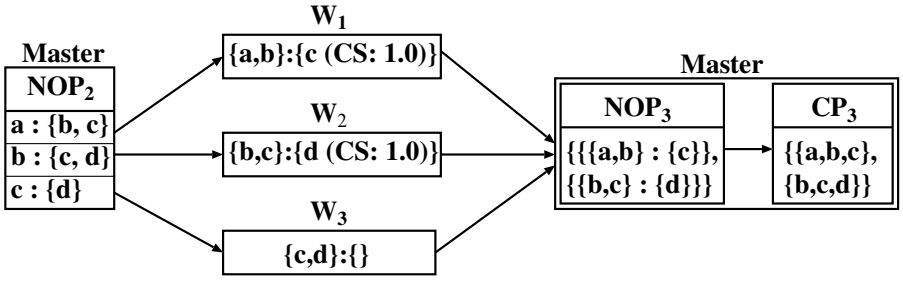


Figure 4.3: Third level iteration

Level 2. Figure 4.2 depicts the second level iteration of *DCPM*. The Master distributes the items $\{a\}$ to W_1, $\{b\}$ to W_2, and $\{c\}$ to W_3. Each Worker computes NOP_2 by pruning the candidate patterns that do not satisfy the *maxOR* constraint. Consider W_1, where item a can be joined with every other item. Thus, the set of possible combinations is $\{\{a,b\},\{a,c\},\{a,d\}\}$. Among these patterns, the pattern $\{a,d\}$ is pruned because $OR(\{a,d\}) = \frac{\|ITD(a) \cap ITD(d)\|}{\|ITD(d)\|} = 1.0$, which is greater than *maxOR* (0.8). Therefore, the $NOPs$ in the form of PPC set at Worker W_1 will be $\{\{a : \{b,c\}\}\}$. The Worker calculates the CS of each generated new pattern and attaches it as shown in Figure 4.2. The coverage support of $\{a,b\}$, $CS(\{a,b\}) = \frac{\|ITD(a) \cup ITD(b)\|}{\|TDB\|} = 0.83$. The Master collects all the second level partial $NOPs$ from Workers and forms $NOP_2 = \{\{a : \{b,c\}\},\{b : \{c,d\}\},\{c : \{d\}\}\}$. Note that NOP_2 here is already in PPC form and is equivalent to $\{\{a,b\},\{a,c\},\{b,c\},\{b,d\},\{c,d\}\}$. Furthermore, we obtain $CP_2 = \{\{a,b\},\{a,c\},\{b,d\},\{c,d\}\}$ by comparing the CS of each pattern in NOP_2 with the *minCS* constraint.

Level 3. Figure 4.3 depicts the third level iteration of *DCPM*. The Master distributes the $PPCs$ $\{a : \{b,c\}\}$, $\{b : \{c,d\}\}$, and $\{c : \{d\}\}$ to W_1, W_2, and W_3 respectively. Consider W_1 with the PPC $\{a : \{b,c\}\}$. Here, b is joined with c to form the candidate pattern $\{a,b,c\}$. W_1 computes the OR and CS for the generated pattern a,b,c using $OR(\{a,b,c\}) = \frac{\|(ITD(a) \cup ITD(b)) \cap ITD(c)\|}{\|ITD(c)\|} = 0.66$ and $CS(\{a,b,c\}) = \frac{\|(ITD(a) \cup ITD(b)) \cup ITD(c)\|}{\|TDB\|} = 1.0$. Since the OR of $\{a,b,c\}$ is 0.66, which is less than 0.8, the pattern $\{a,b,c\}$ forms an NOP and is sent to the Master. The Master collects all the 3^{rd} level partial $NOPs$ from Workers and forms $NOP_3 = \{\{a,b\} : \{c\},\{b,c\} : \{d\}\}$. Similarly, $CP_3 = \{\{a,b\} : \{c\},\{b,c\} : \{d\}\}$ is extracted by comparing the CS of each pattern in NOP_3 with the *minCS* constraint. Furthermore, no new 4^{th} level candidate patterns can be generated because each cluster has only one entry in its LLI. Hence, the *DCPM* terminates.

4.4 Performance Evaluation

We conducted our experiments in the ADA cluster [3] (at IIIT Hyderabad), which consists of 42 Boston SYS-7048GR-TR nodes equipped with dual Intel Xeon E5-2640 v4 processors, providing 40 virtual cores per node. The aggregate theoretical peak performance of ADA is 47.62 TFLOPS. We have conducted our experiments on a cluster of 24 virtual nodes. Each virtual node was allocated with 2 GB of memory.

We have used three datasets, namely BMS-POS dataset [25], the Mushroom dataset [18], and the T10I4D100K dataset [4]. The BMS-POS dataset is a click-stream dataset of an e-commerce company. This dataset consists of 515,596 transactions and 1,656 distinct items. The Mushroom dataset is a dense dataset having 8,124 transactions and 119 distinct items. Moreover, the T10I4D100K is a synthetic dataset generated by a dataset generator. This synthetic dataset has 100,000 transactions and 870 distinct items.

We conduct the experiments by varying the parameters relative frequency threshold ($minRF$), coverage support threshold ($minCS$), overlap ratio threshold ($maxOR$), and the number of machines

Table 4.2: Parameters used in our experiments

Dataset	Parameter	Default value	Variations
Synthetic	$minRF$	0.04	[0.037 - 0.05], step-size: 0.01
	$minCS$	0.3	[0.5 - 1], step-size: 0.1
	$maxOR$	0.6	[0.05 - 0.45], step-size: 0.01
	No. of Machines (NM)	16	[4 - 24], step-size: 4
BMS-POS	$minRF$	0.04	[0.03 - 0.1], step-size: 0.01
	$minCS$	0.5	[0.5 - 1], step-size: 0.1
	$maxOR$	0.4	[0.05 - 0.45], step-size: 0.01
	No. of Machines (NM)	16	[4 - 24], step-size: 4
Mushroom	$minRF$	0.04	[0.04 - 0.2], step-size: 0.02
	$minCS$	0.4	[0.5 - 1], step-size: 0.1
	$maxOR$	0.35	[0.05 - 0.45], step-size: 0.01
	No. of Machines (NM)	16	[4 - 24], step-size: 4

(NM). Table 4.2 summarizes the parameters of our performance study. The performance metrics of our study include total execution time (ET) in seconds for extracting coverage patterns from the transactional dataset and Data Shuffled (DS), which represents the total amount of data shuffled between the machines during the execution. ET is the total time elapsed between program initiation at the presentation of the dataset as input and termination at the delivery of the final coverage patterns subject to the $maxOR$, $minRF$, and $minCS$ constraints. DS is then computed by calculating the total amount of data transmitted and received by the Master node at each level. Given N Worker nodes and K MB of total data to be broadcast by the Master, the total data shuffled for a broadcast to all Worker nodes is ($K \times N$) MB.

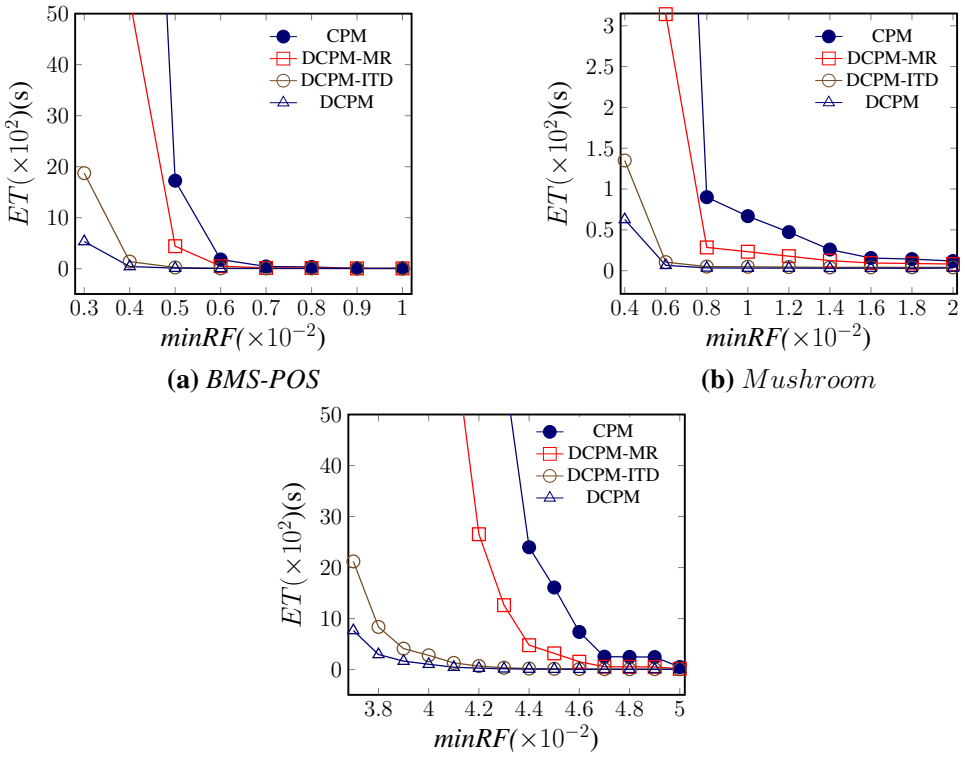
As a reference, for the purpose of meaningful comparison, we have implemented the following approaches:

- **Coverage Pattern Mining algorithm (*CPM*):** This is the *CPM* algorithm, implemented on a single node with 16 GB main memory in the ADA cluster.
- **Distributed *CPM* with MapReduce (*DCPM-MR*)**: We implemented the MapReduce-based *CPM* proposed in [47], in the following manner. First, we distribute the TDB among the Worker nodes and compute relative frequency values for all items in one phase of MapReduce. Further, we calculate one-size non-overlap patterns NOP_1 and broadcast the frequency values of all items to the Worker nodes. Second, we compute two-size non-overlap patterns NOP_2 and coverage patterns CP_2 using one phase of MapReduce. Third, at any k^{th} level, non-overlap patterns NOP_k

and coverage patterns, CP_k are computed using two phases of MapReduce. We implemented the approach using Apache Spark Framework.

- **Distributed *CPM* with ITD (*DCPM-ITD*):** We implemented the MapReduce based *CPM* proposed in [47] by introducing the notion of ITD and optimizing the second MapReduce phase. In this method, we compute the ITD of the given dataset at the start of execution and broadcast it to all Worker nodes. In *DCPM-ITD*, the first MapReduce phase for candidate pattern generation remains the same as the *DCPM-MR*. However, in the next step for computation of CS and OR of the generated candidate patterns, the generated candidate patterns are distributed equally among all the Worker nodes. Each Worker computes the CS and OR for all the candidate patterns it receives and sends them back to the Master. The Master node then computes the $NOPs$ and CPs with respect to the constraints. Note that the data is shuffled without employing any clustered approach. We implemented the approach using Apache Spark Framework.

- ***DCPM*:** We implemented the *DCPM* approach as discussed in Section 4.3. *DCPM* is implemented using Apache Spark architecture [70] and utilizes the default task scheduler of the framework.



(a) *BMS-POS* **(b)** $Mushroom$

(c) $Synthetic$

Figure 4.4: Effect of varying *minRF*

.

4.4.1 Effect of variations in minRF

Figure 4.4 depicts the results of $minRF$ versus ET for BMS-POS, Mushroom, and Synthetic datasets. The results show that execution time decreases with the increase in *minRF* for all approaches. This is due to the decrease in the number of size-one frequent itemsets (items satisfying $minRF$).

Compared to *CPM*, *DCPM-MR* and *DCPM-ITD*, the proposed *DCPM* algorithm takes less time to extract coverage patterns from the datasets. In comparing *DCPM* with *DCPM-ITD*, the execution time of *DCPM* is reduced because *DCPM* does not have the MapReduce step for explicit generation of all the possible candidate patterns. In comparing *DCPM* to *DCPM-MR*, the reduction in execution time is due to inverse transactional data modeling and clustered distribution of non-overlap patterns. Further, in comparing the *DCPM-ITD* and *DCPM-MR*, the reduction in execution time is due to inverse transactional data modeling. *CPM* significantly underperforms w.r.t. three approaches because it is a serial approach and executed on a single machine. Observe that *DCPM* is 37 times faster than *DCPM-MR* and 3.3 times faster for the BMS-POS dataset than *DCPM-ITD* when $minRF$ is 0.04.

The performance results in Figure 4.4(b) and Figure 4.4(c) exhibit similar trends for the Mushroom and Synthetic datasets respectively.

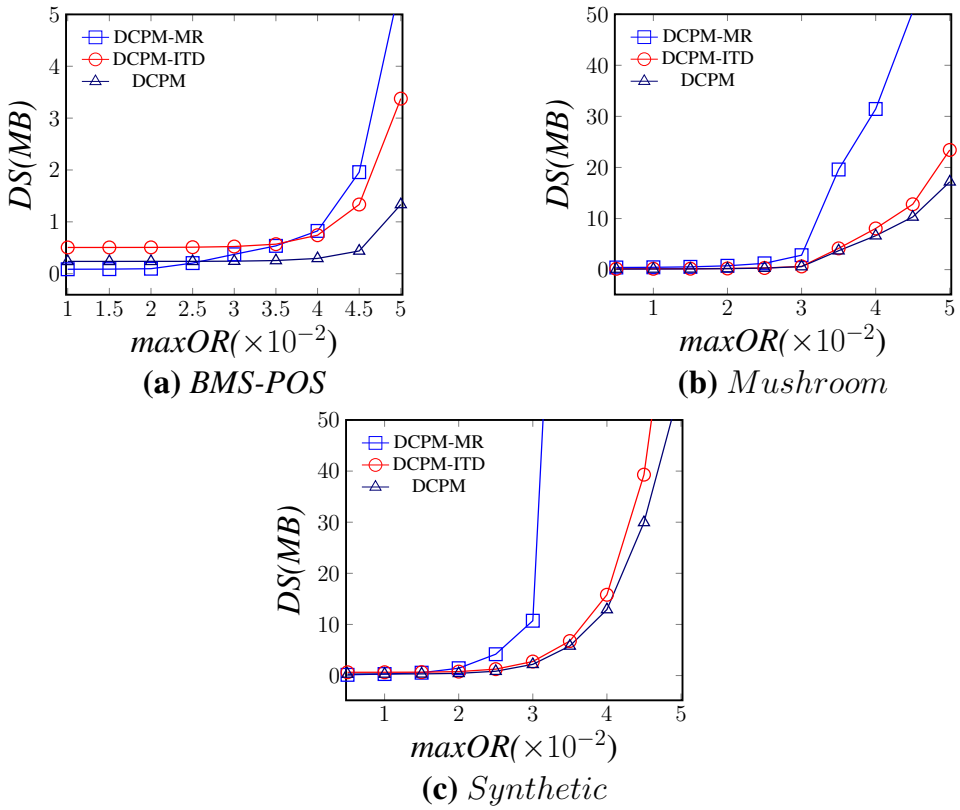


Figure 4.5: Effect of varying *maxOR* on data shuffled

.

4.4.2 Effect of varying *maxOR* on data shuffled

In this experiment, we show the results about the extent of data shuffled between the nodes. Figure 4.5 depicts the variation in data shuffled between nodes of varying $maxOR$ threshold for BMS-POS, Mushroom, and Synthetic datasets. The results in Figure 4.5(a) show that data shuffled increases with an increase in *maxOR* for all the approaches. This is because when *maxOR* increases, the number of non-overlap patterns generated increases. Hence, the amount of data shuffled between the Master and Worker nodes increases.

Overall, the results show that data shuffled under *DCPM* is the least. More data is shuffled in *DCPM-ITD* over *DCPM* because of the clustered representation of candidate patterns in *DCPM*. The amount of data shuffled is significantly less in *DCPM* over *DCPM-MR* because all $NOPs$ are distributed to all nodes under *DCPM-MR*, whereas in *DCPM*, only a partial list of candidate patterns are distributed to each node in a compact form. As *maxOR* increases, the number of candidate patterns at each level increases, thus leading to a lot of data shuffling in *DCPM-MR*.

4.4.3 Effect of variations in *maxOR*

Figure 4.6 depicts the results of $maxOR$ versus ET for BMS-POS, Mushroom, and Synthetic datasets. The results in Figure 4.6(a) show that execution time increases with an increase in *maxOR* for all the approaches. This is because as *maxOR* increases, the number of non-overlap patterns generated increases, which in turn increases ET. The results show that *DCPM* improves the performance significantly over *CPM*, *DCPM-MR* and *DCPM-ITD* approaches. In comparing the *DCPM* algorithm to *DCPM-ITD*, the execution time reduces because *DCPM* does not have the MapReduce step for explicit generation of all the possible candidate patterns. In comparing *DCPM* to *DCPM-MR*, the reduction in execution time is due to inverse transactional data modeling and clustered distribution of non-overlap patterns. Further, the lesser execution time of the former is due to the efficient computation of CS and OR values using ITD.

We observed similar trends in the results for the Mushroom and Synthetic datasets as depicted in Figure 4.6(b) and Figure 4.6(c) respectively.

4.4.4 Effect of variations in NM

Figure 4.7 depicts the result of NM (number of machines) versus ET for BMS-POS, Mushroom, and Synthetic datasets. The results in Figure 4.7(a) show that execution time decreases with an increase in *NM* and gradually reaches saturation value for all the approaches. For all the approaches, ET is the summation of both computation time and communication time. The ET of *DCPM* decreases with the increase in NM as the computation time decreases significantly due to parallel computation. It can be observed that ET is not significantly reduced for both *DCPM* and *DCPM-ITD* as compared to *DCPM-MR* with the increase in NM. This is due to less computation in both *DCPM* and *DCPM-ITD* as compared to *DCPM-MR*. We can observe similar trends in the results for the Mushroom and Synthetic

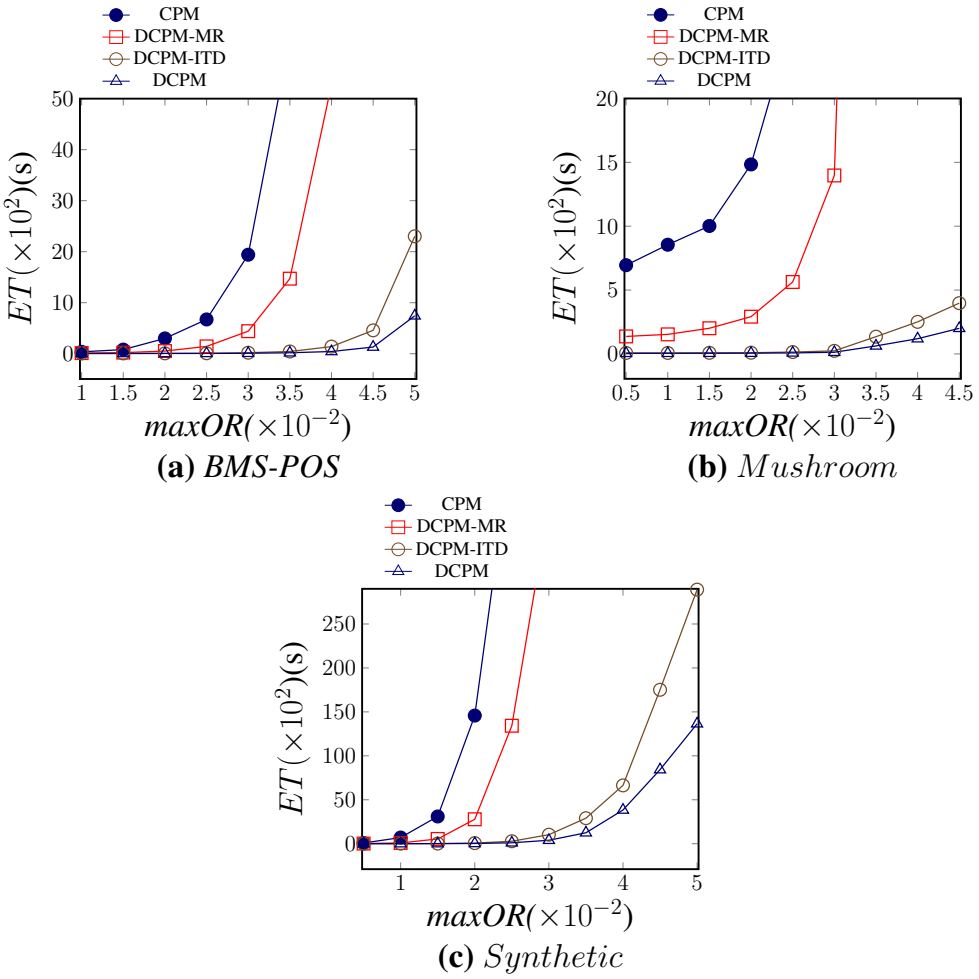


Figure 4.6: Effect of varying *maxOR*

.

datasets as depicted in Figure 4.7(b) and Figure 4.7(c). The values differ due to differences in dataset sizes and distribution of items in the transactions.

4.4.5 Effect of variations in *minCS*

Figure 4.8 depicts the results of $minCS$ versus ET for BMS-POS, Mushroom, and Synthetic datasets. The results in Figure 4.8(a) show that execution time remains almost constant with an increase in *minCS* for all the approaches. This is because as *minCS* increases, the total number of frequent items and non-overlap patterns generated remain unaffected because of fixed *maxOR* and *minCS*. Thus, the execution time ET does not show much variation for any value of *minCS* in all the approaches. It can be observed that the *DCPM* algorithm outperforms other approaches for all datasets.

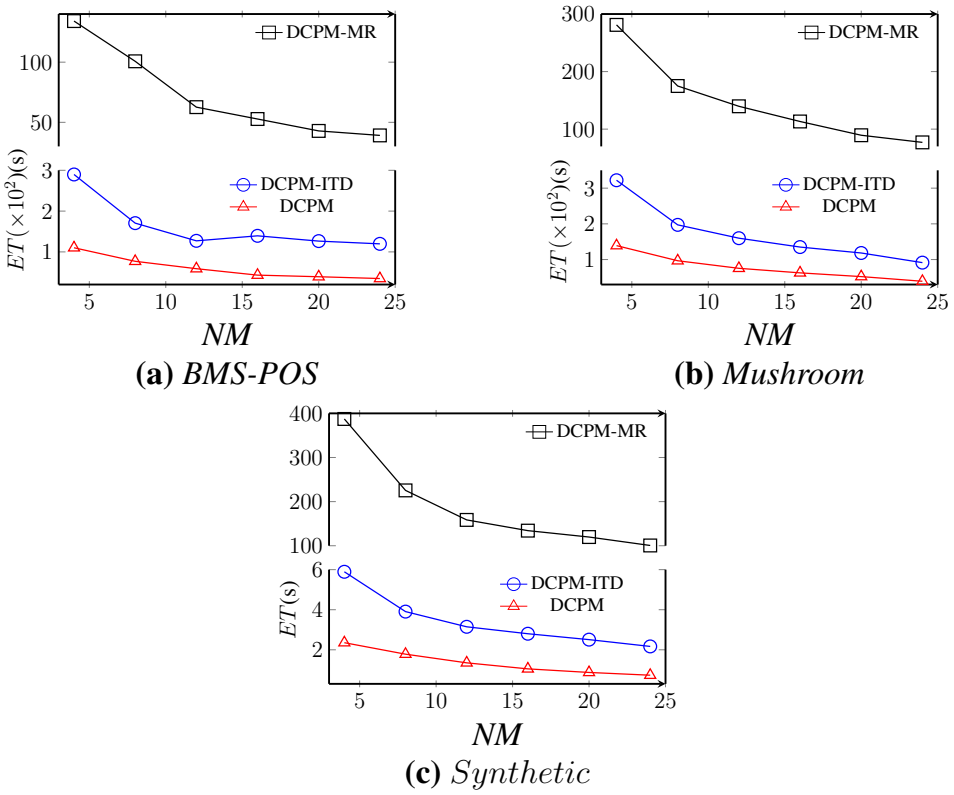


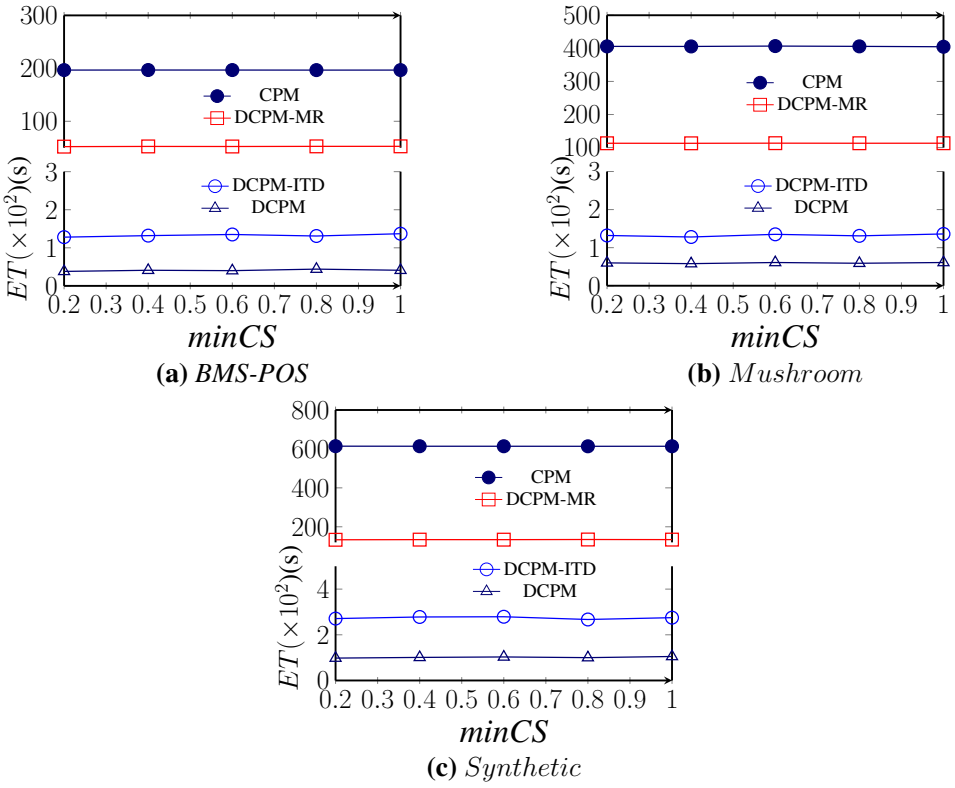
Figure 4.7: Effect of varying *NM*

.

4.5 Summary

In this chapter, we have proposed a Distributed Coverage Pattern Mining (*DCPM*) approach. In this approach, we model the transactional database into an Inverse Transactional Database (ITD) replicate it to each participant site, and transmit the information about candidate patterns in a summarized form by employing the notion of clustering. Our performance evaluation with two real-world datasets and one synthetic dataset demonstrates the effectiveness of the proposed approach in terms of execution time, and data shuffled across nodes.

In the next chapter, we propose an efficient ad-slot allocation in display advertising leveraging the proposed *DCPM* approach.



(a) *BMS-POS* **(b)** $Mushroom$ **(c)** $Synthetic$

Figure 4.8: Effect of varying *minCS*

.

Chapter 5

An improved ad-slot allocation framework for banner advertising

In this chapter, we present an efficient coverage pattern-based scalable framework for ad-slot allocation in display advertising. We first introduce the problem of ad-slot allocation in display advertising. We then discuss the proposed approach, algorithms, and an illustrative example to explain the workings of the framework in a detailed manner. Lastly, we talk about the experiments and show the enhancement in the performance compared to other approaches on various real-time click-stream data sets.

5.1 Introduction

In this section, we briefly introduce the research area and pose the context of the problem.

Display advertising consists of a set of visitors, a set of advertisers, and a publisher as the stakeholders. A visitor is a web page viewer visiting a set of web pages in a single session. The union of all web pages viewed by all the visitors forms the click-stream data set. Each advertiser comes with a demand in terms of the minimum number of visitors viewed one's advertisement. A publisher manages a set of web pages and corresponding ad slots contained in them. The publisher allocates a subset of ad slots to each advertiser based on one's demand and generates revenue in return from the advertisers.

Moreover, the publisher has to ensure a good browsing experience for the web page viewers by avoiding repeated advertisements. Therefore, in display advertising, efficient allocation of ad slots to the advertisers is an interesting research issue. In more formal terms, the research issues go as follows, given a click-stream data set and a set of advertisers along with their demand, the task of the publisher is to allocate a set of web pages to each advertiser such that the advertiser's demand is met and the publisher's revenue is maximized.

5.1.1 Context of the problem

Consider a *Publisher*, set W of web-pages $\{w_1, w_2, w_3, \ldots, w_t\}$, set Φ_s of ad-slots $\{s_1, s_2, s_3, \ldots, s_k\}$ available on all web-pages. Each ad-slot s_j is associated with set of views i.e., $< s_j{:}T^{s_j} >$, where T^{s_j} represent set of all visitors visited the ad-slot s_j. The map table M=$\{m_1, m_2, m_3, \ldots, m_t\}$, where

$m_i :< w_i, [ad - slots] >$ maps web-page w_i to corresponding ad-slots. Given a set $\mathbb{A}$ of advertisers $\{a_1, a_2, a_3, \ldots, a_h\}$, where each advertiser a_i has demand $I(a_i)$ in terms of impressions i.e., a_i:$< I(a_i) >$.

Given an advertiser a_i, a single ad slot may not meet the demanded impressions of a_i. Therefore, we allocate a set of ad-slots, which we refer to as *Ad-Slot Pattern* (ASP). Let ASP=$\{s_p, s_q, \ldots, s_r\}$, $ASP \subseteq \Phi_S$ be an ad-slot pattern, the set of all distinct views of ASP is the union of individual ad-slot views. The set of all distinct views of ASP ($D(ASP)$) is defined as follows:

$$D(ASP) = \bigcup_{j=p,q\ldots r} T^{s_j} \tag{5.1}$$

The number of impressions of an ASP ($I(ASP)$) is the collective distinct number of impressions of individual ad slots of ASP. Formally $I(ASP)$ defined as follow:

$$I(ASP) = \mathsf{D(ASP)} \tag{5.2}$$

The primary motivation for publishers is to allocate available ad slots to advertisers (under guaranteed contract norms) in order to maximize the revenue generated. Under guaranteed contract norms, we can allocate a ASP=$\{s_p, s_q, \ldots, s_r\}$ to an advertiser a_i if and only if the following condition is satisfied.

$$I(ASP) \geq I(a_i) \tag{5.3}$$

From the above equation, it is guaranteed that the total number of distinct views given to the advertiser a_i is greater than or equal to the number of impressions demanded by a_i.

For a given set Φ_s of ad-slots, advertiser a_i and demand $I(a_i)$, multiple ASPs can qualify the constraint shown in Equation 5.3. However, the ASP with the least number of repeated visits is considered to be most interesting because it ensures lesser wastage of views. This can improve the revenue of the publisher. In this regard, we introduce the notion of *overlap ratio* of ASP to measure the extent of overlap among the views corresponding to ASP. We now define the notion of the overlap ratio of ASP as follows.

Overlap of *ASP*: Let $ASP = \{s_p, s_q, \ldots, s_r\}$ be a ad-slot pattern and let $\{T^{s_p}$,T^{s_q},. . .,$T^{s_r}\}$ be the corresponding views of ad-slots belonging to ASP such that $\|T^{s_p}\| \geq \|T^{s_q}\| \geq \ldots \geq \|T^{s_r}\|$. The overlap ratio of ASP is the ratio of the number of distinct views common in $ASP - s_r$ and s_r to s_r. Formally defined as follows:

$$OR(ASP) = \frac{\mathsf{D(ASP - \{s_r\} \cap T^{s_r})}}{\mathsf{T}^{s_r}} \tag{5.4}$$

Here, $0 \leq OR(ASP) \leq 1$. An ASP is interesting when $OR(ASP)$ is less than a user-specified *maximum Overlap Ratio* (*maxOR*) threshold value.

Ad-slot allocation problem: Given a *publisher*, set W of web pages, set Φ_s of ad-slots available on all web pages, web-page to ad-slots map table M, *maxOR* and a set $\mathbb{A}$ of advertisers with corresponding demand in terms of impressions. The problem of the publisher is to assign a set of ad slots (ASP) to each advertiser such that $I(ASP) \geq I(a_i)$ and $OR(ASP) \leq maxOR, \forall a_i \in \mathbb{A}$.

5.2 Proposed Scheme

In this section, we first explain the basic idea and then present the proposed approach and further explain the approach in detail through an illustrative example

5.2.1 Basic Idea

Given a set of web pages with multiple ad-slots and a set of advertisers, our problem is to assign a set of ad-slots to each advertiser satisfying the advertisers' demand and *maxOR* constraint.

A naive approach to solve this problem would be random allocation of ad-slots to advertisers based on the number of user visits from the pool of available ad-slots until the advertiser's demand is met. However, this approach has the following issues. Due to random allocation, multiple ad slots that are visited by the same set of users can be allocated to a single advertiser. This leads to repeated (overlap) views on the advertisement. Therefore, this approach leads to a wastage of views, lowering the number of allocated advertisers and the revenue generated for the publisher.

In literature, an attempt has been to solve the above challenges in the work [31] using the knowledge of coverage patterns. However, this approach assumes that the number of ad slots per web page is limited to 1, which likely deviates from the practical scenario (multiple ad slots per each web-page are observed). Moreover, this approach uses CPPG algorithm [55] (a pattern growth-based non-parallel approach to extract the knowledge of coverage patterns), which is not scalable to large click-stream datasets because of the computational complexity and memory issues. Further, this work also involves every web page for coverage pattern mining (i.e., mine coverage patterns with minRF as 0), which is not computationally feasible for large real-time click-stream data sets. In our current work, we propose an ad-slot allocation approach by considering multiple ad-slots per web page model that addresses the shortcomings of the framework proposed in [31].

The basic idea is as follows. Given a click-stream data and *maxOR* value, we first extract all web-page patterns. Here, a web-page pattern is a set of web pages, which satisfy *maxOR* constraint. For this, we employ *Distributed Coverage Pattern Mining DCPM* algorithm [50] proposed in chapter 4, which performs well for large click-stream data sets. Next, given a map table (M), which maps the web page to its corresponding ad slots. We compute ad-slot patterns (*ASPs*) from web-page patterns by permuting with replacement over corresponding ad-slots on each web page. An ad-slot pattern with high overlap among the user visits may not be interesting because the user visits the same ad slot multiple times, and it may lead to a bad user experience. Therefore, we need to extract ad slots with high impressions and less

overlap. Next, for each ad-slot pattern, we compute the number of *impressions* (I). Here, *impressions* represents the coverage of the ad-slot pattern. We rank ad-slot patterns by sorting them in decreasing value of *impressions* and increasing order of *overlap ration*. The ranked ad-slot patterns constitute the *supply nodes set* w.r.t publisher. On the other hand, the advertisers make their demands in terms of the number of *impressions*. The set of advertisers with their demand constitutes the *demand node set*. Having modeled the *supply nodes set* and *demand node set*, we propose a greedy binary search-based ad-slots allocation approach for mapping (allocating) *supply nodes set* and *demand node set*.

Alternatively, in the above-mentioned approach, ad-slot patterns can also be generated directly from the click-stream data using *DCPM* algorithm by passing reconstituted click-stream data (replacing web pages with corresponding ad-slots) as input. For example, consider an arbitrary transaction T = $\{w_i$,w_j,.....,$w_k\}$ from the click-stream data consisting of set of web-pages. This transaction can be reconstituted into a set of ad slots visited in the user session by simply replacing each web page $w_i \in T$ with the corresponding ad slots. Let the web-page to ad-slot map table M = $\{w_i : \{a_{i_0},\ a_{i_1}..\}$, $w_j : \{a_{j_0},\ a_{j_1}..\}$, ... , $w_k : \{a_{k_0}, a_{k_1}..\}\}$. The reconstituted transaction $T^{'}$= $\{a_{i_0}$, $a_{i_1}..$, a_{j_0}, a_{j_1}.....,a_{k_0}, $a_{k_1}..\}$ is a set of ad-slots. Now this reconstituted click-stream data can be used directly to mine the ad-slot patterns (ASPs). However, this approach is computationally infeasible because of the combinatorial explosion of candidate ad-slot patterns (the number of candidate patterns formed by web pages ($2^W - 1$) is very much smaller than the number of candidate patterns formed by ad-slots ($2^{\Phi_s} - 1$), $\Phi_s \gg W$. But still, the value of $2^w - 1$ is large. In the proposed approach, to reduce the number of candidate patterns, we explored the opportunity that for any ad-slot pattern ASP consisting of an ad-slot from a particular web page w_i does not further join with any ad-slot from the same web page w_i because that would result in an overlap ratio of 1.0 for the resulting pattern. So in the level-wise DCPM algorithm, the redundant step of computing and checking the overlap ratio for such candidate patterns, where the overlap ratio is bound to be 1.0, can be avoided, thus reducing the number of candidate patterns at any level. Therefore, instead of generating ad-slot patterns directly, we should first generate web page patterns and then generate permutations with respect to the ad slots contained in the web pages to obtain the final ad-slot patterns ASPs. Finally, based on the advertiser's demand in terms of the number of impressions, we propose an ad-slot allocation approach to allocate ad-slots to advertisers to maximize the revenue of the publisher.

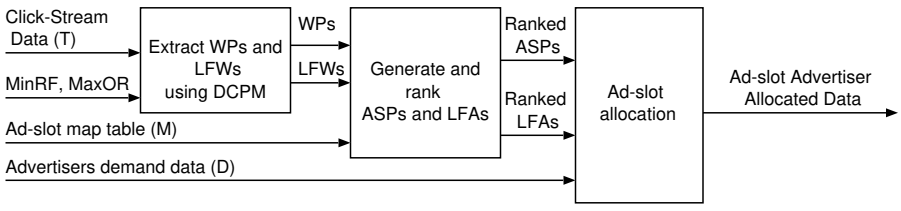


Figure 5.1: Block diagram of proposed framework

5.2.2 Proposed approach

In this section, we present an ad-slot allocation approach that helps publishers assign a set of ad-slots to each advertiser.

Given a *publisher*, set W of web-pages, set Φ_s of ad-slots, web-page to ad-slots map table M, *maxOR* and a set $\mathbb{A}$ of advertisers along with their demand in terms of impressions, the ad-slot allocation problem is to assign a set of ad-slots (ASP) to each advertiser such that $I(ASP) \geq I(a_i)$ and $OR(ASP) \leq maxOR, \forall a_i \in \mathbb{A}$. Figure 5.1 depicts the block diagram of the proposed approach. Algorithm 3 depicts the steps involved in the proposed method. The proposed ad-slot allocation framework consists of the following steps:

1. Extraction of non-overlap Web-page Patterns (WPs).
2. Generation of Ad-Slot Patterns (ASPs) and ranking.
3. Ad-slot pattern Allocation.

Algorithm 3 Ad-slot Allocation Framework

Input: $\mathbb{D}$: Clickstream data; $\mathbb{A}$: Set of advertisers, M: Ad-slot map table; *minRF*: minimum Relative Frequency; *maxOR*: maximum Overlap Ratio
Output: AA: Allocated Advertisers, NAA: Not Allocated Advertisers
Variables: ASPs: Ad-Slot Patterns, LFAs: Low Frequent Ad-slots.

1: ASPs, LFAs $\leftarrow \phi$
2: ASPs, LFAs $\leftarrow$ Extract_Ranked_ASPs($\mathbb{D}$, M, *minRF*, *maxOR*)
3: AA, NAA $\leftarrow$ Allocate_ASPs(ASPs, LFAs, $\mathbb{A}$)

5.2.2.1 Extraction of non-overlap Web-page Patterns

Given a transactional database *TDB*, which consists of t number of web pages as items, our task is to extract all non-overlap Web-page patterns from *TDB*. However, extraction of all possible WPs ($2^t - 1$) of given *TDB* is computationally expensive because the value of t is large. Instead, we can extract only WPs that are viewed by at least a threshold number of viewers, which we call as *minimum Relative Frequency* (*minRF*) threshold. Moreover, to maximize the revenue of the publisher, the overlap between the web page views in the web-page pattern should be less than a *maximum Overlap Ratio* (*maxOR*) threshold value. A WP that satisfies *minRF* and *maxOR* threshold value is called as *non-overlap Web-page pattern (WP)*.

Given a *TDB* and user-specified *minRF* and *maxOR* threshold values, our task is to extract all *non-overlap WPs*. In literature, the work in [50] proposed a Distributed Coverage Pattern Mining (DCPM) approach to extract coverage patterns from *TDB*. The DCPM approach is the state-of-the-art and computationally efficient approach to extract coverage patterns from *TDB*. In this approach, we employ the

DCPM approach (by setting coverage support to zero) to extract all *non-overlap WPs* from *TDB*. In addition, we store all web pages, which do not satisfy the *minRF* constraint. These web pages are referred to as frequent web pages (LFWs).

Algorithm 4 Extract_Ranked_ASPs($\mathbb{D}$, M, *minRF*, *maxOR*)

Input: $\mathbb{D}$: click stream Data; M: ad-slot Map table; *minRF*: minimum Relative Frequency; *maxOR*: maximum Overlap Ratio
Output: ASPs: Ad-Slot Patterns; LFAs: Low Frequency Ad-slots
Variables: Set $W :< P_i, OR(P_i) >$, LFWs: Low Frequency Web-pages

1: W, ASPs, LFWs $\leftarrow \phi$
2: W $\leftarrow$ Extract WPs from $\mathbb{D}$ using DCPM algorithm [50]
3: LFW $\leftarrow$ All web-pages which does not satisfy *minRF*
4: LFAs $\leftarrow$ Set of all ad-slots belonging to web pages in LFW
5: **for all** WP_i in W **do**
6: Generate all combinations of WP_i by replacing web-pages with corresponding ad-slots from map table $\mathbb{M}$
7: ASPs $\leftarrow$ ASPs $\cup$ combinations of WP_i
8: **end for**
9: **for all** ASP_i in $ASPs$ **do**
10: Compute $I(ASP_i)$ using Equation 5.2
11: Update ASP_i with $<ASP_i, I(ASP_i), OR(ASP_i)>$
12: **end for**
13: Sort patterns in ASPs in the descending order of $I(P_i)$ and increasing order of $OR(P_i)$.
14: return ASPs, LFAs

5.2.2.2 Generation of ad-slots patterns

Our final goal is to assign ad slots to advertisers based on the demand in terms of impressions. Having extracted all *non-overlap WPs* from *TDB*, we have to generate all *non-overlap WPs* into Ad-Slots patterns to assign ad-slots to advertisers. Each web page consists of one or more ad slots, and we maintain a map table consisting of mapping between ad slots and web pages. For each *non-overlap web-page pattern* P, we replace every web page $\in P$ with corresponding ad-slots as per the mapping table. Further, we perform permutations over each set of ad slots corresponding to each web page in P to form *Ad-Slot Patterns*. For each *ASP*, we compute impressions using the Equation 5.2. An *ASP* with a large number of impressions may not be interesting if there is a significant overlap among the user visits for a set of ad slots. The measure of overlap among the user visits is captured by the notion of overlap ratio (OR). In addition, we compute the overlap ratio for each *ASP* using Equation 5.4. Having computed $I(ASP)$ and $OR(ASP)$, we rank each *ASP* by sorting *ASPs* in decreasing the value of impressions and increasing the value of the overlap ratio.

To understand the permutations step clearly, let us look at an illustration for the same. Consider a non-overlap web-page pattern WP = $\{w_1, w_2, w_3\}$, and the corresponding Map Table for ad-slots

in web-pages w_1, w_2, w_3 be M = { w_1=$\{a_{11}, a_{12}\}$, w_2 = $\{a_{21}\}$, w_3= $\{a_{31}, a_{32}, a_{33}\}$ }. Then, the corresponding sets of permutations of non-overlap web-page pattern generated would be $\{\{a_{11}, a_{21}, a_{31}\}, \{a_{11}, a_{21}, a_{32}\}, \{a_{11}, a_{21}, a_{33}\}, \{a_{12}, a_{21}, a_{31}\}, \{a_{12}, a_{21}, a_{32}\}, \{a_{12}, a_{21}, a_{33}\}\}$.

/

5.2.2.3 Ad-slot allocation approach

In this section, we formulate advertiser's demand as demand nodes and extracted *ASPs* along with low frequent ad-slots as supply nodes. Now, the task of the ad-slot allocation problem is to allocate supply nodes to demand nodes such that the advertiser's demand is met and the publisher's revenue is maximized. The allocation of supply nodes to demand nodes is done by *allocate* and *delete* operations. We now define the ALLOCATE, DELETE operations, and *closest supply node* follows.

ALLOCATE: It is a task of assigning supply nodes to demand nodes such that the advertiser's demand is met and the publisher's revenue is maximized.

DELETE: It is a task of deleting all the supply nodes which contain an ad-slot/s in common with the allocated supply nodes. This operation is triggered immediately after each *allocate* operation is completed.

Closest supply node: A closest supply node to the demand node a_i is a supply node which minimises $I(a_i) - I(ASP)$.

The ad-slot allocation approach is as follows. Having modeled demand nodes, we first sort the demand nodes in decreasing order of their impressions. Next for each demand node starting with the highest demand, we perform ALLOCATE operation i.e, allocate the supply nodes to demand nodes such that the demand in terms of impressions is met. Next, we perform the *delete* operation. During the allocation process, the following four cases arises.

1. *Case-1:* For a given demand node, there exists at least one supply node from *ASPs* satisfying the demand. Then we allocate the closest supply node to the demand node and perform the DELETE operation.

2. *Case-2:* For a given demand node, there exists no supply node from *ASPs* that can satisfy the demand. Then we allocate the closest supply node to the demand node. In addition, we keep allocating low-frequency ad-slots (LFAs) to the demand node until the advertiser demand is met.

3. *Case-3:* For a given demand node, there exists no supply node from *ASPs* i.e, all the *ASPs* are exhausted. Then we keep allocating the low-frequency ad-slots until the advertiser demand is met.

4. *Case-4:* For a given demand node, there are no supply nodes i.e, neither *ASPs* nor sufficient low-frequency ad-slots are present to allocate. Then we mark such demand nodes as Not Allocated.

Algorithm 5 Allocate_ASPs(ASPs, LFAs, $\mathbb{A}$)

Input: Set of Ad-Slot Patterns ASPs: $\langle P_i, \text{Impressions}(P_i), \text{OR}(P_i)\rangle$; LFA: Low Frequency Ad-slots; $\mathbb{A} : \langle a_j, I(a_j)\rangle$ set of advertisers
Output: AAM: Allocated Advertisers Map $\langle a_j, AP_{a_j}\rangle$, NAA: Not Allocated Advertisers
Variables: AP_{a_j}: Pattern allocated to advertiser a_j, DASPs: Deleted ASPs, DLFAs: Deleted Low-Frequency Ad-slots

Sort advertisers in $\mathbb{A}$ in descending order of their impressions ($I(a_j)$)
for all a_j in $\mathbb{A}$ **do**
    $AP_{a_j} \leftarrow \phi$
    $AP_{a_j} = \text{argmin}_{P_i}$ abs$(I(P_i) - I(a_j)) \ni I(P_i) \geq I(a_j)$ and $P_i \in$ ASPs
    if AP_{a_j} is ϕ **then**
        $AP_{a_j} = \text{argmin}_{P_i}$ abs$(I(P_i) - I(a_j))$ where $P_i \in$ ASPs
        for all s_k in LFAs **do**
            if $I(AP_{a_j}) \geq I(a_j)$ **then**
                BREAK
            end if
            $AP_{a_j} = AP_{a_j} \cup \{s_k\}$
        end for
    end if
    if $I(AP_{a_j}) < I(a_j)$ **then**
        NAA = NAA $\cup\ a_j$
        CONTINUE
    end if
    AAM $\leftarrow$ AAM $\cup\ \langle a_j, AP_{a_j}\rangle$
    DELETE (AP_{a_j}, ASPs, LFAs)
end for
return AAM, NAA
procedure DELETE(AP_{a_j}, ASPs, LFAs)
    DASPs $\leftarrow \{P | \forall P \in \text{ASPs};\ P \cap AP_{a_j} \neq \phi\}$
    ASPs $\leftarrow$ ASPs - DASPs
    DLFAs $\leftarrow AP_{a_j} \cap$ LFA
    LFAs $\leftarrow$ LFAs - DLFAs
end procedure

The above procedure is repeated until either of the demand nodes or supply nodes are completely exhausted or there is no supply node remaining that can satisfy any of the demand nodes.

5.2.3 An Illustrative Example

We demonstrate the working of the Ad-slot Allocation Framework through an example. The sample click stream dataset $\mathbb{D}$ is depicted in Figure 5.2(A) and ad-slot map table M is depicted in Fig 5.2(B). Let the user-specified $minRF$ and $maxOR$ values be 0.3 and 0.6 respectively. The ad-slot allocation framework consists of 4 steps.

(A)

User Id	1	2	3	4	5	6	7	8
Session Data	w_1, w_2, w_4, w_5	w_1, w_3, w_6	w_2, w_3, w_7	w_2, w_5, w_6	w_1, w_4, w_7, w_5	w_1, w_2, w_3, w_4	w_1, w_2, w_4, w_5	w_1, w_3

(B)

Web-page	w_1	w_2	w_3	w_4	w_5	w_6	w_7
Set of Ad-slots	s_{11}	s_{21}, s_{22}	s_{31}	s_{41}	s_{51}, s_{52}	s_{61}, s_{62}	s_{71}, s_{72}

Figure 5.2: Input data (A) Click-stream data (B) Ad-slot map table

Step-1: Extraction of non-overlap web-page patterns: In this step, we extract the WPs and LFWs from the click-stream data using the DCPM algorithm corresponding to given $minRF$ and $maxOR$ threshold values. Figure 5.3(A), Figure 5.3(B) depicts the extracted WPs and LFWs respectively.

Step-2: Generation of ad-slots patterns: Fig 5.4 depicts the generation of ASPs and LFAs. Having extracted WPs and LFWs from the previous step, and given a Map table $\mathbb{M}$, the ad-slots patterns generation consists of the following steps.

1. We replace each web page in the WPs with the corresponding set of ad slots as shown in the Map Table M. For example we replace WP $\{w_1, w_2\}$ with $\{\{s_{11}\}, \{s_{21}, s_{22}\}\}$, because w_1 consists of ad-slot s_{11} and w_2 consists of ad-slots s_{21}, s_{22}.

2. Next, we form the ASPs by generating all possible combinations of ad slots in WP and compute impressions (I(ASP)) for each ASP using Equation 5.2.

(A) **maxOR=0.6 minRF=0.3**

WPs	w_1	w_2	w_3	w_4	w_5	w_1, w_2	w_2, w_3	w_3, w_5	w_3, w_4	w_4, w_5
OR	0.0	0.0	0.0	0.0	0.0	0.6	0.5	0.0	0.25	0.5

(B)

LFWs
w_6
w_7

Figure 5.3: (A) Web-page patterns (B) Low-frequency web-pages

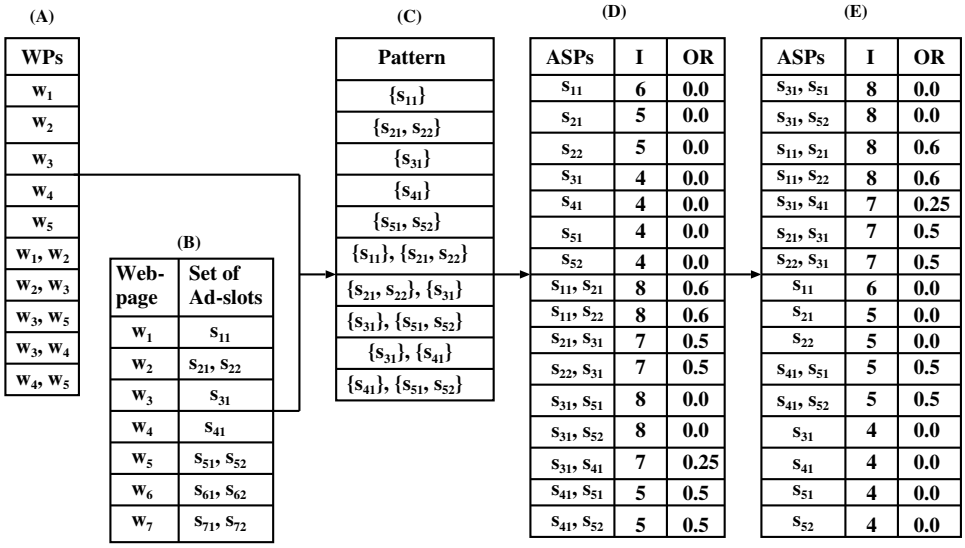


Figure 5.4: Generation of ASPs (A) Web-page patterns (B) Ad-slot map table (C) Ad-slots substituted Web-page patterns (D) Generated Ad-slot patterns (E) Sorted Ad-slot Patterns

3. Sort all the generated ASPs in decreasing order of Impressions (I(ASP)) and increasing order of Overlap Ratio (OR(ASP)).

4. Also, we list all the ad slots corresponding to LFWs and refer to them as Low Frequent Ad-slots (LFAs) as depicted in Figure 5.5.

Step-3: Ad-slot allocation: In this step, we allocate ad slots to advertisers based on demand. Fig 5.5 depicts the ad slots allocated to advertisers (AAM) based on the demand. Also, it depicts the not allocated advertisers (NAA). During the allocation process, 4 cases arise as mentioned in the proposed approach.

Case 1: Consider the advertiser $a_1(< a_1, I(a_1) : 8 >)$, The ASPs available for allocation are $\{ASP_1$-$ASP_{16}\}$ and LFAs available for allocation are $\{LFA_1$-$LFA_4\}$. Now the ASPs satisfying the demand of a_1 are ASP_1, ASP_2, ASP_3 and ASP_4, out of which ASP_1 and ASP_2 are better than ASP_3 and ASP_4 because of low OR. Thus, we allocate ASP_1 to advertiser a_1 (we can also allocate ASP_2 but we are just going with the lexicographical first pattern).

DELETE operation: After allocating ASP_1 to advertiser a_1, we delete all the ASPs containing ad slots in common to ASP_1. Therefore, the patterns ASP_1, ASP_2, ASP_5, ASP_6, ASP_7, ASP_{11}, ASP_{13} and ASP_{15} will be deleted from the ASPs because they contain either of the ad-slots s_{31} or s_{51}.

Case 2: Consider the advertiser $a_3(< a_3, I(a_3) : 6 >)$, after allocating the advertiser a_1 and a_2, the ASPs available for allocation are $\{ASP_9$, ASP_{10}, ASP_{12}, ASP_{13}, ASP_{14}, $ASP_{16}\}$ and LFAs available for allocation are $\{LFA_1$-$LFA_4\}$. There exists no available ASP that satisfies the advertiser a_3's demand of 6 impressions. Thus we allocate an ASP with impressions closest to the demand of a_3. The closest ASPs of a_3 are ASP_9, ASP_{10}, ASP_{12} having 5 impressions, out of which we allocate

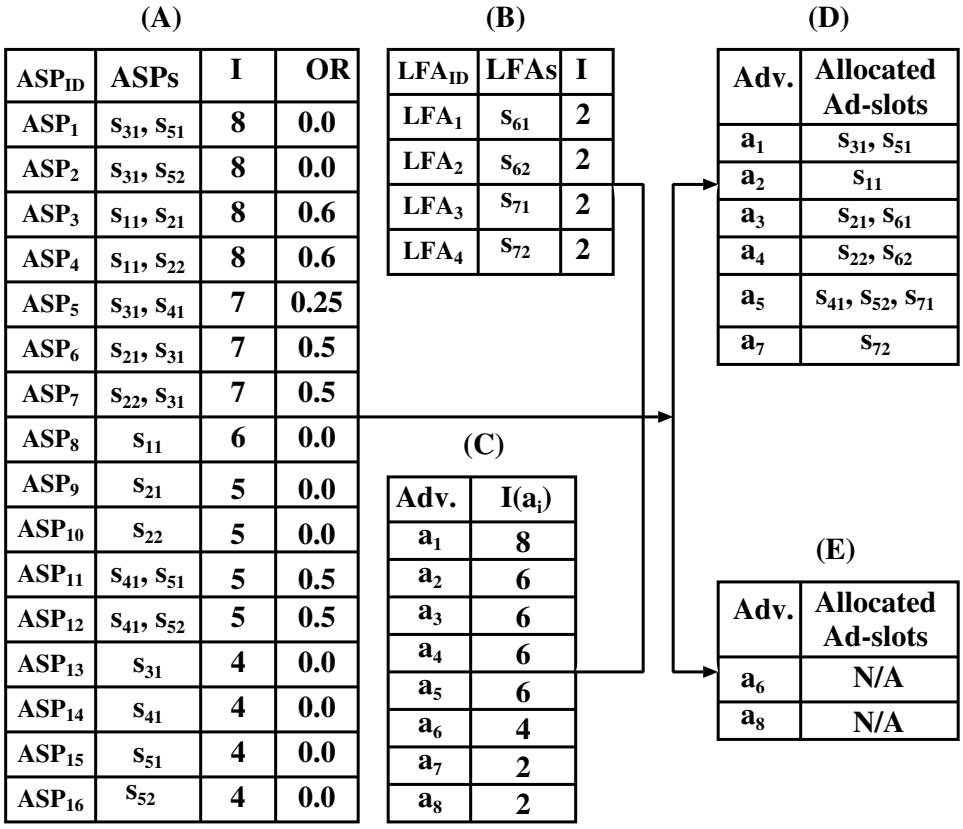


Figure 5.5: Ad-slot allocation method. (A) Sorted Ad-slot patterns (B) Low-frequency ad-slots (C) Advertisers data-set (D) Allocated Advertisers (E) Non-allocated Advertisers

the pattern ASP_9 to a_3. Next, we allocate the balance impressions (I(a_3) - I(ASP_9) i.e., 6-5=1) from LFA. We first allocate LFA_1 and calculate the total number of impressions of the allocated pattern $ASP_9 \cup LFA_1 = \{s_{21}, s_{61}\}$ using the Equation 5.2.

$$I(s_{21}, s_{61}) = |\mathbb{D}^{w_2} \cup \mathbb{D}^{w_6}| = |\{U_1, U_2, U_3, U_4, U_6, U_7\}| = 6$$

Since 6 impressions satisfy the advertiser's demand, we allocate $\{s_{21}, s_{61}\}$ to a_3.

DELETE operation: After allocating ASP_9 and LFA_1, we delete all the ASPs containing ad-slots in common to ASP_9. In this example, since there are no other ASPs with ad slots in common to ASP_9, we delete only ASP_9. Now, we also delete the LFA_1 allocated to a_3 from LFAs.

Case 3: Consider the advertiser $a_7(< a_7, I(a_7) : 2 >)$, after allocating the advertisers $a_1 - a_6$, there are no ASPs available for allocation and the only LFA available for allocation is $\{LFA_4\}$. Since there are no ASPs available, we allocate LFA_4 to satisfy a_7's demand of 2 impressions.

DELETE Operation: After allocating LFA_4, we delete it from LFAs.

Case 4: Consider the advertiser $a_6(< a_6, I(a_6) : 4 >)$, after allocating all the previous advertisers $a_1 - a_5$, there are no ASPs available for allocation, The only LFA available for allocation is $\{LFA_4\}$. Thus, there exists no ASP and LFA satisfying the demand of a_6, i.e., $I(a_6)$ = 4. Thus the advertiser a_6 is added to NAA (Not Allocated Advertiser)

Thus, we finally obtain the output AAM and NAA as shown in Fig 5.5.

5.3 Experiments

5.3.1 Experimental Setup

We have conducted our experiments in the ADA cluster [3] (at IIIT Hyderabad), which consists of 42 Boston SYS-7048GR-TR nodes equipped with dual Intel Xeon E5-2640 v4 processors, providing 40 virtual cores per node. The aggregate theoretical peak performance of ADA is 47.62 TFLOPS. Each virtual core was allocated with 2 GB of main memory.

The experiments were conducted on click-stream datasets. Clickstream data is the information collected about a user browsing a website during a given session. We have used two benchmark click-stream datasets, the Kosarak and the BMS-POS datasets.

The Kosarak dataset is provided by Ferenc Bodon, which contains (anonymized) click-stream data of a Hungarian online news portal [22]. Each user session is a set of web pages visited by the user, and it is modeled as a transaction. Therefore, click-stream data is a set of transactions, and each transaction is a set of items. The Kosarak dataset consists of 9,90,001 transactions and 41,270 distinct items. The BMS-POS dataset is a click-stream dataset [25] of an e-commerce company. This dataset has 5,15,596 transactions and 1,656 distinct items. Notably, the above-mentioned datasets do not contain any information about the number of ad slots contained on each web page. Since our task is to assign ad slots to advertisers, we have generated a Map Table $\mathbb{M}$, which maps ad slots to web pages in transactions. Moreover, we consider that each transaction in the datasets consists of (i) a fixed number of ad slots and (ii) a variable number of ad slots. In the case of a fixed number of ad slots, we assigned 3 ad slots to each web page in each transaction. In the case of the variable number of ad slots, we randomly assigned k ad slots ($1 \leq k \leq 3$) to each web page in each transaction. We have conducted experiments for both cases. The other input for our performance study is the advertiser's demand data $\mathbb{A}$. The demand of each advertiser (a_i) is represented in terms of impressions ($I(a_i)$), and it is generated uniformly at random between 1-2% of the total number of transactions in the respective dataset.

5.3.2 Performance Metrics

In this section, we explain the metrics that we have used for the performance evaluation.

5.3.2.1 Allocated Advertisers Count (AAC)

Represents the number of advertisers who are allocated with the set ad slots as per their respective demands in terms of impressions. A larger value of this metric signifies better utilization of ad space for revenue generation. Cost per impression (CPI) is the most commonly used revenue model [17]. In this scenario, major components include ad slots on web pages of the website, which supply a certain number of impressions, advertisers who demand impressions, and the publisher who allocates the ad slots that match the demands of advertisers through the server and generate revenue under the CPI revenue model.

5.3.2.2 Total Revenue Generated (TRG)

Represents the total revenue generated by the publisher for allocating the ad slots to the advertisers. The work in [31] proposed a Cost-Per-Impression (CPI) model, where the publisher earns units equal to the number of Impressions of an advertiser if one's demand is met. TRG is formally defined as follows:

$$TRG = \sum_{a_i \in AA} (I(a_i)) \tag{5.5}$$

Note that the higher the value of TRG better is the performance of the ad-slot allocation framework.

5.3.2.3 Average Repeated Advertisements(ARA)

ARA represents the average repetition of the number of unique advertisements in a user session. It is a metric that captures the amount of boredom created by the repetition of the advertisements in a given session. A lower value of this metric signifies lesser repetition of the advertisements to the visitors and thus a better user experience. The *ARA* is defined as follows:

$$ARA = \frac{\sum_{k=1}^{\mathsf{D}} \mathbf{card}(\bigcup_{w \in D_k}(SA(w)) - \bigcup_{w \in D_k} SA(w))}{\mathsf{D}} \tag{5.6}$$

Here, $\bigcup_{w \in T_k}(SA(w))$ represents the bag union (allows repeating items) of all the advertisements belonging to web-page $w \in D_k$ and $\bigcup_{w \in D_k} SA(w))$ represents the union (does not allows repeating items) of all the advertisements belongs to web-page $w \in D_k$. Therefore, the ARA captures the notion of overall advertisement repeatability across the click stream dataset D.

We have selected the above performance metrics in such a way as to reflect the real world in the closest possible way. In all our experiments, we have varied the number of advertisers (N_A) parameter and presented the results.

Algorithm 6 Visit Frequency-based(D, A, minRF, minCS, maxOR)

Input: D: Click Stream Data; A<a_i, d_i>: set of advertisers (a_i is id of advertiser; minRF: minimum relative frequency;

Output: AAM: Allocated Advertisers Map $< a_j, AP_{a_j} >$

1: Sort A in the descending order of $impressions(d_i)$
2: Create a set S of available ad-slots using click-stream data
3: **for all** a_i in A **do**
4: Impressions_allocated= 0
5: Ad-slots_allocated Demanded_impressions = t_i
6: **while** Demanded_impressions $>$ Impressions_allocated **do**
7: R = { x | Pick random element x : x $\in$ S }
8: AAM[a_i] = AAM[a_i] $\cup$ R
9: S $\leftarrow$ S - R
10: Impressions_allocated = | AAM[a_i]|
11: **end while**
12: **end for**
13: return AAM

5.3.3 Approaches implemented

In our performance evaluation, we compare our proposed approach with three other reference approaches. They are as follows:

1. Single-Add-slot Visit Frequency based method (SAVF).
2. Multi-Ad-slot Visit Frequency based method (MAVF).
3. Single-Ad-slot Pattern-based method (SAP).

We have implemented all three reference approaches. We now present the reference approaches.

5.3.3.1 Single-Ad-slot Visit Frequency based method (SAVF)

In this method, we first store all web pages along with several impressions. When an advertiser places one's demand in terms of the number of impressions, we randomly allocate web pages to the advertiser until his demand is met. Note that we will allocate one web page to one advertiser only. This

process is repeated until either all of the advertisers are allocated a set of web pages as per their demand or the available web page set is completely exhausted.

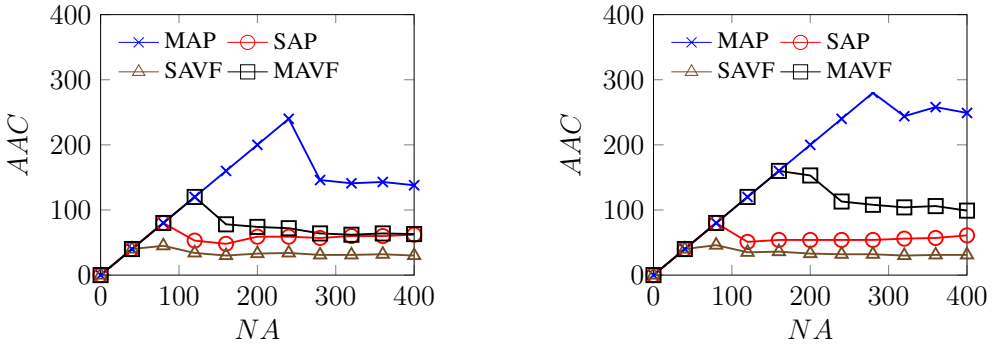
5.3.3.2 Multi-Ad-slot Visit Frequency based method (MAVF)

In this method, we first store all ad slots and the number of impressions. When an advertiser places one's demand in terms of the number of impressions, we start randomly allocating ad slots to the advertiser until his demand is met. Note that we will allocate one ad slot to one advertiser only. This process is repeated until either all of the advertisers are allocated a set of ad slots as per their demand or the available ad slots set is completely exhausted. Please refer to Algorithm 6 for pseudo-code.

5.3.3.3 Single-Ad-slot Pattern-based method (SAP)

Given click-stream data and advertiser demand constraints, this method considers that each web page contains a single ad slot. The approach is as follows. We first extract all web-page patterns (WP) from click-stream data subject to *minRF* and *maxOR* constraints using the DCPM algorithm. Second, we sort the WPs in decreasing order of their $I(WP)$, where $I(WP)$ represents the total number of impressions of the WP. Third, we start allocating the WPs to each advertiser until the demand is met. This method is quite similar to the proposed method, except that, *WPs* are used for allocation instead of *ASPs*.

Notably, in the proposed approach, we have proposed a framework to allocate ASPs to advertisers subject to *advertisers demand*, *minRF*, *maxOR* constraints in Multi-Ad-slot Pattern based scenario, which we henceforth refer it as *MAP*.

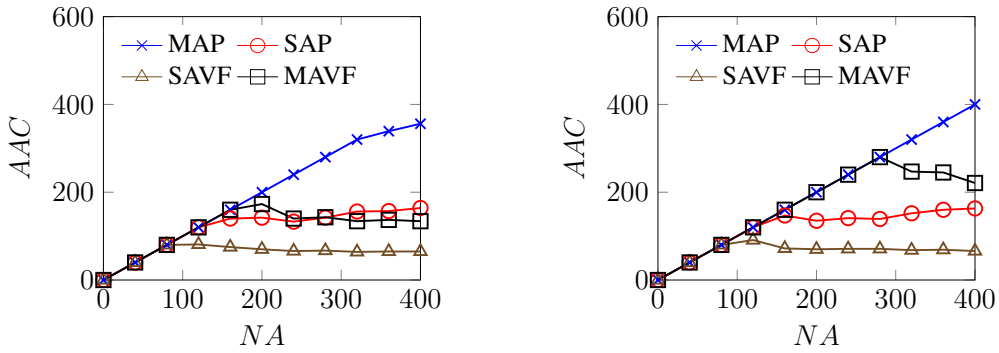


(a) Slots per web-page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.6: KOSARAK: Effect of varying NA on AAC

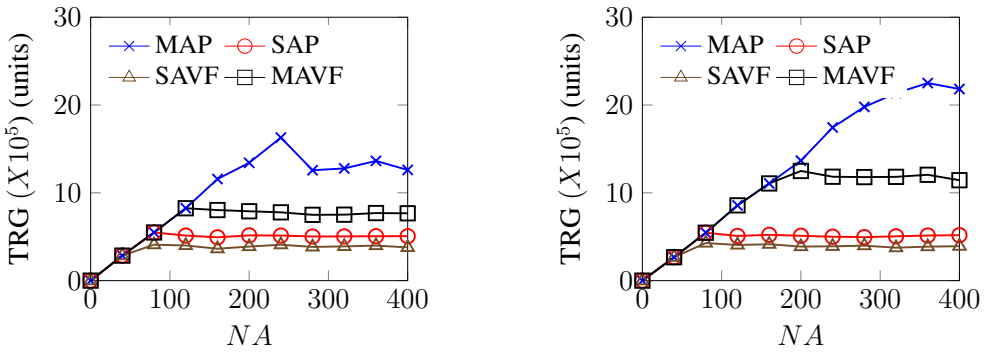
5.3.4 Results: Effect of variation of number of advertisers NA

Note that, in each of the figures shown. Figure (a) depicts the results for web pages with ad-slots in the range of 1 to 3 at random. Figure (b) depicts the results for web pages with only one ad slot.



(a) Slots per page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.7: BMSPOS: Effect of varying NA on AAC

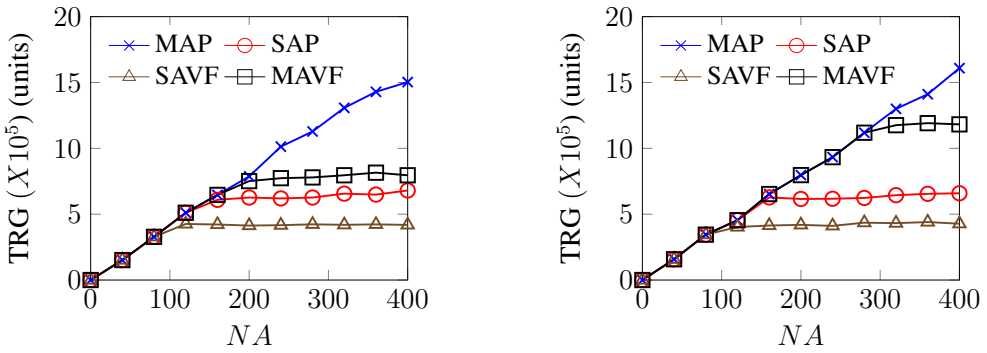


(a) Slots per page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.8: KOSARAK: Effect of varying NA on Revenue

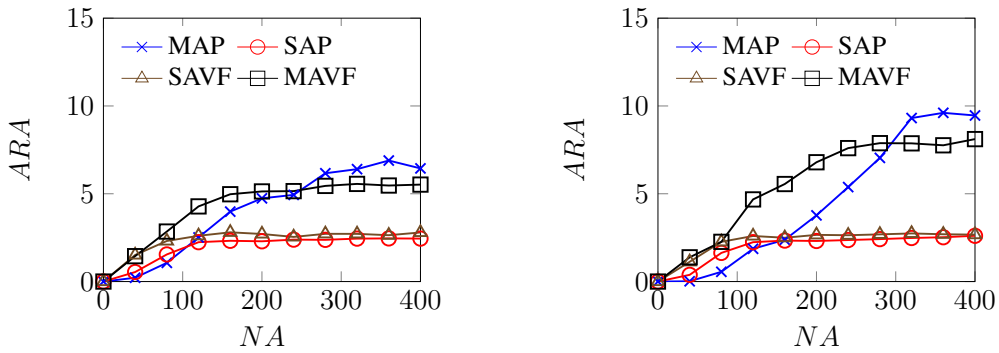
5.3.4.1 Effect on Allocated Advertiser Count (AAC)

Figure 5.6 depicts the variation in *AAC* with the increase in the number of advertisers for the KOSARAK dataset. The results show that *AAC* increases with an increase in *NA* to a certain level, decreases a little, and remains almost constant thereafter in all the approaches. The results show that when the number of advertisers is low, all the methods exhibit the same trend. For example, when *NA*=40, *ACC*=40 in all the approaches. This is because of the fact that when the total number of advertisers is low, the total number of distinct views/impressions needed to satisfy the demands of the given advertisers is also low. However, as the value of *NA* increases, the proposed *MAP* method outperforms all the other methods as depicted in the results. This is because ASPs used *MAP* allocation method leverage the notion of overlap, which ensures lesser repetition of user views and thus minimizes the wastage of impressions. The multi-ad-slot methods i.e., *MAVF* and *MAP* perform better than single-ad-slot methods *SAVF* and *SAP* because of the enhanced ad-space for allocation. Moreover, it can also be observed in Fig 5.6(a) that the saturation value of AAC in rand (1-3) is less than the saturation value of *AAC* in rand 3 as depicted in Figure 5.6(b). This is because of the increased number of ad slots available for allocation. Thus, the new proposed framework clearly shows the effectiveness of advertising space



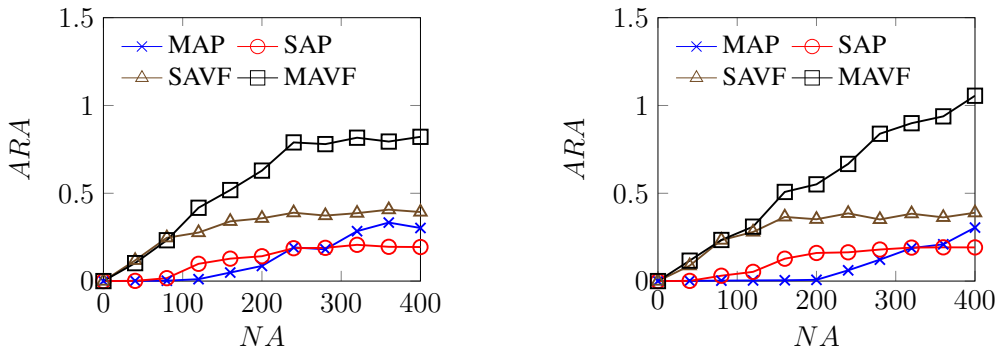
(a) Slots per page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.9: BMSPOS: Effect of varying NA on Revenue



(a) Slots per page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.10: KOSARAK: Effect of varying NA on ARP



(a) Slots per page: rand(1-3) **(b)** Slots per web-page: 3

Figure 5.11: BMSPOS: Effect of varying NA on ARP

management. The experiments on the BMSPOS dataset in Figure 5.7 also show a similar trend and the *MAP* method exhibits significant performance improvement over the rest of all methods.

5.3.4.2 Effect on Total Revenue Generated (TRG)

Figure 5.8 depicts the variation in *TRG* with the increase in the number of advertisers (*NA*) for the KOSARAK dataset. The results show that *TRG* increases steadily with an increase in *NA* to a certain level and reaches saturation in all the approaches. This is because, as the number of advertisers increases, all the approaches only allocate the ad slots to a certain extent. This is because the total number of ad slots available for the publisher to allocate is limited and would be exhausted at some point. However, observe that the proposed *MAP* method outperforms all the other methods as depicted in the results. This is because the *MAP* method leverages the notion of overlap in the allocation process, ensuring less repetition of user views. Moreover, *MAP* uses the low frequent ad slots, further improving ad-slot allocation. Moreover, it can also be observed in Fig 5.8(a) that the saturation value of AAC in rand (1-3) is less than the saturation value of *AAC* in rand 3 as depicted in Figure 5.8(b). This is because of the increase in number of ad slots available for the allocation process.

Figure 5.9 depicts the variation in *TRG* with the increase in the number of advertisers (*NA*) for the BMSPOS dataset. The results show a trend similar to KOSARAK dataset, but the values differ due to differences in dataset sizes.

5.3.4.3 Effect on Average Repeated Advertisements (ARA)

Figure 5.10 depicts the variation in *ARA* with the increase in the number of advertisers (*NA*) for the KOSARAK dataset. The results show that *ARA* increases steadily with an increase in *NA* to a certain level and reaches a saturation thereafter in all the approaches. This is because as the number of advertisers increases, the number of advertisement places increases, resulting in an increase in *ARA*. Observe that in the proposed method, the value of ARA is slightly low at a lower value of *NA* and slightly higher at a higher value of *NA*. However, compared to the proportionality increase in *AAC*, the proportionality increase in *AR* is significantly low compared to other methods.

Figure 5.11 depicts the variation in *ARA* with the increase in the number of advertisers (*NA*) for the BMSPOS dataset. The results show a trend similar to KOSARAK dataset but the values of *ARA* differ due to differences in dataset sizes. Observed that in the BMSPOS dataset, unlike the KOSARAK dataset, in the *MAP* method, the value of ARP is less than *SAVF* and *MAVF* methods for all values of *NA* even though *AAC* increases.

5.4 Summary

This chapter proposes an efficient ad-slot allocation framework in display advertising that considers multiple ad-slots per web page. Our performance evaluation with three real datasets demonstrates the effectiveness of the proposed framework in terms of revenue generated for the publisher and repetition of advertisements for the visitor. In the next chapter, we present the summary of the thesis.

Chapter 6

Summary and Conclusions

In this chapter, we present a summary followed by future work directions.

6.1 Summary

Banner advertising is the most common and widely employed online advertising method used to increase awareness of products and services. Ad-slot allocation in banner advertising is an interesting research issue. The existing works on ad allocation have employed diverse optimization techniques to establish theoretical solution bounds. These approaches have involved optimizing the objective function either by maximizing or minimizing it under modeled mathematical constraints. Others have used reinforcement learning and deep learning techniques to optimize the publisher's revenue generation. In [31], a coverage pattern-based ad-slot allocation framework was proposed but suffers from certain issues like scalability for large click-stream datasets and constraints of only one ad slot per web page. To address these issues, we made two major contributions to this thesis.

Firstly, to address the issue of scalability, we proposed an efficient coverage pattern mining technique, the Distributed Coverage Pattern Mining (*DCPM*) approach. *DCPM*, which is more scalable than all of the state-of-the-art approaches to coverage pattern mining. In this approach, we model the transactional database into an Inverse Transactional Database (ITD) replicate it to each participant site, and transmit the information about candidate patterns in a summarized form by employing the notion of clustering. We performed extensive experiments using two real-world datasets and one synthetic dataset and compared them with the state-of-the-art approach. The results demonstrate the effectiveness of our proposed approach in terms of execution time, and data shuffled across nodes.

Secondly, we have addressed the problem of ad-slot allocation with multiple ad-slots per web page. We have proposed an efficient ad-slot allocation framework for a given click stream dataset. In this approach, we first extract all the ad slot patterns (*ASPs*) from click-stream data using the *DCPM* approach. Next, for each ASP, we compute the impressions and sort them in descending of those impressions. The advertisers place their demand in the form of impressions, too. We then propose a greedy binary search-based ad-slots allocation approach for mapping (allocating) ASPs to advertisers to maximize the

publisher's revenue. In case of a shortage of impressions, we allocate the low-frequency ad slots (*LFAs*), which are not a part of any ASPs, to fulfill the shortage of impressions for the demand placed. Our performance evaluation using two real datasets demonstrates that our proposed framework significantly improves the publisher's revenue by efficiently allocating the available ad slots to advertisers compared to existing baseline methods.

6.2 Future work

We are interested in exploring the following ideas as part of future work.

1. In this thesis, we have focused on investigating the distributed implementation for level-wise coverage pattern mining approach. However, the pattern growth-based approach *CPPG* [55] for mining coverage patterns has the potential to further reduce the execution time of mining patterns through reduced search space. Hence, as part of future work, we want to explore the distributed implementation of *CPPG* algorithm.

2. In the ad-slot allocation framework proposed in this thesis, we have considered all the ad-slots to have identical dimensions. However, in the practical scenario, the size and positions of the ad slots can vary vastly. Hence, as part of our future work, we want to extend the ad allocation framework further by considering the dimension of variation of ad slots in terms of size and positioning on the web page.

3. In this thesis, the ad-slot allocation approach is proposed based on the assumption that the user-visit pattern at the time of ad-slot allocation remains the same as the pattern observed in the click-stream data provided as input to the ad-slot allocation framework. However, in the real-world scenario, the user visit pattern can vary every day because of factors like changes in the content of the web pages, the occurrence of a new event in a few web pages, etc. Hence, as part of future work, we want to investigate robust ad-space management frameworks that can handle the variations in user visit patterns.

Related Publications

1. **Preetham Sathineni**, A. Srinivas Reddy, P. Krishna Reddy, Anirban Mondal; An Efficient Distributed Coverage Pattern Mining Algorithm; Big data Analytics; Springer; 2021.

Under Pipeline

1. **Preetham Sathineni**, A. Srinivas Reddy, P. Krishna Reddy; Ad-slot Allocation for Banner Advertising using Non-overlap Patterns;

Bibliography

[1] How do consumer buzz and traffic in social media marketing predict the value of the firm? 30(2):213–238, 2013.

[2] *KDD '20: Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, New York, NY, USA, 2020. Association for Computing Machinery.

[3] ADA. `http://hpc.iiit.ac.in/wiki/index.php/Ada_User_Guide`, September Accessed in September 2021.

[4] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD*, volume 22, pages 207–216, 1993.

[5] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2496–2505, 2013.

[6] K. Amin, M. Kearns, P. Key, and A. Schwaighofer. Budget optimization for sponsored search: Censored learning in mdps, 2012.

[7] A. Budhiraja, A. Ralla, and P. K. Reddy. Coverage pattern based framework to improve search engine advertising. *International Journal of Data Science and Analytics*, pages 1–13, 2018.

[8] A. Budhiraja and P. K. Reddy. An approach to cover more advertisers in adwords. In *Proc. International Conference on Data Science and Advanced Analytics*, pages 1–10. IEEE, 2015.

[9] A. Budhiraja and P. K. Reddy. An improved approach for long tail advertising in sponsored search. In *Proc. Database Systems for Advanced Applications*, pages 169–184. Springer, 2017.

[10] H. Cai, K. Ren, W. Zhang, K. Malialis, J. Wang, Y. Yu, and D. Guo. Real-time bidding by reinforcement learning in display advertising. In *Proc. of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, page 661–670, New York, NY, USA, 2017. Association for Computing Machinery.

[11] C. Campbell and L. Marks. Good native advertising isn't a secret. *Business Horizons*, 58:599–606, 07 2015.

[12] M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A. R. Karlin, C. Mathieu, and M. Schwarz. Greedy bidding strategies for keyword auctions. In *Proc. of the 8th ACM Conference on Electronic Commerce*, EC '07, page 262–271, New York, NY, USA, 2007. Association for Computing Machinery.

[13] K. Chavan, P. Kulkarni, P. Ghodekar, and S. Patil. Frequent itemset mining for big data. In *International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1365–1368, 2015.

[14] W. Chen, T.-Y. Liu, and X. Yang. Reinforcement learning behaviors in sponsored search. *Applied Stochastic Models in Business and Industry*, 32:n/a–n/a, 02 2016.

[15] Y. Chen, P. Berkhin, B. Anderson, and N. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1307–1315, 08 2011.

[16] K.-W. Chon and M.-S. Kim. BIGMiner: a fast and scalable distributed frequent pattern miner for big data. *Cluster Computing*, 21(3):1507–1520, 2018.

[17] Y. Cui, R. Zhang, W. Li, and J. Mao. Bid landscape forecasting in online ad exchange marketplace. pages 265–273, 08 2011.

[18] D. Dua and C. Graff. UCI machine learning repository, Last accessed in September 2021.

[19] B. Edelman and M. Ostrovsky. Strategic bidder behavior in sponsored search auctions. *Decision Support Systems*, 43(1):192–198, 2007. Mobile Commerce: Strategies, Technologies, and Applications.

[20] U. Feige, N. Immorlica, V. Mirrokni, and H. Nazerzadeh. A combinatorial allocation mechanism with penalties for banner advertising. In *Proc. of the 17th international conference on World Wide Web*, pages 169–178, 2008.

[21] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.

[22] B. Ferenc. Fimi-frequent itemset mining implementations repository, 2003.

[23] L. Gangumalla, P. K. Reddy, and A. Mondal. Multi-location visibility query processing using portion-based transactional modeling and pattern mining. *Data Mining and Knowledge Discovery*, 33(5):1393–1416, 2019.

[24] A. Ghosh, P. McAfee, K. Papineni, and S. Vassilvitskii. Bidding for representative allocations for display advertising, 2009.

[25] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter*, 6(1):109–117, 2004.

[26] A. Goldfarb and C. Tucker. Online display advertising: Targeting and obtrusiveness. *Marketing Science*, 30(3):389–404, 2011.

[27] F. Gui, Y. Ma, F. Zhang, M. Liu, F. Li, W. Shen, and H. Bai. A distributed frequent itemset mining algorithm based on spark. In *International Conference on Computer Supported Cooperative Work in Design*, pages 271–275, 2015.

[28] J. Guo and Y.-g. Ren. Research on improved a priori algorithm based on coding and mapreduce. In *Web Information System and Application Conference*, pages 294–299, 2013.

[29] D. Hoffman and M. Fodor. Can you measure the roi of your social media marketing? *MIT Sloan Management Review*, 52:41–49, 10 2010.

[30] K. Kaushik, P. K. Reddy, A. Mondal, and A. Ralla. An incremental framework to extract coverage patterns for dynamic databases. *International Journal of Data Science and Analytics*, pages 1–19, 2021.

[31] V. N. S. Kavya and P. K. Reddy. Coverage patterns-based approach to allocate advertisement slots for display advertising. In *Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proc. 16*, pages 152–169. Springer, 2016.

[32] A. Kumar, R. Bezawada, R. Rishika, R. Janakiraman, and P. K. Kannan. From social to sale: The effects of firm generated content in social media on customer behavior. *Journal of Marketing*, 80, 10 2015.

[33] A. Lambrecht and C. Tucker. When does retargeting work? information specificity in online advertising. *Journal of Marketing Research*, 50(5):561–576, 2013.

[34] X. Li and D. Guan. Programmatic buying bidding strategies with win rate and winning price estimation in real time mobile advertising. pages 447–460, 05 2014.

[35] G. Liao, Z. Wang, X. Wu, X. Shi, C. Zhang, Y. Wang, X. Wang, and D. Wang. Cross dqn: Cross deep q network for ads allocation in feed. In *Proc. of the ACM Web Conference 2022*, WWW '22, page 401–409, New York, NY, USA, 2022. Association for Computing Machinery.

[36] J. C.-W. Lin, U. Ahmed, G. Srivastava, J. M.-T. Wu, T.-P. Hong, and Y. Djenouri. Linguistic frequent pattern mining using a compressed structure. 51(7):4806–4823.

[37] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proc. Special Interest Group on Knowledge Discovery and Data Mining*, pages 337–341. ACM, 1999.

[38] P. Maillé and B. Tuffin. Auctions for online ad space among advertisers sensitive to both views and clicks. *Electronic Commerce Research*, 18, 09 2018.

[39] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54, 10 2007.

[40] M. Morimoto and S. Chang. Consumers' attitudes toward unsolicited commercial e-mail and postal direct mail marketing methods. *Journal of Interactive Advertising*, 7:1–11, 09 2006.

[41] S. Najafi-Asadolahi and K. Fridgeirsdottir. Cost-per-click pricing for display advertising. *Manufacturing & Service Operations Management*, 16(4):482–497, 2014.

[42] Z. P. Ogihara, M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proc. KDD*, pages 283–286, 1997.

[43] T. Qin, W. Chen, and T.-Y. Liu. Sponsored search auctions. *ACM Transactions on Intelligent Systems and Technology*, 5:1–34, 01 2015.

[44] H. Qiu, R. Gu, C. Yuan, and Y. Huang. YAFIM: A parallel frequent itemset mining algorithm with spark. In *IEEE International Parallel Distributed Processing Symposium Workshops*, pages 1664–1671, 2014.

[45] H. Qiufeng, L. Qiang, H. Shiya, and C. Yingcong. Research on distributed parallel eclat optimization algorithm. In *Proc. International Conference on Artificial Intelligence and Big Data*, pages 149–154, 2020.

[46] A. Ralla, P. K. Reddy, and A. Mondal. An incremental technique for mining coverage patterns in large databases. In *Proc. International Conference on Data Science and Advanced Analytics*, pages 211–220. IEEE, 2019.

[47] A. Ralla, S. Siddiqie, P. K. Reddy, and A. Mondal. Coverage pattern mining based on MapReduce. In *Proc. ACM IKDD CoDS-COMAD*, page 209–213, 2020.

[48] S. Rathee and A. Kashyap. Adaptive-miner: an efficient distributed association rule mining algorithm on spark. *Journal of Big Data*, 5(1):1–17, 2018.

[49] O. J. Rutz and R. E. Bucklin. From generic to branded: A model of spillover in paid search advertising. *Journal of Marketing Research*, 48(1):87–102, 2011.

[50] P. Sathineni, A. S. Reddy, P. K. Reddy, and A. Mondal. An efficient distributed coverage pattern mining algorithm. In *Big Data Analytics: 9th International Conference, BDA 2021, Virtual Event, December 15-18, 2021, Proc. 9*, pages 322–340. Springer, 2021.

[51] K. K. Sethi and D. Ramesh. HFIM: a spark-based hybrid frequent itemset mining algorithm for big data processing. *The Journal of Supercomputing*, 73(8):3652–3668, 2017.

[52] P. Singh, S. Singh, P. K. Mishra, and R. Garg. RDD-Eclat: Approaches to parallelize eclat algorithm on spark rdd framework. In *Proc. International Conference on Computer Networks and Communication Technologies*, pages 755–768, 2020.

[53] P. G. Srinivas, P. K. Reddy, S. Bhargav, R. U. Kiran, and D. S. Kumar. Discovering coverage patterns for banner advertisement placement. In *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 133–144. Springer, 2012.

[54] P. G. Srinivas, P. K. Reddy, S. Bhargav, R. U. Kiran, and D. S. Kumar. Discovering coverage patterns for banner advertisement placement. In *Proc. PAKDD*, pages 133–144, 2012.

[55] P. G. Srinivas, P. K. Reddy, and A. V. Trinath. Cppg: efficient mining of coverage patterns using projected pattern growth technique. In *In PAKDD*, pages 319–329, 2013.

[56] P. G. Srinivas, P. K. Reddy, A. V. Trinath, S. Bhargav, and R. U. Kiran. Mining coverage patterns from transactional databases. *Journal of Intelligent Information Systems*, 45(3):423–439, 2015.

[57] B. Sripada, K. R. Polepalli, and U. K. Rage. Coverage patterns for efficient banner advertisement placement. In *Proc. World Wide Web*, pages 131–132. ACM, 2011.

[58] W. Tafesse. Content strategies and audience response on facebook brand pages. *Marketing Intelligence amp Planning*, 33:927–943, 09 2015.

[59] T. Teixeira, M. Wedel, and R. Pieters. Emotion-induced engagement in internet video advertisements. *Journal of Marketing Research*, 49(2):144–159, 2012.

[60] A. Trinath, P. G. Srinivas, and P. K. Reddy. Content specific coverage patterns for banner advertisement placement. In *Proc. International Conference on Data Science and Advanced Analytics*, pages 263–269. IEEE, 2014.

[61] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram. Optimal online assignment with forecasts. In *Proc. of the 11th ACM conference on Electronic commerce*, pages 109–118, 2010.

[62] S. Wang, M. Guo, J. Guo, and T. Tan. A hybrid genetic algorithm for ad-slot allocation in internet advertising. *IEEE Transactions on Evolutionary Computation*, 18(6):821–834, 2014.

[63] B. Wojdynski and N. Evans. Going native: Effects of disclosure position and language on the recognition and evaluation of online native advertising. *Journal of Advertising*, pages 1–12, 03 2016.

[64] D. Wu, X. Chen, X. Yang, H. Wang, Q. Tan, X. Zhang, J. Xu, and K. Gai. Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, oct 2018.

[65] J. M.-T. Wu, G. Srivastava, M. Wei, U. Yun, and J. C.-W. Lin. Fuzzy high-utility pattern mining in parallel and distributed hadoop framework. *Information Sciences*, 553:31–48, 2021.

[66] O. Yahya, O. Hegazy, and E. Ezat. An efficient implementation of apriori algorithm based on hadoop-mapreduce model. *International Journal of Reviews in Computing*, 12:59–67, 2012.

[67] S. Yang, G. Xu, Z. Wang, and F. Zhou. The parallel improved apriori algorithm research based on spark. In *International Conference on Frontier of Computer Science and Technology*, pages 354–359, 2015.

[68] J. Yuan, W. Xu, and R. Zhou. Multi-objective ad-slot allocation for display advertising. *Knowledge-Based Systems*, 171:36–48, 2019.

[69] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: Measurement and analysis, 2013.

[70] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache Spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[71] J. Zhang, B. Hao, B. Chen, C. Li, H. Chen, and J. Sun. Hierarchical reinforcement learning for course recommendation in moocs. In *Proc. of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

[72] W. Zhang, J. Qin, W. Guo, R. Tang, and X. He. Deep learning for click-through rate estimation. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4695–4703, 08 2021.

[73] X. Zhao, C. Gu, H. Zhang, X. Yang, X. Liu, J. Tang, and H. Liu. Dear: Deep reinforcement learning for online advertising impression in recommender systems. *Proc. of the AAAI Conference on Artificial Intelligence*, 35(1):750–758, May 2021.