OPTIMIZED BUDGET SELECTION FOR LOCAL DIFFERENTIAL PRIVACY

by
SEYEDPOUYA SEYEDKAZEMI

Submitted to the Graduate Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
July 2025

OPTIMIZED BUDGET SELECTION FOR LOCAL DIFFERENTIAL PRIVACY

APPROVED BY

Prof. Dr. YÜCEL SAYGIN .....
(Dissertation Supervisor)

Assoc. Prof. Dr. ÖZNUR TAŞTAN .....

Asst. Prof. Dr. TUĞÇE YÜKSEL .....

Prof. Dr. ALPTEKİN KÜPÇÜ .....

Assoc. Prof. Dr. ALİ İNAN .....

DATE OF APPROVAL: July 23, 2025

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my sincere appreciation to my advisors, Prof. Dr. Yücel Saygın and Asst. Prof. Dr. Mehmet Emre Gürsoy, for their steadfast guidance, support, and encouragement throughout my doctoral studies at Sabancı University. Their profound expertise and commitment to mentorship have been instrumental in shaping my academic progress and professional development. During challenging times, their insightful perspectives, critical questions, and constructive feedback consistently helped me improve and refine my research.

I would also like to express my sincere gratitude to Assoc. Prof. Dr. Öznur Taştan and Asst. Prof. Dr. Tuğçe Yüksel for their valuable insights and support as members of my dissertation committee. My appreciation further extends to Prof. Dr. Alptekin Küpçü and Assoc. Prof. Dr. Ali İnan, whose participation in my final dissertation defense and constructive feedback are greatly anticipated.

I am thankful for the financial support provided by The Scientific and Technological Research Council of Turkey (TÜBİTAK) through the 2244 Industrial PhD Program, under project number 119C045, which made this research possible.

Lastly, I owe my deepest thanks to my family—Mirjavad Seyedkazemi, Hamideh Farajzadeh, and Seyedpayam Seyedkazemi—for their unwavering love, patience, and belief in me. Their continuous encouragement has been the cornerstone of my perseverance, and this dissertation would not have been possible without them.

ABSTRACT

OPTIMIZED BUDGET SELECTION FOR LOCAL DIFFERENTIAL PRIVACY

SEYEDPOUYA SEYEDKAZEMI

Computer Science and Engineering
Ph.D Dissertation, July 2025

Dissertation Supervisor: Prof. Yücel Saygın
Dissertation Co-Supervisor: Assist. Prof. M. Emre Gürsoy

Keywords: Privacy, local differential privacy, optimization, budget selection, utility constraints, Capacity planning, Internet of Things, smart home.

In recent years, local differential privacy (LDP) has emerged as a popular privacy standard and received significant attention from academia and the industry. There is increasing interest in its deployment in industrial applications, smart homes, and smart cities. A pivotal problem in LDP is how to select the value of the privacy budget parameter ε, which controls the tradeoff between the strength of privacy protection and utility loss.

In this dissertation, we propose Optimus, a utility-driven and optimization-based approach for ε budget selection in LDP. Given a utility constraint, Optimus formulates the problem of budget selection as an optimization problem and contains five solution methods for finding the minimal ε that satisfies the given constraint. We experimentally evaluate the five solution methods in Optimus using five popular LDP protocols, two datasets, and varying utility constraints. We find that the ε budgets selected by the different methods are consistent, which is important considering that LDP is randomized and the optimization process is heuristic. Furthermore, we perform comparative analyses regarding the methods' efficiency and hyperparameters to assist in their future use. To the best of our knowledge, Optimus is the first work to provide a quantitative and algorithmic approach to solving the challenge of ε budget selection in LDP.

Beside Optimus which is a general approach for utility-driven budget selection in

LDP, we also consider the capacity planning problem specifically, as a practical application in smart homes. Since, the main premise of LDP is that data is perturbed to protect privacy, therefore consumption statistics estimated via LDP are inherently noisy. When noisy estimates are used for capacity planning, they can lead to false positives (false claims of capacity exceedance) or false negatives (actual exceedances are neglected).

To address these concerns, we propose a system called CAPRI for capacity planning and optimized budget selection in smart city applications under LDP. Based on a specified set of conditions (e.g., number of clients, possible consumption values, LDP protocol) and constraints (e.g., false positive probability should be below 0.01), CAPRI is able to determine the ε privacy budget which simultaneously satisfies the desired constraints and maximizes clients' privacy. To do so, CAPRI proposes an optimization-based problem formulation and a search-based solution which relies on LDP simulations. We experimentally validate and demonstrate the effectiveness of CAPRI using real-world and synthetic datasets, three popular LDP protocols, and various constraints and conditions.

ÖZET

YEREL FARKLILAŞTIRILMIŞ GIZLILIK IÇIN OPTIMIZE EDILMIŞ BÜTÇE SEÇIMI

SEYEDPOUYA SEYEDKAZEMI

Bilgisayar Bilimi ve Mühendisliği Doktora Tezi, Temuz 2025

Tez Danışmanı: Prof. Dr. Yücel Saygın
Tez Eş-Danışmanı: Assist. Prof. Dr. M. Emre Gürsoy

Anahtar Kelimeler: Gizlilik, yerel farklılaştırılmış gizlilik, optimizasyon, bütçe seçimi, fayda kısıtları, kapasite planlaması, Nesnelerin İnterneti, akıllı ev.

Son yıllarda, yerel farklılaştırılmış gizlilik (LDP), popüler bir gizlilik standardı olarak öne çıkmış ve hem akademi hem de endüstri tarafından önemli ölçüde ilgi görmüştür. Endüstriyel uygulamalar, akıllı evler ve akıllı şehirlerde LDP'nin kullanımı giderek artan bir ilgiyle karşılanmaktadır. LDP'deki temel sorunlardan biri, gizlilik korumasının gücü ile fayda kaybı arasındaki dengeyi kontrol eden gizlilik bütçesi parametresi ε'in nasıl seçileceğidir.

Bu tezde, Optimus adını verdiğimiz, fayda odaklı ve optimizasyon tabanlı bir ε bütçe seçimi yaklaşımı öneriyoruz. Verilen bir fayda kısıtı doğrultusunda, Optimus bütçe seçimi problemini bir optimizasyon problemi olarak formüle eder ve bu problemi çözmek için beş farklı çözüm yöntemi sunar. Optimus'un beş çözüm yöntemi; beş popüler LDP protokolü, iki farklı veri kümesi ve çeşitli fayda kısıtlarıyla birlikte deneysel olarak değerlendirilmiştir. Elde edilen sonuçlar, farklı yöntemler tarafından seçilen ε bütçelerinin birbiriyle tutarlı olduğunu göstermektedir ki, bu durum hem LDP'nin rastgelelik içermesi hem de optimizasyon sürecinin sezgisel olması bakımından önemlidir. Ayrıca, yöntemlerin verimliliği ve hiperparametreleri üzerine karşılaştırmalı analizler gerçekleştirerek, gelecekteki kullanım için rehberlik sunmaktayız. Bildiğimiz kadarıyla, Optimus, LDP'de ε bütçe seçimi sorununa nicel ve algoritmik bir çözüm sunan ilk çalışmadır.

Optimus, LDP altında genel bir fayda odaklı bütçe seçimi yaklaşımı sunarken, aynı

zamanda akıllı evlerde pratik bir uygulama olarak kapasite planlama problemini de özel olarak ele almaktayız. LDP'nin temel varsayımı, verilerin gizliliğini korumak amacıyla gürültülenmesidir; bu nedenle LDP ile tahmin edilen tüketim istatistikleri doğası gereği gürültülüdür. Gürültülü tahminlerin kapasite planlamasında kullanılması ise, yanlış pozitiflere (kapasite aşımı olduğu halde yanlış alarm verme) veya yanlış negatife (gerçek aşımın göz ardı edilmesi) yol açabilir.

Bu sorunları çözmek için, CAPRI adını verdiğimiz bir sistem öneriyoruz. Bu sistem, LDP altında akıllı şehir uygulamalarında kapasite planlaması ve optimize edilmiş bütçe seçimi sunar. Belirli koşullar (örneğin, müşteri sayısı, olası tüketim değerleri, kullanılan LDP protokolü) ve kısıtlar (örneğin, yanlış pozitif olasılığı 0.01'den küçük olmalıdır) doğrultusunda, CAPRI istenen gizlilik seviyesini en üst düzeye çıkarırken aynı zamanda bu kısıtları sağlayan ε gizlilik bütçesini belirleyebilmektedir. Bunu gerçekleştirmek için, CAPRI optimizasyon tabanlı bir problem formülasyonu ve LDP simülasyonlarına dayalı bir arama tabanlı çözüm sunmaktadır. CAPRI, gerçek ve sentetik veri kümeleri, üç popüler LDP protokolü ve çeşitli kısıt ve koşullarla yapılan deneylerle doğrulanmış ve etkinliği gösterilmiştir.

This thesis is dedicated to my family
For their endless love and support...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Local differential privacy (LDP) has emerged as an accepted standard for privacy-preserving data collection Cormode, Jha, Kulkarni, Li, Srivastava & Wang (2018); Wang, Blocki, Li & Jha (2017); Yang, Guo, Zhu, Tjuawinata, Zhao & Lam (2023). It has recently been applied in various contexts and data analysis tasks such as frequency estimation da Costa Filho & Machado (2023); Wang et al. (2017), heavy hitter identification Wang, Li & Jha (2021); Zhu, Cao, Xue, Wu & Zhang (2023), set-valued data analysis Huang, Xue, Zhu, Wei, Sun & Lu (2024); Wang, Li, Zhong, Chen, Wang, Zhou, Peng, Qian, Du & Yang (2023); Wang, Li & Jha (2018), geospatial data analysis Du, Hu, Zhang, Fang, Chen, Zheng & Gao (2023); Hong, Jung & Shim (2022); Tire & Gursoy (2024), and deep learning Arachchige, Bertok, Khalil, Liu, Camtepe & Atiquzzaman (2019); Truex, Liu, Chow, Gursoy & Wei (2020); Wang, Chen, Jiang & Zhao (2023). It has also been deployed in several industry products such as Google Chrome, Apple iOS, and Microsoft Windows aggregatable privacy-preserving ordinal response (2020); Ding, Kulkarni & Yekhanin (2017); Erlingsson, Pihur & Korolova (2014). In LDP, each client locally perturbs their sensitive data on their own device before sending the perturbed output to the data collector. Since privacy is ensured before the data leaves the client's device, LDP enables data collection and sharing in untrusted environments, e.g., when clients do not trust the data collector or the communication medium between the client and the data collector.

A fundamental problem in LDP is how to select the value of the privacy budget ε. The ε budget directly impacts the strength of privacy protection as well as utility loss – a lower ε provides stronger privacy but also increases utility loss. The problem of budget selection has been studied in centralized DP from the perspectives of economic models, utility, and disclosure and identification risks Chen, Yu, Tai, Li, Tsou, Huang & Lin (2017); Hsu, Gaboardi, Haeberlen, Khanna, Narayan, Pierce & Roth (2014); John, Denker, Laud, Martiny, Pankova & Pavlovic (2021); Kazan & Reiter (2023). There also exist works to offer explanations and illustrations of DP/LDP to end users so that end users may make more informed data sharing

decisions themselves Cummings, Kaptchuk & Redmiles (2021); Karegar, Alaqra & Fischer-Hübner (2022); Nanayakkara, Smart, Cummings, Kaptchuk & Redmiles (2023); Xiong, Wu, Wang, Proctor, Blocki, Li & Jha (2022). However, to the best of our knowledge, no technical solution or automated method exists for ε budget selection in LDP. The most relevant work in this space are LDPLens Gursoy, Liu, Chow, Truex & Wei (2022). However, LDPLens focuses on adversarial analysis of LDP protocols and ε, and it requires manual human effort to visualize the privacy-utility tradeoff.

On the other hand, smart homes and smart cities are two emerging paradigms that leverage popular Internet of Things (IoT), cyber-physical system, and artificial intelligence techniques to provide value-added services and improve residents' quality of life Zhang, Ni, Yang, Liang, Ren & Shen (2017); Zheng, Apthorpe, Chetty & Feamster (2018). In many smart city and smart home applications, multiple clients consume a shared resource with limited capacity, e.g., electricity, water, and drainage systems. In such circumstances, it is important to perform *capacity planning* carefully so that the shared resource is utilized effectively and its capacity is not exceeded. While capacity planning is typically performed by taking into account clients' consumption statistics, clients' consumption records are privacy-sensitive. For example, an adversary who knows the energy consumption records of a household can infer which home appliances are running Eibl & Engel (2014), predict whether a household is occupied or the residents are on holiday Eibl, Burkhart & Engel (2019); Kleiminger, Beckel & Santini (2015), and determine household characteristics and socio-economic status Anderson, Lin, Newing, Bahaj & James (2017); Beckel, Sadamori & Santini (2013); Czétány, Vámos, Horváth, Szalay, Mota-Babiloni, Deme-Bélafi & Csoknyai (2021). Thus, there is a compelling need to protect the privacy of clients' consumption records.

Due to the growing popularity of LDP, there is increasing interest in its deployment in industrial applications and smart homes Gai, Xue, Zhu, Yang, Liu & He (2022); Ou, Qin, Liao, Li & Zhang (2020). However, as mentioned before, the main premise of LDP is that the underlying data are perturbed or randomized to protect privacy. Therefore, when LDP is applied to consumption records, it causes consumption statistics to become noisy. When noisy statistics are used for capacity planning, they can lead to false positives (false claims of capacity exceedance) or false negatives (actual capacity exceedances are neglected). Considering these risks, the adoption of LDP can be hindered.

In this dissertation, we propose two software platforms: Optimus and CAPRI. The general one, **Optimus** is an optimization-based approach for ε budget selection in

LDP under utility constraints. The second, CAPRI, is a specific version of Optimus that is used for ε budget selection in LDP for capacity planning purposes. Consider a data collector who wishes to collect users' emoji usage or webpage visit statistics using LDP. The data collector would like to maximize clients' privacy by selecting a small ε, but has a maximum tolerable utility loss amount ω for downstream tasks. Optimus formulates the utility loss amount as a constraint and enables the selection of an optimized ε value under this constraint. More specifically, given a utility constraint in the form "ULoss should be $\leq \omega$ with probability $\geq \pi$", the goal of Optimus is to find the smallest ε that should be used so that clients' privacy is maximized while simultaneously the utility constraint is met. We formalize this as an optimization problem and develop five solution methods. Our solution methods rely on an intuitive and effective strategy: perform numerous offline LDP simulations with the given domain, protocol and ε within Optimus (no real client involvement), and determine whether the constraint is satisfied. If the current ε satisfies the constraint, then smaller ε values will be searched to find the smallest ε that satisfies the constraint. If not, then larger ε values will be searched to find an ε that satisfies the constraint.

On the other hand, it is computationally infeasible to search through all possible ε values. Therefore, Optimus utilizes customized search and optimization methods: Binary Search (BS), Relaxed Binary Search (RBS), Dynamic Binary Search (DBS), Multi-Pass (MPass), and Bayesian Optimization (Bayes). BS relies on the well-known binary search algorithm but occasionally suffers from eliminating the wrong half of the search space when the optimal ε is too similar to the current ε, due to randomness in LDP simulations. RBS addresses this issue by eliminating a smaller portion of the search space instead of half. DBS addresses the issue by performing fewer LDP simulations when the optimal ε and current ε are distant, but more simulations when they are similar. A higher number of simulations increases the correctness of search space elimination. MPass initially starts with a large search space and large *step* size, searches through the space in large steps, and identifies the region to further focus on. Each consecutive pass reduces the search space and *step* size by "zooming in" to the search space identified in the previous pass. Finally, Bayes uses Bayesian optimization Snoek, Larochelle & Adams (2012); Wang, Jin, Schmitt & Olhofer (2023) executed with a custom-designed objective function for ε budget selection. Our use of five different solution methods with diverse optimization algorithms enables the verification of whether the ε budgets selected by different methods are consistent (i.e., in agreement) with one another. Considering that LDP simulations are randomized and the optimization algorithms are heuristic, it is valuable to not rely on one algorithm alone, but verify the consistency of the

selected ε using multiple methods.

We propose another novel software system called CAPRI for informed capacity planning under LDP. In CAPRI, the planner specifies a set of conditions (e.g., number of clients, possible consumption values, LDP protocol) and a constraint (e.g., false positive probability should be below a threshold). CAPRI is then able to select the ε privacy budget that should be used in LDP so that the desired constraints will be satisfied under the given conditions. Here, ε is the key "privacy budget" parameter of LDP, which controls the privacy-utility tradeoff. Lower ε causes stronger privacy but higher estimation error (increased likelihood of false positives and false negatives). Higher ε causes weaker privacy but lower error. Therefore, it is desirable to select the lowest ε which satisfies the desired constraint.

CAPRI formulates the ε budget selection problem as an optimization problem. A novel solution is developed for solving this optimization problem, which iteratively searches for the optimized budget by narrowing the search space in each iteration. Numerous internal LDP simulations are performed in each iteration to find whether the desired constraint is satisfied. If the constraint is satisfied, then lower ε budgets will be searched; otherwise, higher ε will be searched.

In addition to the solution proposed in CAPRI, we also derive a theoretical upper bound using Chebyshev's inequality. Upon numerically comparing CAPRI's solution with the upper bound, we observe that the bound is usually not tight, e.g., false negative probabilities are substantially overestimated especially in low ε regimes. Thus, using CAPRI's solution rather than theoretical bounds, the planner can achieve a more precise selection of the ε budget, which improves false positive (or false negative) probability estimations and clients' privacy.

We experimentally evaluate the five solution methods in Optimus using five popular LDP protocols (GRR, RAPPOR, OUE, BLH, OLH), two datasets, and varying utility constraints. We find that the methods are generally consistent, i.e., in agreement with one another, since the differences between their selected ε budgets are small. BS can be relatively inconsistent due to the LDP randomness problem explained above. In terms of execution time, we find that MPass and Bayes take substantially longer compared to other methods, whereas DBS is the fastest. RBS generally gives a good tradeoff between efficiency and consistency. It has slightly higher execution times compared to BS and DBS, but not as high as MPass or Bayes. In addition, its ε budget selections are generally consistent with the other methods.

In addition, we experimentally show the effectiveness of CAPRI using a real-world water consumption dataset and synthetic datasets using various thresholds, capac-

ities, LDP protocols, and parameters. Under more stringent conditions, such as low FP thresholds or capacities close to true consumption, we observe that CAPRI selects a higher ε budget to reduce error. Under more relaxed conditions, lower ε is selected since the constraints are easier to meet, and therefore clients' privacy can be improved by lowering ε. These validate our expectations and CAPRI's behavior. Furthermore, the impacts of different LDP protocols (such as GRR, RAPPOR, OUE) and different numbers of simulations in CAPRI are experimentally demonstrated. Results show that higher number of simulations cause more reliable and stable convergence for CAPRI's optimization.

2. RELATED WORK

In this section, we present and discuss works related to Optimus under three categories: (i) works offering intuitive explanations of DP/LDP and the role of ε, (ii) works measuring the privacy implications of ε in LDP, and (iii) works assisting in ε selection in DP and LDP. In addition, we discuss works related to CAPRI in another four categories: (iv) DP and LDP in smart home and smart city applications, (v) High-dimensional crowdsourced data publication under LDP, (vi) Data stream collection under LDP, and (vii) Communicating DP/LDP and the impact of ε.

2.1 Explaining DP/LDP and The Role of ε to End Users

Several recent studies have explored the problem of explaining centralized differential privacy (DP), local DP (LDP), and the impact of ε to end users. Cummings et al. Cummings et al. (2021) investigated users' expectations from DP and their willingness to share data using DP systems. They proposed a framework to take into account the interplay between DP explanations and a user's privacy concerns. Xiong et al. Xiong, Wang, Li & Jha (2020) performed four online human subject experiments investigating the effects of using different approaches to communicate DP techniques to end users in a health app setting. They found that when users are shown descriptions that explain the implications instead of the definitions of DP/LDP, users had higher comprehension and willingness to share information with LDP. In Xiong et al. (2022), Xiong et al. developed illustrations for communicating the impact of ε in DP, LDP, and Shuffle DP using location data and heatmaps. Bullek et al. Bullek, Garboski, Mir & Peck (2017) measured users' comfort, understanding, and trust in a randomized response-based privacy protection mechanism. They found that allowing users to see the perturbation amount increased users' trust in the mechanism. Karegar et al. Karegar et al. (2022) studied the use of metaphors

to explain the privacy protection of DP and LDP. Franzen et al. Franzen, Nuñez von Voigt, Sörries, Tschorsch & Müller-Birn (2022) proposed the adoption of risk communication formats from the medical domain to communicate privacy guarantees to end users. Nanayakkara et al. Nanayakkara et al. (2023) developed and evaluated three methods that convey the probabilistic guarantees of DP and ε via risk communication and usability.

While these works are beneficial in explaining DP, LDP, and the impacts of ε to end users, they differ from our work in three fundamental ways. First, they are mostly focused on explaining the *privacy* impacts of DP/LDP and ε, rather than *utility*. Second, several of them aim to understand and influence the users' trust in the privacy protection mechanism or change users' data sharing decisions. In contrast, we do not consider the scenario where users may opt out of data sharing. Third, these works do not have the goal of optimizing the selection of the ε budget using a utility constraint, which is the focus of our work.

2.2 Measuring the Privacy Implications of ε

There are several works in LDP which propose technical methods for measuring the privacy implications of ε. Murakami et al. Murakami & Takahashi (2020) evaluated re-identification risks in LDP using obfuscated data collected from users. Gadotti et al. Gadotti, Houssiau, Annamalai & de Montjoye (2022) proposed pool inference attacks in which the adversary defines pools of interest and then aims to determine the user's pool. The proposed attack is evaluated on the Count Mean Sketch (CMS) mechanism. Gursoy et al. Gursoy et al. (2022) studied LDP protocols from the perspective of a Bayesian adversary, and measured the adversary's inference ability under varying protocols and ε. Arcolezi et al. Arcolezi, Gambs, Couchot & Palamidessi (2023) studied the risks of multi-dimensional data collection under LDP. Finally, Gursoy et al. Gursoy (2024) studied longitudinal attacks against LDP protocols, i.e., attacks stemming from multiple subsequent data collections. While these works offer mathematical and experimental evaluations of LDP protocols and ε budgets, they all focus on the *privacy* aspect, whereas Optimus focuses on the utility aspect.

2.3 Assisting in ε Selection

Under centralized DP, Lee and Clifton Lee & Clifton (2011) recommended to determine the value of ε by linking it with the risk of identifying an individual. Hsu et al. Hsu et al. (2014) proposed an economical method for choosing ε by formulating a benefit (e.g., monetary compensation) versus risk (e.g., sensitive information leakage or exposure) approach. Chen et al. Chen et al. (2017) proposed a data-driven framework for determining ε by jointly considering risk of disclosure and utility. John et al. John et al. (2021) developed an interactive tool called Differential Privacy Policy (DPP) for visualizing the impacts of ε and helping a data owner decide how much noise is sufficient before statistical data is shared. Kazan and Reiter Kazan & Reiter (2023) proposed a framework for selecting ε based on its relationship with the Bayesian posterior probability of disclosure. All of these works are under the *centralized* form of DP, therefore they are not suitable for LDP.

Most relevant to our work are LDPLens Gursoy et al. (2022) and CAPRI Seyedkazemi, Gursoy & Saygin (2024) which are designed for LDP. LDPLens can assist in protocol and budget selection in LDP by considering the privacy-utility tradeoff. However, LDPLens focuses on the adversarial aspect rather than the utility aspect, and it requires manual human effort to visualize the privacy-utility tradeoff and analyze the results. On the other hand, CAPRI is focused on capacity planning in industrial environments, with metrics and algorithms tailored to this specific purpose. Hence, neither LDPLens nor CAPRI provides a general solution to the problem considered in this paper, which is to enable automated and optimized ε selection under utility constraints.

In this section, we summarize related works under four categories and describe the main differences between our paper and those works that fall under each category.

2.4 DP and LDP in smart home and smart city applications.

Over the last decade, differential privacy (DP) has become a popular standard for privacy protection. While DP has seen adoption in the IoT and smart grid domains, most of this adoption has used the *centralized* form of DP Marks, Montano, Chong,

Raavi, Islam, Cerny & Shin (2021). In Razavi, Arefi, Smith, Ledwich, Nourbakhsh & Minakshi (2022), a DP framework was constructed for energy forecasting using Bayesian neural networks. In Gough, Santos, AlSkaif, Javadi, Castro & Catalão (2021), the impact of DP on smart meter data was examined from the perspectives of electricity bills and power system providers. More recently, local differential privacy (LDP) has emerged and received significant attention from the industry and academia Cormode et al. (2018); Ding et al. (2017); Erlingsson et al. (2014); Wang et al. (2017); Wang, Zhao, Hu, Yang, Ren & Lam (2021). Several LDP protocols were developed and analyzed under varying conditions Cormode, Maddock & Maple (2021); Gursoy et al. (2022). In this paper, we use the more recent LDP notion rather than DP.

Regarding the use of LDP in smart home and smart city applications, a singular spectrum analysis-based LDP method has been proposed in Ou et al. (2020) to prevent inference of household appliances. In Gai et al. (2022), a data aggregation scheme with LDP was developed based on randomized response. In Islam, Badsha, Sengupta, Khalil & Atiquzzaman (2022), utility-optimized LDP (a variant of LDP) was used in designing a collaborative network intrusion detection system for electric vehicle charging stations. However, these works have not studied the capacity planning and budget selection problems that are the main focus of this paper.

2.5 High-dimensional crowdsourced data publication under LDP.

Several recent works have studied the problem of high-dimensional crowdsourced data publication (data release) under LDP. Ren et al. Ren, Yu, Yu, Yang, Yang, McCann & Philip (2018) proposed LoPub for publishing high-dimensional data under LDP, which utilizes Expectation-Maximization and Lasso regression for improving the utility of multi-dimensional joint distribution estimation. Copula functions were used in Wang, Yang, Ren, Yu & Yang (2019) and a junction tree-based approach was used in Liu, Tang, Hu, Jin, Guo, Stoyanovich, Teubner, Mamoulis, Pitoura & Mühlig (2023) for improving utility in high-dimensional data publication. A related problem concerning high-dimensional data is the release of marginals or marginal distributions under LDP, which was studied in Zhang, Wang, Li, He & Chen (2018) and Cormode, Kulkarni & Srivastava (2018). The problem of mean estimation from multi-dimensional data was studied in Wang, Xiao, Yang, Zhao, Hui, Shin, Shin & Yu (2019) and frequency estimation was studied in Arcolezi, Couchot, Al Bouna &

Xiao (2021) and da Costa Filho & Machado (2023). In Duan, Ye & Hu (2022) and Duan, Ye, Hu & Sun (2024), enhancements of LDP mechanisms were proposed for improving utility in high-dimensional space.

There are several ways in which our work differs from these works. First, we treat clients' consumption values as singular values rather than high-dimensional records with multiple attributes. Second, we do not study the problem of data or marginal publication. Thus, we do not face the challenges of high-dimensionality or multi-dimensional data publication. In addition, existing works do not study capacity planning or ε budget selection, which are the focus of our paper.

2.6 Data stream collection under LDP.

LDP can also be used in scenarios where clients' data is not static but streaming. Wang et al. Wang, Chen, Zhang, Su, Cheng, Li, Li & Jha (Wang et al.) studied the continuous release of data streams under centralized and local DP. In Bao, Yang, Xiao & Ding (2021), the Correlated Gaussian Mechanism (CGM) was proposed for injecting temporally correlated noise into streaming data collection to achieve LDP. In Ren, Shi, Yu, Yang, Zhao & Xu (2022), w-event LDP was proposed for infinite data streams, which aims to guarantee LDP for w consecutive timestamps. In Gao & Zhou (2023), w-event LDP was combined with sampling and post-processing using Kalman filters to enhance utility. Fuzzy counting from data streams under LDP was studied in Vatsalan, Bhaskar & Kaafar (2022), and collecting correlation-constrained sensor streams was studied in Fu, Ye, Du & Hu (2023). Finally, Li et al. Li, Liu, Lou, Hong, Zhang, Qin & Ren (2024) studied the problem of top-k frequent item estimation from data streams under LDP and bounded memory. While the capacity planning and ε budget selection problems in CAPRI are different from these works, they can be used to extend CAPRI to streaming scenarios. In its current version, CAPRI takes into account the current consumption of a client rather than modeling the client's consumption as a data stream. In the future, client consumptions can be modeled as data streams, and methods such as w-event LDP can be adapted for continuous budget selection and capacity planning.

2.7 Communicating DP/LDP and the impact of ε.

Few recent works studied the problem of communicating DP/LDP notions and the impact of ε budget to end users. In Xiong et al. (2020), four online human subject experiments were performed to investigate the effects of different approaches to communicate DP techniques. Results showed that end users prefer LDP to DP when sharing their sensitive information, understanding that LDP offers stronger privacy than DP. In Xiong et al. (2022), explanatory illustrations were used to help end users understand how noise protects their sensitive information in DP, LDP, and Shuffler DP models. In Cummings et al. (2021), users' expectations from DP were investigated. In Nanayakkara et al. (2023), a survey was conducted to quantitatively evaluate different approaches to conveying DP guarantees to end users. While all of these works are useful towards explaining the privacy guarantees of DP/LDP and the impacts of ε to end users, they do not provide a quantitative, utility-driven method for selecting the ε budget as done in this paper.

3. BACKGROUND & PROBLEM FORMULATION

3.1 Local Differential Privacy

Local differential privacy (LDP) is a widely used notion for privacy-preserving data collection. In a typical LDP setting, there are several clients (*users*) and a data collector (*server*). Let $P = \{u_1, u_2, u_3, ...\}$ denote the client population with clients u_1, u_2, and so forth. Client u_i's true value is denoted by v_i. The domain of possible values is denoted by V, such that for all v_i, it holds that: $v_i \in V$. In LDP, each client locally encodes and perturbs their v_i using a randomized algorithm Ψ and then sends the perturbed output to the data collector. To ensure LDP, Ψ must satisfy the following definition.

[ε-LDP] A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any two inputs v_1, v_2 in the domain V:

$$\forall y \in Range(\Psi): \quad \frac{\Pr[\Psi(v_1) = y]}{\Pr[\Psi(v_2) = y]} \leq e^{\varepsilon}$$

where $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

Here, ε is a key parameter of LDP called the *privacy budget*, controlling the tradeoff between privacy and utility. A lower ε provides stronger privacy but also reduces utility.

The data collector collects perturbed outputs from the clients in P and then proceeds to perform estimation in order to recover aggregate statistics. For value $v \in V$, let f_v denote the true observation count of v in the population, and let $\bar{f}_v$ denote the estimated count. Several LDP protocols were developed in the literature with different motivations such as reducing the client-side computation cost, client-server network bandwidth usage, or minimizing the estimation error, i.e., the difference

between f_v and $\bar{f}_v$ under varying conditions.

3.2 LDP Protocols

While our budget selection problem formulation and solutions are generic (not tied to a specific LDP protocol), we use five popular LDP protocols from the literature to demonstrate their practicality: GRR, RAPPOR, OUE, BLH, and OLH. Below, we briefly describe each of these protocols.

Generalized Randomized Response (GRR). The randomized response principle was first introduced by Warner in Warner (1965). GRR extends this principle and applies it to LDP in a way that supports arbitrary V and ε. For client u_i with true value v_i, the perturbation algorithm Ψ_{GRR} modifies v_i and produces $y_i \in V$ with the following probabilities:

$$\Pr[\Psi_{\text{GRR}}(v_i) = y_i] = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+|V|-1} & \text{if } y_i = v_i \\ q = \frac{1}{e^{\varepsilon}+|V|-1} & \text{if } y_i \neq v_i \end{cases}$$

The client u_i sends the resulting y_i to the server. This satisfies ε-LDP since $p/q = e^{\varepsilon}$.

The server receives perturbed responses from all clients. To perform estimation for a value $v \in V$, the server first counts $\widehat{f}_v$ which is defined as the total number of clients u_j in P who reported $y_j = v$. Then, the server estimates $\bar{f}_v$ as:

$$\bar{f}_v = \frac{\widehat{f}_v - |P| \cdot q}{p - q}$$

Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) is a system originally developed by Google and implemented in Chrome Erlingsson et al. (2014). The original version of RAPPOR utilizes Bloom filters to encode strings and other types of high-dimensional data into bitvectors. In this paper, we utilize a variant of RAPPOR with unary encoding, which is commonly used in the literature Arcolezi, Couchot, Al Bouna & Xiao (2022); Gursoy et al. (2022); Gursoy, Tamersoy, Truex, Wei & Liu (2019); Qin, Yang, Yu, Khalil, Xiao & Ren (2016); Wang et al. (2017). Client u_i with true value v_i initializes a bitvector B_i with length equal to $|V|$ full of 0s, and sets $B_i[v_i] = 1$. The perturbation algorithm Ψ_{RAP} takes B_i as input and produces a perturbed bitvector B'_i. To generate B'_i,

Ψ_{RAP} processes each bit in B_i individually, either preserving or flipping it with the following probabilities:

$$\forall_{j\in[0,|V|-1]}: \quad \Pr[B_i'[j]=1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_i[j]=1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_i[j]=0 \end{cases}$$

The client u_i sends the resulting B_i' to the server.

The server receives perturbed bitvectors from all clients. To perform estimation for $v \in V$, the server first finds $Sup(v)$, which is defined as the number of received bitvectors satisfying: $B_i'[v]=1$. Then, the server estimates $\bar{f}_v$ as:

$$\bar{f}_v = \frac{Sup(v)+|P|\cdot(\alpha-1)}{2\alpha-1}$$

where α is the bit keeping probability: $\alpha = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$.

Optimized Unary Encoding (OUE) was proposed in Wang et al. (2017). It employs the same bitvector encoding process as RAPPOR with unary encoding explained above. However, the bit preservation and bit flipping probabilities in OUE are different so that estimation accuracy can be increased. Client u_i initializes bitvector B_i such that $B_i[v_i]=1$ and all remaining positions are 0s. The perturbation algorithm Ψ_{OUE} takes B_i as input and produces a perturbed bitvector B_i' where:

$$\forall_{j\in[0,|V|-1]}: \quad \Pr[B_i'[j]=1] = \begin{cases} \frac{1}{2} & \text{if } B_i[j]=1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_i[j]=0 \end{cases}$$

The client u_i sends the resulting B_i' to the server.

The server receives perturbed bitvectors from all clients. To perform estimation for $v \in V$, the server again finds $Sup(v)$, which is defined as the number of received bitvectors satisfying: $B_i'[v]=1$. Then, the server estimates $\bar{f}_v$ as:

$$\bar{f}_v = \frac{2\cdot\left((e^{\varepsilon}+1)\cdot Sup(v)-|P|\right)}{e^{\varepsilon}-1}$$

Binary Local Hashing (BLH) is a hash-based adaptation of the succinct histograms idea from Bassily & Smith (2015) used in several works Arcolezi & Gambs (2024); Gursoy et al. (2022); Wang et al. (2017). Let $\mathcal{H}$ be a universal hash function family where each hash function $H \in \mathcal{H}$ maps a value from V to a single bit, i.e., $H: V \to \{0,1\}$. Each client u_i randomly draws a hash function from $\mathcal{H}$, i.e., $H_i \leftarrow_{\$} \mathcal{H}$. Then, the client computes bit b_i as: $b_i \leftarrow H_i(v_i)$. The result of the

encoding is $\langle H_i, b_i \rangle$. Then, the perturbation algorithm Ψ_{BLH} perturbs b_i to b'_i:

$$\Pr[b'_i = 1] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon + 1} & \text{if } b_i = 1 \\ \frac{1}{e^\varepsilon + 1} & \text{if } b_i = 0 \end{cases}$$

The client u_i sends $\langle H_i, b'_i \rangle$ to the server.

The server receives tuples of the form $\langle H_i, b'_i \rangle$ from all clients $u_i \in P$. To perform estimation for some value $v \in V$, the server computes $Sup(v)$ as the total number of clients whose reported tuples satisfy the constraint: $b'_i = H_i(v)$. Then, the server estimates $\bar{f}_v$ as:

$$\bar{f}_v = \frac{(e^\varepsilon + 1) \cdot (2 \cdot Sup(v) - |P|)}{e^\varepsilon - 1}$$

Optimized Local Hashing (OLH) was proposed in Wang et al. (2017). It differs from BLH by extending the output space of the hash functions in family $\mathcal{H}$ to non-binary. In OLH, it is assumed that hash functions in $\mathcal{H}$ output integers between $[0, k-1]$, where $k \geq 2$ is a parameter of the protocol. Using larger k is beneficial when ε and V are large since it addresses BLH's utility loss when binary encoding is suboptimal, e.g., due to hash collisions. The recommended value of k was derived as $k = e^\varepsilon + 1$ in Wang et al. (2017,1); we use this k value by default.

Let $\mathcal{H}$ be a universal hash function family where each $H \in \mathcal{H}$ maps a value from V to an integer in the range $[0, k-1]$, i.e., $H : V \to [0, k-1]$. Each client u_i randomly draws a hash function from $\mathcal{H}$, i.e., $H_i \leftarrow_{\$} \mathcal{H}$. Then, the client computes integer x_i as: $x_i \leftarrow H_i(v_i)$. The result of the encoding is $\langle H_i, x_i \rangle$. The perturbation algorithm Ψ_{OLH} perturbs x_i to x'_i:

$$\forall_{j \in [0, k-1]} : \quad \Pr[x'_i = j] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon + k - 1} & \text{if } x_i = j \\ \frac{1}{e^\varepsilon + k - 1} & \text{if } x_i \neq j \end{cases}$$

The client u_i sends $\langle H_i, x'_i \rangle$ to the server.

The server receives tuples of the form $\langle H_i, x'_i \rangle$ from all clients $u_i \in P$. To perform estimation for some value $v \in V$, the server computes $Sup(v)$ as the total number of clients whose reported tuples satisfy the constraint: $x'_i = H_i(v)$. Then, the server estimates $\bar{f}_v$ as:

$$\bar{f}_v = \frac{(e^\varepsilon + k - 1) \cdot (k \cdot Sup(v) - |P|)}{(e^\varepsilon - 1) \cdot (k - 1)}$$

3.3 Utility Metrics and Constraints

The estimators used in LDP protocols are typically unbiased, i.e., in expectation: $[\bar{f}_v] = f_v$. However, since the protocols are randomized, we generally obtain $\bar{f}_v \neq f_v$ in actual LDP executions. The difference between $\bar{f}_v$ and f_v causes utility loss. The data collector would like to reduce utility loss so that downstream tasks can be performed accurately.

In this paper, we consider the case where the data collector has a *utility constraint*, i.e., the utility loss should be no larger than a threshold ω. This constraint can be imposed by the accuracy requirements or the tolerable error amount in the downstream task. In order to mathematically and experimentally quantify the amount of utility loss, well-defined utility metrics must be used. While our problem formulation and solution are general (i.e., different metrics and constraints can be used), below we exemplify some potential utility metrics that are used in our experiments and in other LDP works.

ℓ_1 and ℓ_2 error. The average difference between f_v and $\bar{f}_v$ can be measured across $v \in V$ using ℓ_1 or ℓ_2 norms:

$$\frac{\left(\sum_{v \in V} |f_v - \bar{f}_v|^\beta\right)^{\frac{1}{\beta}}}{|V|}$$

where $\beta = 1$ or 2 depending on the norm used.

Kendall-tau coefficient. Preservation of *relative* item popularities (i.e., for two items v_1, $v_2 \in V$, is f_{v_1} higher or lower than f_{v_2}) is an important aspect for correct discovery of frequent items and heavy hitters. The Kendall-tau coefficient can be used to measure how well relative item popularities are preserved. Two items v_1, $v_2 \in V$ are said to be concordant if one of the following is true:

$$f_{v_1} > f_{v_2} \text{ and } \bar{f}_{v_1} > \bar{f}_{v_2}$$
$$f_{v_1} \leq f_{v_2} \text{ and } \bar{f}_{v_1} \leq \bar{f}_{v_2}$$

In other words, they are concordant if their relative popularities agree with and without LDP. If neither is true, i.e., their relative popularities disagree, then v_1 and v_2 are said to be discordant. Given the number of concordant and discordant pairs

in $V \times V$, the Kendall-tau coefficient τ is calculated as:

$$\tau = \frac{\text{(\# of concordant pairs) - (\# of discordant pairs)}}{\text{(\# of concordant pairs) + (\# of discordant pairs)}}$$

This metric takes values between -1 and +1, where -1 implies complete disagreement, +1 implies complete agreement, and 0 implies lack of correlation.

3.4 Problem Formulation

Consider that the data collector has a utility constraint: with probability π, the utility loss (e.g., measured using one of the metrics from Section 3.3) should be no larger than a threshold ω. The problem of *budget selection under utility constraints* can be formulated as follows: Given the utility constraint, what value of ε should be used so that clients' privacy will be maximized while the given utility constraint is met? This problem is challenging due to multiple reasons:

- Since LDP is randomized, it is not possible to guarantee the satisfaction of a utility constraint with perfect certainty. Hence, we must incorporate a confidence threshold π to indicate that the constraint will be satisfied with probability $\geq \pi$. The value of π can be large (such as $\pi = 0.9$ or 0.95) to confidently claim that the utility constraint will be met in practice.
- If the ε budget is small, then the utility loss will be high and the utility constraint is less likely to be satisfied. If the ε budget is large, then the utility loss will be low and the utility constraint is more likely to be satisfied. However, large ε is detrimental to clients' privacy. Therefore, it is desirable to select the ε budget that satisfies the utility constraint but remains as small as possible.
- The selection of the ε budget depends on the LDP protocol. Since all protocols have different client-side perturbation and server-side estimation procedures, their utility losses are also different (e.g., different protocols have different variances Wang et al. (2017)). Hence, the LDP protocol in use must be taken into account when selecting ε.

With the above considerations in mind, we formulate the budget selection problem as an optimization problem. Let PROT denote the LDP protocol being used, including

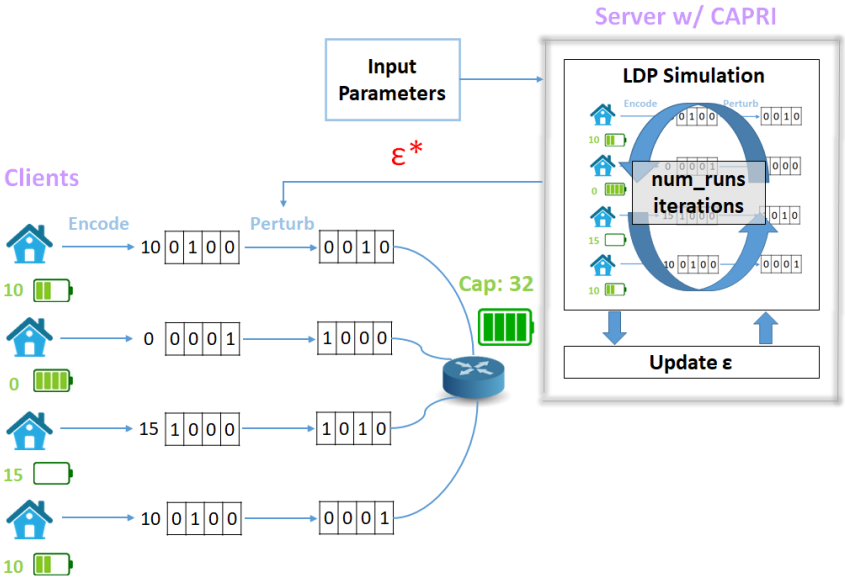


Figure 3.1 Problem setting and architecture

but not limited to GRR, RAPPOR, OUE, BLH, and OLH. Let ULoss denote the utility loss metric including but not limited to those given in Section 3.3. Let ω denote the utility threshold and π denote the probability threshold. We aim to find:

$$\begin{aligned} \text{minimize} \quad & \varepsilon \\ \text{subject to} \quad & \text{ULoss} \leq \omega \text{ with probability} \geq \pi \\ \text{given} \quad & \text{Prot}, V \end{aligned}$$

In other words, we aim to find the smallest ε (to maximize clients' privacy) that satisfies the utility constraint (i.e., ULoss $\leq \omega$ with probability $\geq \pi$). The protocol Prot and the domain V are also taken into consideration since different protocols can give more accurate or less accurate results depending on V; hence, the selected ε must be protocol-dependent.

3.5 Capacity Planing Problem Setting and Notation

Consider a setting in which there are N clients (households), collectively denoted by population $P = \{u_1, u_2, ..., u_N\}$. Each client seeks to consume a shared resource with limited total capacity. In the real world, this could correspond to water consumption, power consumption, usage of sewer systems, etc. Let *cap* denote the total capacity of the shared resource and let v_i denote the consumption amount of client u_i. The

True Total Consumption (TTC) amount for the whole population is: $TTC = \sum_{i=1}^{N} v_i$. From a capacity planning perspective, if $TTC \leq cap$, then all clients can be served successfully. However, if $TTC > cap$, then service capacity is exceeded and problems such as power outages may occur. Therefore, it is important to forecast capacity exceedances accurately.

We exemplify this setting with the help of Figure 3.1. There exist $N = 4$ clients who consume power and request to charge their batteries. For the 4 clients presented in the figure, $v_1 = 10$, $v_2 = 0$, $v_3 = 15$, and $v_4 = 10$. Therefore, $TTC = 35$. Since the capacity of the system is given as $cap = 32$ in Figure 3.1, $TTC > cap$ holds, therefore all clients should not be served simultaneously.

When all clients' consumption amounts v_i can be collected in plaintext, computing TTC and checking whether it exceeds cap is straightforward as exemplified above. On the other hand, several studies have shown that observing fine-grained consumption amounts can cause adversarial inferences Anderson et al. (2017); Beckel et al. (2013); Czétány et al. (2021); Eibl et al. (2019); Eibl & Engel (2014); Kleiminger et al. (2015); Radovanovic, Unterweger, Eibl, Engel & Reichl (2022). To prevent inferences, there is growing interest in applying privacy-preserving techniques to consumption values. In this paper, we study the application of local differential privacy (LDP). For example, in Figure 3.1, each client uses a bitvector-based LDP protocol to encode and perturb their v_i. Then, the server needs to perform capacity planning using the perturbed bitvectors received from clients.

3.6 Capacity Planning Problem Statement

Since data is collected using LDP, the server observes the Estimated Total Consumption (ETC) amount rather than TTC based on LDP protocol outputs. ETC is computed as:

$$ETC = \sum_{v \in V} v \cdot \bar{X}(v)$$

ETC is typically different from TTC because of estimation error. This difference may cause incorrect capacity exceedance flags to be raised (i.e., ETC exceeds cap but TTC is lower than cap) or actual capacity exceedances to be neglected (i.e., ETC is lower than cap but TTC exceeds cap). The four possible circumstances are

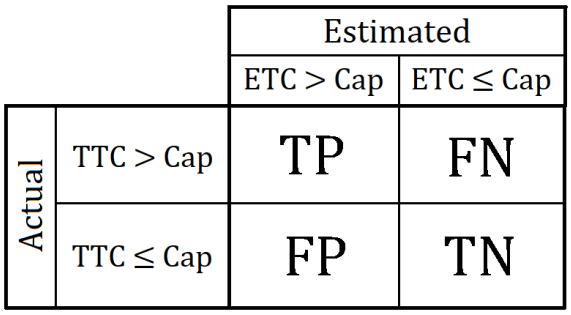


Figure 3.2 Illustration of TP, TN, FP, FN in our context

defined as:

- True Positive (TP): $TTC > cap$ and $ETC > cap$
- False Positive (FP): $TTC \leq cap$ and $ETC > cap$
- True Negative (TN): $TTC \leq cap$ and $ETC \leq cap$
- False Negative (FN): $TTC > cap$ and $ETC \leq cap$

These four circumstances are also illustrated in Figure 3.2.

For the capacity planner, it is important to determine the value of ε so that FP and FN probabilities are low, whereas TP and TN probabilities are high. Notice that under a fixed *cap*, increasing ε is expected to decrease estimation error, thereby reducing the FP and FN probabilities. Yet, increasing ε also has the negative impact of reducing clients' privacy. Therefore, the problem of budget selection is to find a minimal ε value such that clients' privacy is maximized while FP and FN probabilities are kept below an acceptable threshold (resp. TP and TN probabilities are kept above a threshold).

3.7 Capacity Planning Problem Optimization-Based Formulation

In the context of CAPRI, V denotes the domain of possible consumption values, $X(v)$ denotes the number of clients whose consumption is equal to v, and *cap* denotes the total system capacity. We denote by PROT the LDP protocol being used – it can be one of GRR, RAPPOR, OUE (see Section 3.2 for their descriptions) but CAPRI is not necessarily limited to these three protocols, i.e., other LDP protocols can also be used in the future. Then, given a maximum acceptable FN threshold denoted by

τ, CAPRI formulates the following optimization problem for budget selection:

$$\begin{aligned} \text{minimize} \quad & \varepsilon \\ \text{subject to} \quad & \Pr\Big[\text{FN}\Big] < \tau \\ \text{given} \quad & V,\ \{X(v) \mid v \in V\},\ \mathit{cap},\ \textsc{prot} \end{aligned}$$

This optimization aims to find the minimum ε budget (strongest privacy level) which satisfies the constraint that the FN probability should be below the acceptable threshold τ. Note that while the above optimization places the constraint on FNs, it is possible to modify this optimization problem so that the constraint is placed in terms of TPs, FPs or TNs. For example, if a FP threshold τ' is desired, then line 4.2.3 should be written as: $\Pr\Big[\text{FP}\Big] < \tau'$. Instead, if a TP or TN threshold τ^* is desired, then line 4.2.3 should be written as: $\Pr\Big[\text{TP}\Big] \geq \tau^*$ or $\Pr\Big[\text{TN}\Big] \geq \tau^*$. Indeed, CAPRI's solution approach works for all four types of constraints (TP, TN, FP, FN).

4. SOLUTION METHODS

4.1 OPTIMUS Methodology

Having formulated the budget selection problem as an optimization problem in Section 3.4, we now present the five solution methods in Optimus, which are named according to their underlying principles: Binary Search (BS), Relaxed Binary Search (RBS), Dynamic Binary Search (DBS), Multi-Pass Search (MPass), and Bayesian Optimization (Bayes).

4.1.1 Binary Search (BS)

The first method is inspired by the well-known Binary Search (BS) algorithm. It is a simulation-based approach with the following intuition. Say that ε_{cur} is the current budget value. Optimus *simulates* LDP data collection many times with ε_{cur} using synthetic client populations. We emphasize that these data collections are simulations with synthetic clients; therefore, there is no privacy loss for actual clients in P. The amount of utility loss is measured in each collection. If utility loss is $\leq \omega$ with probability $\geq \pi$, then ε_{cur} satisfies the given utility constraint. If ε_{cur} satisfies the utility constraint, considering that the goal of our optimization problem is to find the smallest ε that satisfies the constraint, Optimus checks whether budgets smaller than ε_{cur} can satisfy the constraint. On the other hand, if ε_{cur} does not satisfy the utility constraint, then budget values that are larger than ε_{cur} are explored in the next iterations.

In BS, this iterative search process is guided by an algorithm inspired by binary

Algorithm 1: Binary Search (BS) Based Solution

Input : V, PROT, ω, π
Parameters: ε_{min}, ε_{max}, $\mathcal{N}$, φ
Output : Selected budget ε^*

while $\varepsilon_{max} - \varepsilon_{min} \geq \varphi$ **do**
  $\varepsilon_{cur} \leftarrow (\varepsilon_{min} + \varepsilon_{max})/2$
  $satisfied \leftarrow 0$
  $sum\_uloss \leftarrow 0$
  for $i = 1$ *to* $\mathcal{N}$ **do**
    Simulate LDP data collection using PROT, V and ε_{cur} to obtain estimations $\bar{f}_v$ for $v \in V$
    $uloss \leftarrow$ Compute utility loss using $\bar{f}_v$
    $sum\_uloss \leftarrow sum\_uloss + uloss$
    if $uloss \leq \omega$ **then**
      $satisfied \leftarrow satisfied + 1$
  $pr\_satisfied \leftarrow satisfied/\mathcal{N}$
  $avg\_uloss \leftarrow sum\_uloss/\mathcal{N}$
  if $avg\_uloss \leq \omega$ *and* $pr\_satisfied \geq \pi$ **then**
    $\varepsilon_{max} \leftarrow \varepsilon_{cur}$
  else
    $\varepsilon_{min} \leftarrow \varepsilon_{cur}$
return ε_{max}

search. That is, given a range of budget values to select from (i.e., ε_{min} and ε_{max} denoting the boundaries of the initial search space), the BS algorithm initially determines $\varepsilon_{cur} = (\varepsilon_{min} + \varepsilon_{max})/2$. If ε_{cur} satisfies the utility constraint, the right half of the current search space (i.e., values between ε_{cur} and ε_{max}) are eliminated so that the search continues towards smaller budgets. However, if ε_{cur} does not satisfy the constraint, then the left half of the current search space (i.e., values between ε_{min} and ε_{cur}) are eliminated so that the search continues towards larger budgets and a budget that satisfies the constraint can be found. This way, in each iteration of the algorithm, half of the current search space is eliminated and the middle value in the remaining search space is used as the next ε_{cur}. The BS algorithm proceeds in this manner until the search space becomes too small (smaller than φ, where φ is a parameter).

In Algorithm 1, we provide an algorithmic overview of our BS method. The algorithm takes V, PROT, ω, and π as inputs, which are the same as the inputs to our optimization problem. In addition, the algorithm has four parameters: ε_{min}, ε_{max}, $\mathcal{N}$ and φ. Among them, ε_{min} and ε_{max} denote the minimum and maximum values allowed for the ε budget, corresponding to the boundaries of the initial search space. In order to ensure the search space is large enough so that a suitable ε can

be found, ε_{min} can be set arbitrarily small and ε_{max} can be set arbitrarily large. $\mathcal{N}$ is a parameter controlling how many LDP simulations will take place in each iteration of the algorithm (i.e., each different value of ε_{cur}). Higher $\mathcal{N}$ is beneficial to reduce the randomness caused by LDP, but it also increases the execution time of Algorithm 1. Finally, the φ parameter is used as the termination condition, i.e., it ensures that the algorithm will terminate when the search space becomes smaller than φ. Given the inputs and parameters, the output of Algorithm 1 is the selected budget value ε^*.

The main *while* loop of Algorithm 1 iterates as long as the difference between ε_{max} and ε_{min} is larger than φ. In each iteration, the current budget value ε_{cur} is determined as the average of ε_{min} and ε_{max} (line 2). Between lines 5-10, Algorithm 1 simulates LDP data collection using the protocol PROT, domain V, and ε_{cur}. This simulation is done by: (i) generating a synthetic client population, (ii) having all clients in this synthetic population perturb and report their data to the server using the client-side perturbation procedures of PROT, and finally, (iii) the server performs estimation using the estimation procedures of PROT. As described in Section 3.2, the results of this estimation is $\bar{f}_v$ for the items in V. The amount of utility loss for this particular simulation is computed on line 7 using $\bar{f}_v$ and the desired utility metric. Considering that $\mathcal{N}$ simulations are performed with ε_{cur}, the total utility loss across all simulations is tracked using the *sum_uloss* variable in Algorithm 1. The algorithm also keeps track of how many times the constraint was satisfied, using the $satisfied$ variable which is updated on lines 9-10. After $\mathcal{N}$ simulations with ε_{cur} are complete, the average utility loss and the probability of satisfying the utility constraint across $\mathcal{N}$ simulations are computed (lines 11-12). If the average utility loss is $\leq \omega$ and the probability of satisfaction is $\geq \pi$, then smaller ε should be searched by Algorithm 1 in the next iterations, which is ensured by setting $\varepsilon_{max} \leftarrow \varepsilon_{cur}$. Otherwise, larger ε values are searched in the next iterations, which is ensured by setting $\varepsilon_{min} \leftarrow \varepsilon_{cur}$. The algorithm continues in the next iteration using the updated ε_{min} or ε_{max}.

4.1.2 Relaxed Binary Search (RBS)

The second solution method in Optimus is Relaxed Binary Search (RBS), which is similar to BS but incorporates the following intuition. We observed that in BS, when the optimal (best choice) ε is slightly different than the current ε_{cur}, due to the randomness in LDP, it is possible that the BS algorithm eliminates the wrong

Algorithm 2: RBS Solution

Input : V, PROT, ω, π
Parameters: ε_{min}, ε_{max}, $\mathcal{N}$, φ, g
Output : Selected budget ε^*

1 **while** $\varepsilon_{max} - \varepsilon_{min} \geq \varphi$ **do**
2 $\quad step \leftarrow (\varepsilon_{max} - \varepsilon_{min})/g$
3 $\quad$ Same as lines 2-12 of Algorithm 1
4 $\quad$ **if** $avg\_uloss \leq \omega$ *and* $pr\_satisfied \geq \pi$ **then**
5 $\quad\quad \varepsilon_{max} \leftarrow \varepsilon_{cur} + step$
6 $\quad$ **else**
7 $\quad\quad \varepsilon_{min} \leftarrow \varepsilon_{cur} - step$
8 **return** ε_{max}

half of the search space. For example, consider $\varepsilon_{cur} = 0.5$ and the optimal ε is 0.48. Then, it is possible, due to the randomness in LDP, that BS finds $\varepsilon_{cur} = 0.5$ does not satisfy the utility constraint, and therefore budgets smaller than 0.5 are eliminated from the search space. Even if this happens, the BS solution will eventually converge to a selected ε^* close to 0.5 (e.g., 0.52 or 0.53) but these ε^* values are nevertheless larger than the best choice of 0.48, which indicates that the selected ε^* is suboptimal (redundant privacy loss for clients).

To address this issue, RBS modifies BS by introducing a *step* size whose value is controlled by parameter g. We provide the algorithmic overview of RBS in Algorithm 2. Its inputs and outputs are identical to Algorithm 1 with the addition of the g parameter. The step size is determined according to the g parameter and the current ε_{min} and ε_{max} as: $step \leftarrow (\varepsilon_{max} - \varepsilon_{min})/g$. Then, when updating ε_{max} or ε_{min} for the next iterations of binary search, instead of assigning ε_{cur} to ε_{max} or ε_{min} directly, a relaxation of *step* is added (lines 4-7 of Algorithm 2). This ensures that slightly lower and higher values remain in the search space for the next iterations, therefore RBS does not suffer from the issue explained in the previous paragraph. On the other hand, since the size of the search space does not shrink as quickly in RBS as in BS, RBS is generally slower than BS.

4.1.3 Dynamic Binary Search (DBS)

The third solution method in Optimus is Dynamic Binary Search (DBS), which is motivated by the following observation. In BS, the elimination of the wrong half of the search space occurs because of the randomness in LDP, which can be

Algorithm 3: DBS Solution

Input : V, PROT, ω, π
Parameters: ε_{min}, ε_{max}, $\mathcal{N}_{min}$, $\mathcal{N}_{max}$, φ, C
Output : Selected budget ε^*

1 **while** $\varepsilon_{max} - \varepsilon_{min} \geq \varphi$ **do**
2 Same as lines 2-12 of Algorithm 1 using $\mathcal{N}_{min}$ in place of $\mathcal{N}$
3 $\mathcal{N}_{desired} \leftarrow \text{MIN}\Big(\Big\lfloor \frac{C}{|avg\_uloss\ -\ \omega|} \Big\rfloor, \mathcal{N}_{max}\Big)$
4 $\mathcal{N}_{add} \leftarrow \mathcal{N}_{desired} - \mathcal{N}_{min}$
5 **for** $i = 1$ *to* $\mathcal{N}_{add}$ **do**
6 Simulate LDP data collection using PROT, V and ε_{cur} to obtain estimations $\bar{f}_v$ for $v \in V$
7 $uloss \leftarrow$ Compute utility loss using $\bar{f}_v$
8 $sum\_uloss \leftarrow sum\_uloss + uloss$
9 **if** $uloss \leq \omega$ **then**
10 $satisfied \leftarrow satisfied + 1$
11 $pr\_satisfied \leftarrow satisfied / \mathcal{N}_{desired}$
12 $avg\_uloss \leftarrow sum\_uloss / \mathcal{N}_{desired}$
13 **if** $avg\_uloss \leq \omega$ *and* $pr\_satisfied \geq \pi$ **then**
14 $\varepsilon_{max} \leftarrow \varepsilon_{cur}$
15 **else**
16 $\varepsilon_{min} \leftarrow \varepsilon_{cur}$
17 **return** ε_{max}

circumvented by enforcing a sufficiently large $\mathcal{N}$. However, using a large value of $\mathcal{N}$ in cases where ε_{cur} is too different from the optimal ε is counterproductive, since the utility loss caused by ε_{cur} remains substantially far from ω, and large $\mathcal{N}$ causes the algorithm to terminate slower since more number of simulations need to be performed. In such cases, large $\mathcal{N}$ incurs the cost of increased execution time but does not provide much benefit in terms of finding the optimal budget. Hence, it is desirable to: (i) use large $\mathcal{N}$ when ε_{cur} is similar to optimal ε, i.e., when the utility loss is similar to ω, and (ii) use small $\mathcal{N}$ when ε_{cur} is far from optimal ε, i.e., when the utility loss is far from ω.

To achieve this, DBS proposes to use variable values of $\mathcal{N}$. That is, for each ε_{cur}, we dynamically determine the value of $\mathcal{N}$. The algorithmic description of DBS is given in Algorithm 3. As shown in Algorithm 3, we first perform $\mathcal{N}_{min}$ simulations using ε_{cur} (where $\mathcal{N}_{min}$ is a parameter of DBS) and calculate the average utility loss avg_uloss for these simulations. Then, we determine the desired $\mathcal{N}$ value for ε_{cur}, denoted by $\mathcal{N}_{desired}$, as:

$$\mathcal{N}_{desired} \leftarrow \text{MIN}\Big(\Big\lfloor \frac{C}{|avg\_uloss\ -\ \omega|} \Big\rfloor, \mathcal{N}_{max}\Big)$$

Because of the $avg\_uloss - \omega$ term in the denominator, when the average utility loss is similar to ω, larger $\mathcal{N}_{desired}$ is obtained. This achieves the goal of determining $\mathcal{N}_{desired}$ according to the similarity between avg_uloss and ω. The parameter C in the numerator acts as a constant multiplier to reduce or amplify the difference between avg_uloss and ω when determining $\mathcal{N}_{desired}$. Values of C can range from 10^{-6} to 10^{6} (we experiment with varying C in Section 5). Finally, to ensure that the execution times remain feasible, $\mathcal{N}_{desired}$ should not be extremely large. Hence, an upper bound of $\mathcal{N}_{max}$ is enforced in Equation 17.

In Algorithm 3, after $\mathcal{N}_{min}$ initial simulations are performed (line 2) and $\mathcal{N}_{desired}$ is determined (line 3), the additional number of simulations that are needed to reach $\mathcal{N}_{desired}$ is computed as: $\mathcal{N}_{add} \leftarrow \mathcal{N}_{desired} - \mathcal{N}_{min}$ (line 4). Afterward, $\mathcal{N}_{add}$ simulations are performed (lines 5-10), and the overall $pr\_satisfied$ and avg_uloss values across $\mathcal{N}_{desired} = \mathcal{N}_{min} + \mathcal{N}_{add}$ number of simulations are found on lines 11-12. Finally, half of the search space is eliminated depending on whether $pr\_satisfied$ and avg_uloss satisfy the given utility constraint (lines 13-16).

4.1.4 Multi-Pass Search (MPass)

The fourth solution in Optimus is the Multi-Pass (MPass) algorithm, which is a new search algorithm we propose for our problem. The rationale behind MPass is to initially start with a large search space, search through it in large steps (increments), and identify the region of the space to further focus on. Then, in each consecutive pass, the region identified in the previous pass is searched with a smaller step size. This causes both the size of the search space as well as the step size to be reduced in each pass. Consequently, each consecutive pass narrows down the search space and "zooms in" to the useful ε values identified in the previous passes. Hence, the search has increasingly higher precision in each pass. After sufficiently many passes, a precise ε value can be determined. In practice, we observed that the number of passes does not need to be large, e.g., 4 or 5 passes are sufficient.

For example, say that the initial search space is $[1, 10]$ and the initial step size is 1. Then, in the first pass, ε values 1, 2, 3, ..., 10 are tested in increments of 1. Among them, ε values that satisfy the given utility constraint will be stored in a list. Say that the minimum ε value that satisfies the given constraint is 3. Then, in the next pass, the range $[3 - step,\ 3 + step]$ will be searched using smaller increments. For example, the range $[2, 4]$ can be searched using increments of 0.1. Again, the smallest ε satisfying the utility constraint will be found, and in the next pass, the

Algorithm 4: MPass Solution

Input : V, PROT, ω, π
Parameters: ε_{min}, ε_{max}, $\mathcal{N}$, $\mathcal{S}$, g
Output : Selected budget ε^*

```
1  sat_eps ← Initialize as empty list
2  step ← 1
3  for pass = 1 to S do
4      ε_cur ← ε_min
5      while ε_cur ≤ ε_max do
6          satisfied ← 0
7          sum_uloss ← 0
8          for i = 1 to N do
9              Simulate LDP data collection using PROT, V and ε_cur to obtain
                estimations f̄_v for v ∈ V
10             uloss ← Compute utility loss using f̄_v
11             sum_uloss ← sum_uloss + uloss
12             if uloss ≤ ω then
13                 satisfied ← satisfied + 1
14         pr_satisfied ← satisfied/N
15         avg_uloss ← sum_uloss/N
16         if avg_uloss ≤ ω and pr_satisfied ≥ π then
17             Add ε_cur to sat_eps
18         ε_cur ← ε_cur + step
19     ε_min ← MIN(sat_eps) − step
20     ε_max ← MIN(sat_eps) + step
21     step ← step/g
22 return MIN(sat_eps)
```

search space will be further reduced. For example, if the minimum ε in the previous pass was 2.8, then the range in the next pass will be $[2.7, 2.9]$, which can be searched in increments of 0.01. After performing a total number of $\mathcal{S}$ passes in this fashion (where $\mathcal{S}$ is a parameter of MPass), MPass finds the smallest ε that satisfies the given constraint and returns it.

An important advantage of MPass in comparison to previous solutions is that it does not eliminate a half of the search space in each pass. For example, recall that in BS, if the algorithm finds that the current ε_{cur} satisfies the given utility constraint, then values higher than ε_{cur} are eliminated. As explained earlier, due to the randomness in LDP, it is possible for this decision to be erroneous. In contrast, MPass enables values that are close to ε_{cur} in both halves of the search space to be kept (i.e., values that fall within $[\varepsilon_{cur} - step,\ \varepsilon_{cur} + step]$ are kept) whereas values substantially different from ε_{cur} are eliminated. Hence, MPass does not suffer from the problem in BS, and therefore it remains more robust to LDP randomness.

Algorithm 5: Objective function formulation in Bayes

Input : V, PROT, ω, ε_{cur}, $\mathcal{N}$

1 $sum\_uloss \leftarrow 0$
2 **for** $i = 1$ *to* $\mathcal{N}$ **do**
3 Simulate LDP data collection using PROT, V and ε_{cur} to obtain estimations $\bar{f}_v$ for $v \in V$
4 $uloss \leftarrow$ Compute utility loss using $\bar{f}_v$
5 $sum\_uloss \leftarrow sum\_uloss + uloss$
6 $avg\_uloss \leftarrow sum\_uloss/\mathcal{N}$
7 **return** ABS$(avg_uloss - \omega)$

The MPass algorithm is given in Algorithm 4. The ε values satisfying the utility constraint are kept in a list called *sat_eps* which is initially empty (line 1), and the *step* size is initialized (line 2). The loop between lines 3-21 performs each pass, where $\mathcal{S}$ denotes the total number of passes. Within each pass, ε_{cur} is varied between ε_{min} (line 4) and ε_{max} (line 5) in increments of *step*. For each ε_{cur}, similar to the previous solutions, $\mathcal{N}$ LDP simulations are performed and the average utility loss and probability of satisfying the utility constraint are computed (lines 6-15). If the utility constraint is satisfied, ε_{cur} is added to *sat_eps* (line 17). By the end of the current pass (lines 19-21), the ε values satisfying the utility constraint have been found and stored in *sat_eps*. Using the minimum among them, the search space for the next pass is determined by updating the values of ε_{min} and ε_{max} (lines 19-20). The step size for the next pass is reduced on line 21. After all passes are complete, the algorithm returns the minimum ε that satisfies the utility constraint as the selected budget (line 22).

4.1.5 Bayesian Optimization (Bayes)

The fifth solution in Optimus is based on Bayesian optimization, a prominent approach to black-box function optimization which is particularly useful when the function has no known analytical form and function evaluations are costly in terms of resources Snoek et al. (2012); Wang et al. (2023). These properties make Bayesian optimization a suitable solution in our context since: (i) In our context, utility loss is computed through LDP simulations, which are randomized. It is not possible to derive a closed-form analytical function for measuring utility loss in general. (ii) Function evaluations are done by repeating $\mathcal{N}$ LDP simulations and calculating the average utility loss. These simulations are indeed costly in terms of execution time.

Algorithm 6: Bayes Solution

Input : V, PROT, ω, π
Parameters: ε_{min}, ε_{max}, $\mathcal{N}$, *init_points*, *iter_num*
Output : Selected budget ε^*

1 $history \leftarrow$ Initialize as empty list
2 **for** $i = 1$ *to init_points* **do**
3 Sample ε_{cur} randomly between ε_{min} and ε_{max}
4 $res \leftarrow$ Evaluate Alg. 5 using V, PROT, ω, ε_{cur}, $\mathcal{N}$
5 Store the pair (ε_{cur}, res) in *history*
6 Train surrogate model $\mathcal{M}$ using *history*
7 **for** $i = 1$ *to iter_num* **do**
8 Select ε_{cur} from $\mathcal{M}$ using the acquisition function
9 $res \leftarrow$ Evaluate Alg. 5 using V, PROT, ω, ε_{cur}, $\mathcal{N}$
10 Store the pair (ε_{cur}, res) in *history*
11 Update $\mathcal{M}$ using (ε_{cur}, res)
12 $(\varepsilon^*, res) \leftarrow$ Find the pair in *history* which has minimum res
13 **return** ε^*

In order to use Bayesian optimization, first the objective function, which will be minimized by the optimizer, needs to be formulated. A custom objective function formulation is provided in Algorithm 5. This formulation calculates the average utility loss using ε_{cur} and $\mathcal{N}$ LDP simulations, similar to the earlier approaches. Then, the absolute value difference between *avg_uloss* and ω is measured and returned. Intuitively, this ensures that Bayesian optimization attempts to obtain the budget that achieves *avg_uloss* closest to ω. The larger the difference, the higher the return value of Algorithm 5, therefore a higher penalty is incurred.

After formulating the objective function, Bayesian optimization is applied as shown in Algorithm 6. In the first phase (lines 2-6 of Algorithm 6), a Gaussian Process is initialized as the surrogate model using *init_points* number of initial data points from the search space. These data points are randomly selected ε_{cur} values within the range $\varepsilon_{cur} \in [\varepsilon_{min}, \varepsilon_{max}]$ and their corresponding objective function evaluations using Algorithm 5. They are used to train the initial surrogate model and provide a probabilistic estimate across the entire search space. Then, in the second phase (lines 7-11), *iter_num* number of iterations are performed. In each iteration, the acquisition function is used to determine the next ε_{cur} to sample based on the surrogate model. Here, we use Expected Improvement (EI) as the acquisition function Mockus (1974); Wang et al. (2023). The objective function is evaluated using the selected ε_{cur}, and the surrogate model is updated according to the evaluation result. After *iter_num* iterations, the ε value which yielded *avg_uloss* closest to ω is returned.

4.1.6 Time Complexity of solutions

In this section, we analyze the time complexity of each proposed solution. Let $\mathcal{N}$ denote the number of LDP (Local Differential Privacy) simulations performed. Suppose $\varepsilon_{\min}$ and $\varepsilon_{\max}$ represent the minimum and maximum values of the privacy budget ε, respectively. Additionally, let φ be the parameter that controls the precision granularity when searching for the optimal ε value.

The time complexity of the Binary Search (BS) algorithm is formulated as:

$$T_{\text{BS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot(T_{\text{PROT}}+T_{\text{ULOSS}})\right)$$

This expression captures three main components:

- The logarithmic factor $\log_2\left(\frac{\varepsilon_{\max}\varepsilon_{\min}}{\varphi}\right)$ reflects the number of iterations required by the binary search to converge within the desired ε precision.
- $\mathcal{N}$ accounts for the number of times the protocol and utility loss are evaluated.
- T_{PROT} and T_{ULOSS} represent the time complexities of the LDP protocol execution and the utility loss computation, respectively. These depend on the specific protocol and utility function being used.

  Let $|V|$ and $|P|$ denote the domain size and the number of participating clients, respectively. If the BS method is applied using the GRR (Generalized Randomized Response) protocol and the Kendall-Tau distance as the utility function, the overall time complexity becomes:

  $$T_{\text{BS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left((|V|+|P|+|P|+|V||P|+|V|)+|V|^2\right)\right)$$

  Simplifying the redundant terms yields:

  $$T_{\text{BS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left(|V||P|+|V|^2\right)\right)$$

  When using either the RAPPOR or OUE protocols, along with the same Kendall-Tau utility function, the operations involve slightly more overhead:

$$T_{\mathrm{BS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left(|V|+|P|+|V||P|+|V||P|+|V|+|V|^2\right)\right)$$

Here, the extra $|V|$ and $|V||P|$ terms reflect additional bit-wise encoding or decoding operations typically required by RAPPOR and OUE protocols. Upon simplification:

$$T_{\mathrm{BS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left(|V||P|+|V|^2\right)\right)$$

Next, we analyze the time complexities of other proposed search algorithms, including RBS, DBS, MPass, and Bayes. Each has its unique cost structure depending on the search mechanism used.

For the RBS method, assuming the convergence rate depends on the geometric factor $\frac{2g}{2+g}$, the complexity becomes:

$$T_{\mathrm{RBS}} \in \mathcal{O}\left(\log_{\frac{2g}{2+g}}\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left(T_{\mathrm{PROT}}+T_{\mathrm{ULOSS}}\right)\right)$$

Here, the logarithmic base reflects the randomized convergence behavior, with g controlling the step size of extra interval that is supposed to be added to the left or right intervals.

The DBS method, initially has a similar expression:

$$T_{\mathrm{DBS}} \in \mathcal{O}\left(\log_2\left(\frac{\varepsilon_{\max}-\varepsilon_{\min}}{\varphi}\right)\cdot\mathcal{N}\cdot\left(T_{\mathrm{PROT}}+T_{\mathrm{ULOSS}}\right)\right)$$

However, its cost can be reformulated based on two distinct components: $N_{\min}$, the minimum number of LDP simulation runs and utility evaluation, and N_{add}, the number of adaptive number of runs added during the search:

$$T_{\mathrm{DBS}} \in \mathcal{O}\left(\log_2(\varepsilon_{\max}-\varepsilon_{\min})\cdot(N_{\min}+N_{\mathrm{add}})\cdot(T_{\mathrm{PROT}}+T_{\mathrm{ULOSS}})\right)$$

In the MPass algorithm, the time complexity is linear with respect to the total number of passes over the ε range. Let S be the number of steps, and g again be the precision controller, the expression is:

$$T_{\text{MPass}} \in \mathcal{O}\left([(\varepsilon_{\max} - \varepsilon_{\min}) + (S-1) \cdot 2g] \cdot N \cdot (T_{\text{PROT}} + T_{\text{ULOSS}})\right)$$

Finally, the Bayesian Optimization approach has the most complex structure. This formulation incorporates the full Bayesian optimization pipeline, including model initialization, acquisition-based decision making, and iterative updates, which makes it computationally more expensive.

$$\begin{aligned} T_{\text{Bayes}} \in \mathcal{O}\Big(& \text{initpoints} \cdot N \cdot (T_{\text{PROT}} + T_{\text{ULOSS}}) + T_{\text{TrainModel}} \\ & + \text{iternum} \cdot T_{\text{Acquisition}} \cdot N \cdot (T_{\text{PROT}} + T_{\text{ULOSS}}) \cdot T_{\text{UpdateModel}} \\ & + \text{initpoints} + \text{iternum} \Big) \end{aligned}$$

4.2 CAPRI Methodology

For budget selection in capacity planning problem, we use Binary Search (BS) method that is presented for Optimus. It is evident that all other methods proposed in the Optimus can also be used.

4.2.1 CAPRI Solution Method

The main algorithm behind CAPRI's solution is given in Algorithm 7. The inputs of Algorithm 7 follow from the optimization problem presented in the previous section. Additionally, Algorithm 7 contains four parameters:

– *num_runs* is a parameter that controls how many LDP simulations will take place in each iteration of CAPRI. Higher *num_runs* is better to

Algorithm 7: CAPRI's Main Algorithm

Input : V, $\{X(v) \mid v \in V\}$, cap, PROT, $metric$ (one of: TP, TN, FP, FN), threshold τ
Params: num_runs, ε_{min}, ε_{max}, φ
Output: Selected budget ε^*

1 $TTC \leftarrow \sum_{v \in V} v \cdot X(v)$
2 **while** $\varepsilon_{min} \leq \varepsilon_{max} - \varphi$ **do**
3 $\quad \varepsilon \leftarrow (\varepsilon_{min} + \varepsilon_{max})/2$
4 $\quad$ Initialize num_TP, num_FP, num_TN, num_FN as 0
5 $\quad$ **for** $k = 0$ ***to*** num_runs **do**
6 $\quad\quad ETC \leftarrow \text{SIMULATELDP}(V, \{X(v)\}, \text{PROT}, \varepsilon)$
7 $\quad\quad$ **if** $TTC > cap$ ***and*** $ETC > cap$ **then**
8 $\quad\quad\quad num\_TP \leftarrow num\_TP + 1$
9 $\quad\quad$ **else if** $TTC \leq cap$ ***and*** $ETC \leq cap$ **then**
10 $\quad\quad\quad num\_TN \leftarrow num\_TN + 1$
11 $\quad\quad$ **else if** $TTC \leq cap$ ***and*** $ETC > cap$ **then**
12 $\quad\quad\quad num\_FP \leftarrow num\_FP + 1$
13 $\quad\quad$ **else if** $TTC > cap$ ***and*** $ETC \leq cap$ **then**
14 $\quad\quad\quad num\_FN \leftarrow num\_FN + 1$
15 $\quad pr \leftarrow num\_metric/num\_runs$
16 $\quad$ **if** $\text{CHECK\_CONSTRAINT}(metric, pr, \tau)$ **then**
17 $\quad\quad \varepsilon_{max} \leftarrow \varepsilon$
18 $\quad$ **else**
19 $\quad\quad \varepsilon_{min} \leftarrow \varepsilon$
20 **return** ε_{max}

remove the impact of randomness and achieve more stable convergence. However, it also causes the algorithm to take longer time.

- ε_{min} and ε_{max} are the minimum and maximum values that the ε budget is allowed to take. They are used as boundaries of the search space for ε. It is possible to guarantee that the selected ε will remain acceptably low by setting the upper limit ε_{max} appropriately.

- φ is a parameter that controls the degree of precision for ε. If φ is high, such as 0.5, then the search for the optimized budget will conclude faster; however, the selected budget will be less precise. If φ is low such as 0.001, then the search will take longer time but the selected budget will have higher precision.

Given the inputs and parameters, the output of Algorithm 7 is the selected budget ε^*.

First, Algorithm 7 computes TTC on line 1. Then, the main loop of the algorithm executes between lines 2-19, as long as the difference between current

Algorithm 8: Check_Constraint Function

Input : *metric*, *pr*, τ
Output: Boolean result (True or False)

```
1  if metric = FP or metric = FN then
2  |  if pr < τ then
3  |  |  return True
4  |  else
5  |  |  return False
6  else if metric = TP or metric = TN then
7  |  if pr ≥ τ then
8  |  |  return True
9  |  else
10 |  |  return False
```

ε_{min} and current ε_{max} is higher than the precision parameter φ. In each iteration of the loop, the current ε is found as the mean of current ε_{min} and ε_{max}. For the current ε, the number of TPs, TNs, FPs and FNs observed are all set to 0 (lines 3-4). Afterwards, between lines 5-14, *num_runs* many LDP simulations are performed, and in each simulation, *ETC* is computed. By comparing *TTC* and *ETC* against the capacity *cap*, Algorithm 7 decides whether the current simulation results in a TP, FP, TN or FN. Accordingly, the number of TPs, TNs, FPs and FNs are updated between lines 7-14. Then, according to the *metric* of the constraint (one of TP, TN, FP, FN), line 15 computes the empirical probability of that metric, e.g., if *metric* = FN, then line 15 computes $\Pr\Big[\text{FN}\Big]$. Note that this is the empirical probability (likelihood) based on *num_runs* internal simulations. Finally, on line 16, the algorithm checks whether the empirical probability satisfies the τ threshold. If so, then a smaller ε value can be searched in the next iteration, which is done by updating ε_{max} to become equal to the ε in the current iteration. If the constraint is not satisfied, i.e., because the current ε is too strict, then a larger ε should be searched in the next iteration. To do so, ε_{min} is updated to become equal to the ε used in the current iteration. As ε_{min} and ε_{max} are updated in each iteration, they become closer. The loop terminates when the difference between them becomes smaller than or equal to φ (line 2).

The CHECK_CONSTRAINT function, which is used on line 16 of Algorithm 7, is given in Algorithm 8. It takes as input the *metric* (one of TP, TN, FP, FN), the empirical probability of that metric *pr* (e.g., if *metric* = FN, then *pr* is equal to the empirical $\Pr\Big[\text{FN}\Big]$), and the threshold τ. If *metric* is FP or FN, then lower *pr* is better, and $pr < \tau$ is desired as the constraint. Consequently, when *metric* is FP or FN, Algorithm 8 returns True if $pr < \tau$ and False otherwise. On the other hand, if *metric* is TP or TN, then higher *pr*

Algorithm 9: SimulateLDP Function

Input : V, $\{X(v) \mid v \in V\}$, PROT, ε
Output: ETC

1 $N \leftarrow \sum_{v \in V} X(v)$
2 Generate client population $P = \{u_1, u_2, ..., u_N\}$ such that $\forall v$, $X(v)$ clients have value equal to v
3 $\mathcal{R} \leftarrow \{\}$
4 **foreach** $u_i \in P$ **do**
5 $\quad y_i \leftarrow \text{PERTURB}_{\text{PROT}}(v_i, \varepsilon)$
6 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup y_i$
7 **foreach** $v \in V$ **do**
8 $\quad \bar{X}(v) \leftarrow \text{ESTIMATE}_{\text{PROT}}(\mathcal{R}, \varepsilon)$
9 $ETC \leftarrow \sum_{v \in V} v \cdot \bar{X}(v)$
10 **return** ETC

is better, and $pr \geq \tau$ is desired as the constraint. Consequently, when *metric* is TP or TN, Algorithm 8 returns True if $pr \geq \tau$ and False otherwise. The usage rationale of the CHECK_CONSTRAINT function in CAPRI is to guide the search for ε, i.e., if the constraint is being satisfied by the current ε then lower values of ε will be explored; if the constraint is not being satisfied since the current ε is too strict then higher ε will be explored.

Finally, the SIMULATELDP function, which is used on line 6 of Algorithm 7, is given in Algorithm 9. On lines 1-2, the function first simulates the existence of a client population P which satisfies the given $X(v)$. Then, $\mathcal{R}$ is initialized as an empty set on line 3; it is later used to store perturbed responses from clients. The loop between lines 4-6 ensures that we simulate LDP data perturbation and collection for each client $u_i \in P$ using the client-side perturbation algorithms of the corresponding LDP protocol PROT. For example, if PROT = GRR then Equation 3.2 is used for perturbation, if PROT = RAPPOR then Equation 3.2 is used for perturbation, etc. Afterwards, the loop between lines 7-8 ensures that we perform LDP estimation for each $v \in V$, and consequently, we recover $\bar{X}(v)$ for all v. Here, the server-side estimation algorithms of the corresponding LDP protocol PROT are used, e.g., Equation 3.2 for GRR, Equation 3.2 for RAPPOR, etc. Finally, on lines 9-10, ETC is computed using the recovered $\bar{X}(v)$ values and returned as the output of the simulation.

4.2.2 Comparison to Theoretical Bound

CAPRI's solution is practical by nature, i.e., it relies on LDP simulations for measuring the TP, TN, FP and FN probabilities. On the other hand, it is also possible to derive theoretical bounds for TP, TN, FP and FN probabilities. In this section, to demonstrate the added value of CAPRI and justify its practical solution rather than relying on a straightforward application of theoretical bounds, we provide a comparison between CAPRI and theoretical bounds.

Consider the following scenario: We have a binary domain $V = \{0, 1\}$ and the GRR protocol is used. Since $|V| = 2$, it follows from Equation 3.2 that $p = \frac{e^{\varepsilon}}{e^{\varepsilon}+1}$ and $q = \frac{1}{e^{\varepsilon}+1}$. The number of clients is $N = X(0) + X(1)$. Furthermore, it follows from the definitions of TTC and ETC that: $TTC = X(1)$ and $ETC = \bar{X}(1)$. According to Chebyshev's inequality, we derive that the following holds:

$$\Pr\Big[\text{FN}\Big] \leq \frac{p \cdot q \cdot N}{(p-q)^2 \cdot (TTC - cap)^2}$$

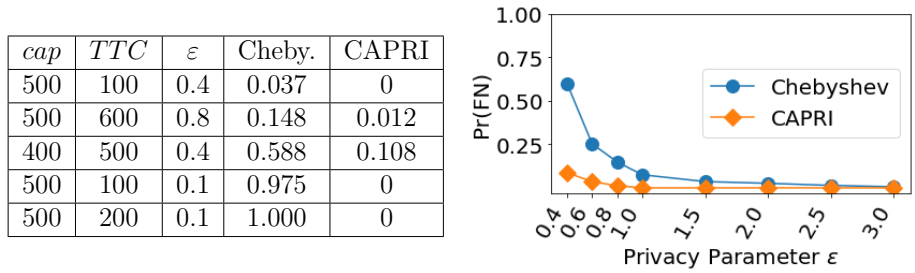
Due to maintain the readability, we give the full derivation of Equation 4.2.3 in next section.

Equation 4.2.3 gives a theoretical bound on the FN probability according to Chebyshev's inequality. However, we found that this is not a tight bound in our setting. We demonstrate this with multiple examples in Figure 4.1. We create scenarios with different N, cap, TTC and ε values, and for each scenario, we compute Pr[FN] in two ways: using Equation 4.2.3 (denoted by "Chebyshev" in Figure 4.1) and using LDP simulations as done in CAPRI (denoted by "CAPRI" in Figure 4.1).

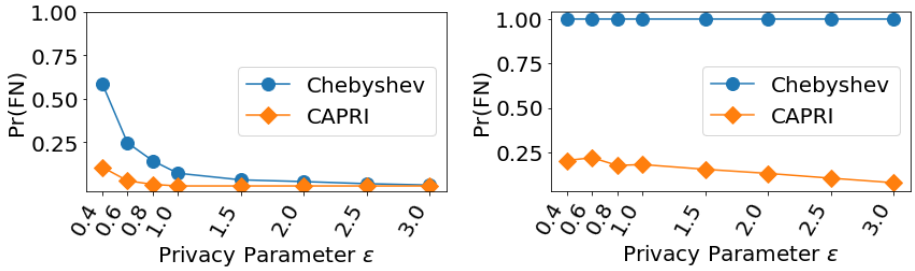
In Figure 4.1a, we create five scenarios with $N = 1000$ and different cap, TTC and ε values. In all cases with $cap > TTC$, CAPRI's method correctly finds that Pr[FN] = 0. On the other hand, non-zero probabilities are found by the Chebyshev bound. In cases with $cap < TTC$, non-zero probabilities are found by both Chebyshev and CAPRI. Yet, Chebyshev's findings are typically much higher than CAPRI, e.g., 0.588 for Chebyshev whereas 0.108 for CAPRI. This shows that Equation 4.2.3 is typically a loose bound which overestimates Pr[FN]. In Figures 4.1b, 4.1c and 4.1d, we fix cap and TTC and compute Pr[FN] according to Chebyshev and CAPRI. We once again observe that under each ε, Pr[FN] with Chebyshev is an overestimation of CAPRI. Their difference is usually higher when ε is low, but they converge as ε becomes higher.

We emphasize that the empirical results found by CAPRI do not violate or contradict the Chebyshev bound. According to the Chebyshev inequality, Equation 4.2.3 states that Pr[FN] must be lower than or equal to the bound.

cap	*TTC*	ε	Cheby.	CAPRI
500	100	0.4	0.037	0
500	600	0.8	0.148	0.012
400	500	0.4	0.588	0.108
500	100	0.1	0.975	0
500	200	0.1	1.000	0



(a) $N = 1000$ **(b)** $cap = 500$, $TTC = 600$



(c) $cap = 400$, $TTC = 500$ **(d)** $cap = 500$, $TTC = 500$

Figure 4.1 Comparing theoretical Chebyshev bound versus empirical results (CAPRI) in terms of Pr[FN]. It can be observed that the Chebyshev bound is indeed not tight when compared against empirical results.

In all empirical results of CAPRI, Pr[FN] is indeed lower than or equal to this bound. However, since the empirical results are *much* lower than the Chebyshev bound, the bound is not tight. If a loose bound is used in practice, as Figure 4.1 shows, unnecessarily high ε budgets may be selected for a given Pr[FN] constraint. This would unnecessarily reduce clients' privacy. Instead, using the empirical methodology of CAPRI, a tighter computation of Pr[FN] enables selecting a lower and more precise ε budget, which improves clients' privacy.

4.2.3 Proof of Chebyshev Bound

In the previous section, we stated that the following holds according to the Chebyshev inequality:

$$\Pr[\text{FN}] \leq \frac{p \cdot q \cdot N}{(p-q)^2 \cdot (TTC - cap)^2}$$

Below, we give the full derivation for this bound.

First recall that FN occurs when:

$$ETC \leq cap < TTC$$

By rearranging this inequality, we get:

$$ETC - TTC \geq TTC - cap$$

This implies that the estimation error satisfies:

$$\text{Estimation Error} \geq TTC - cap$$

According to the inequalities above, the FN probability, denoted by Pr[FN], is equal to:

$$\Pr[\text{FN}] = \Pr[TTC - ETC \geq TTC - cap]$$

Recall from the main paper that the domain is binary $V = \{0, 1\}$, and therefore ETC is equal to the estimated number of 1s, i.e., $ETC = \bar{X}(1)$. From the estimation procedure of GRR, we know that:

$$ETC = \bar{X}(1) = \frac{\widehat{X}(1) - N \cdot q}{p - q}$$

where $\widehat{X}(1)$ is equal to the total number of clients in the population P who report 1. There are two ways in which a client can report 1: (i) the client's true value is $v_i = 1$ and after the GRR protocol is applied, the client reports 1 with probability p, (ii) the client's true value is $v_i = 0$ and after the GRR protocol is applied, the client reports 1 with probability q. $\widehat{X}(1)$ is the sum of these two cases. That is, we can write:

$$\widehat{X}(1) = A + B$$

where A and B are two random variables denoting the first and second cases, respectively.

Random variable A: There are TTC many clients whose true value is 1. For each client, he/she reports 1 with probability p. Therefore, A can be repre-

sented using a Binomial distribution with TTC trials and p success probability:

$$A = \text{Bin}(TTC, p)$$

By properties of Binomial distribution, the variance of A is:

$$\begin{aligned}\text{Var}[A] &= TTC \cdot p \cdot (1-p) \\ &= TTC \cdot p \cdot q\end{aligned}$$

Random variable B: There are $N - TTC$ many clients whose true value is 0. For each client, he/she reports 1 with probability q. Therefore, B can be represented using a Binomial distribution with $N - TTC$ trials and q success probability:

$$B = \text{Bin}(N - TTC, q)$$

By properties of Binomial distribution, the variance of B is:

$$\begin{aligned}\text{Var}[B] &= (N - TTC) \cdot q \cdot (1-q) \\ &= (N - TTC) \cdot p \cdot q\end{aligned}$$

Properties of Estimation: Since GRR estimation is unbiased, Equation 4.2.3 satisfies:

$$\mathbb{E}[ETC] = \frac{\mathbb{E}[A] + \mathbb{E}[B] - N \cdot q}{p - q} = TTC$$

Following from Equation 4.2.3, we have:

$$\begin{aligned}\Pr[\text{FN}] &= \Pr[TTC - ETC \geq TTC - cap] \\ &= \Pr[|ETC - TTC| \geq TTC - cap]\end{aligned}$$

Application of Chebyshev inequality: To compute the probability stated in Equation 4.2.3, we can use the Chebyshev inequality. According to the Chebyshev inequality, if Z is a random variable with finite expected value $\mathbb{E}[Z]$ and finite non-zero variance $\text{Var}[Z]$, for any real number $a > 0$:

$$\Pr[|Z - \mathbb{E}[Z]| \geq a] \leq \frac{\text{Var}[Z]}{a^2}$$

Applying Equation 4.2.3 to Equation 4.2.3, and recalling from Equation 4.2.3

that $\mathbb{E}[ETC] = TTC$, we obtain:

$$\Pr[|ETC - TTC| \geq TTC - cap] \leq \frac{\mathrm{Var}[ETC]}{(TTC - cap)^2}$$

Now we need to compute $\mathrm{Var}[ETC]$:

$$\begin{aligned}
\mathrm{Var}[ETC] &= \mathrm{Var}\left[\frac{A + B - N \cdot q}{p - q}\right] \\
&= \mathrm{Var}\left[\frac{1}{p-q} \cdot A + \frac{1}{p-q} \cdot B - \frac{N \cdot q}{p-q}\right] \\
&= \mathrm{Var}\left[\frac{1}{p-q} \cdot A + \frac{1}{p-q} \cdot B\right] \\
&= (\frac{1}{p-q})^2 \cdot \mathrm{Var}[A] + (\frac{1}{p-q})^2 \cdot \mathrm{Var}[B] \\
&\quad + 2 \cdot \frac{1}{p-q} \cdot \frac{1}{p-q} \cdot \mathrm{Cov}[A, B] \\
&= (\frac{1}{p-q})^2 \cdot \Big(\mathrm{Var}[A] + \mathrm{Var}[B] + 2 \cdot \mathrm{Cov}[A, B]\Big) \\
&= (\frac{1}{p-q})^2 \cdot \Big(\mathrm{Var}[A] + \mathrm{Var}[B]\Big) \\
&= (\frac{1}{p-q})^2 \cdot \Big(TTC \cdot p \cdot q + (N - TTC) \cdot p \cdot q\Big)
\end{aligned}$$

Since A and B are independent random variables, $\mathrm{Cov}[A, B] = 0$. Therefore, we arrive at:

$$\mathrm{Var}[ETC] = (\frac{1}{p-q})^2 \cdot (N \cdot p \cdot q)$$

Substituting Equation 4.2.3 into Equation 4.2.3:

$$\begin{aligned}
&\Pr[|ETC - TTC| \geq TTC - cap] \\
&\qquad \leq \frac{(\frac{1}{p-q})^2 \cdot (N \cdot p \cdot q)}{(TTC - cap)^2}
\end{aligned}$$

Therefore, recalling from Equation 4.2.3 that $\Pr[\mathrm{FN}] = \Pr[|ETC - TTC| \geq TTC - cap]$, we arrive at what we want to prove:

$$\Pr[\mathrm{FN}] \leq \frac{p \cdot q \cdot N}{(p-q)^2 \cdot (TTC - cap)^2}$$

5. EXPERIMENTAL EVALUATION

5.1 Experiments on OPTIMUS

5.1.1 Experiment Setup for Optimus

We implemented all components of Optimus in Python, including all LDP protocols, utility metrics, and solution methods. In this section, we perform experiments to evaluate the consistency and execution times of the different methods, as well as the impacts of their hyperparameters. We conduct experiments with five LDP protocols (GRR, RAPPOR, OUE, BLH, OLH), the five solution methods proposed in Section 4 (BS, RBS, DBS, MPass, Bayes), and two datasets: Emojis and MSNBC. By default, we used the Kendall-tau coefficient as our utility metric with varying τ. In all solution methods, we used the default range of $\varepsilon_{min} = 0.001$ and $\varepsilon_{max} = 20$ to cover a large range of possibilities. We used $\varphi = 0.01$ as it typically offers sufficient precision for ε in BS, RBS, and DBS, $g = 4$ in RBS, $C = 1$ in DBS, and $\mathcal{S} = 5$ in MPass as the default hyperparameter values. More details about the datasets are given below.

Emoji dataset is based on the emoji usage statistics found on `emojistats.org`. Since there exist too many unique emojis, to limit the size of V, we included only those emojis that were sent at least 500.000 times. For each emoji, we retrieved their frequency f_v from `emojistats.org` and normalized them using the frequencies of the remaining emojis in V.

MSNBC dataset contains logs from `msnbc.com` for September 28, 1999. We

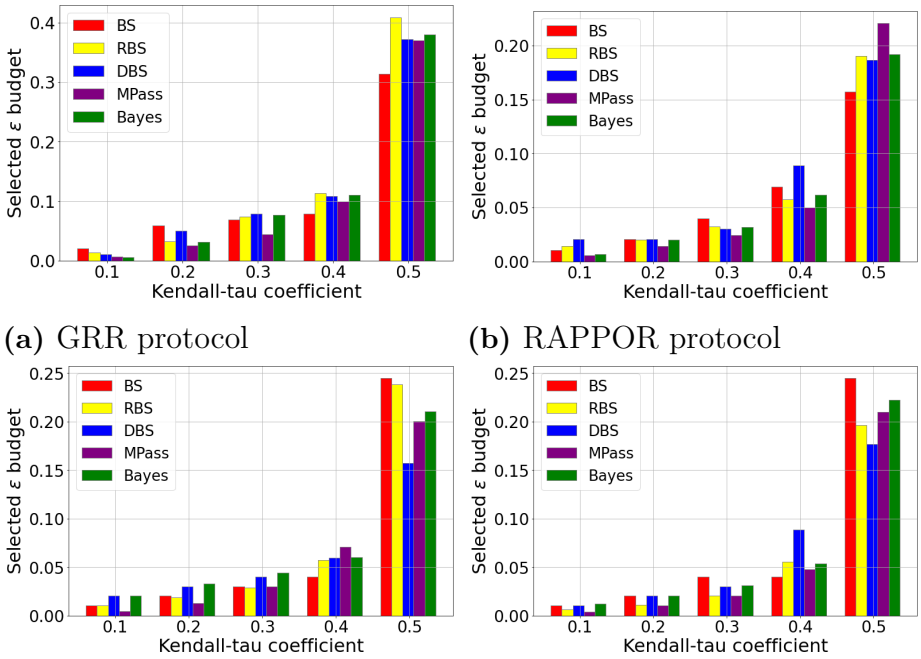
retrieved it from the UCI ML Repository[1]. Each row in the dataset represents a user's sequence of page visits during that 24-hour period. The visits are recorded by page categories, such as news, tech, weather, and sports. There are $|V| = 17$ categories. Many users either visit only one category of pages, or if they visit multiple categories, their visits are concentrated in one dominant category. Therefore, for each user, we identified their most frequently visited category and used this category as the user's true value. We calculated f_v of each category after this pre-processing.

5.1.2 Consistency of Solution Methods

We first compare BS, RBS, DBS, MPass, and Bayes in terms of the consistency of their results, i.e., whether their selected ε budgets are in agreement. Results with the MSNBC dataset are presented in Figure 5.1 and results with the Emoji dataset are presented in Figure 5.2. The experiments are performed separately for each different LDP protocol. In Figure 5.2d, results with only BS, RBS and DBS solutions are given since MPass and Bayes have prohibitively long execution times (their total execution time on this dataset exceeded 2 days). In the remaining plots, all five solution methods are included. In all plots, we observe that as the value of the correlation coefficient increases, the selected ε budget also increases. For example, ε is selected to be below 2 in Figure 5.2a when τ is 0.3; however, it becomes close to 6 when τ is 0.9. This is because a higher correlation coefficient requires the relative item popularity rankings with and without LDP to have increasingly higher resemblance, which means the utility constraint gets stricter. As a result, to decrease utility loss in LDP, all solution methods in Optimus increase the ε value. This validates our intuition that *stricter utility constraints cause the optimized ε budget to become larger.*

Next, we consider the following question: *Are the results of the solution methods consistent, i.e., in agreement?* This is an important question due to two main reasons: (i) The solution methods are heuristic, therefore they are not guaranteed to always find the optimal ε. Hence, it is valuable to not rely on a single method alone, and use multiple of them to validate each other. (ii) If a method outputs substantially different results compared to the others, this

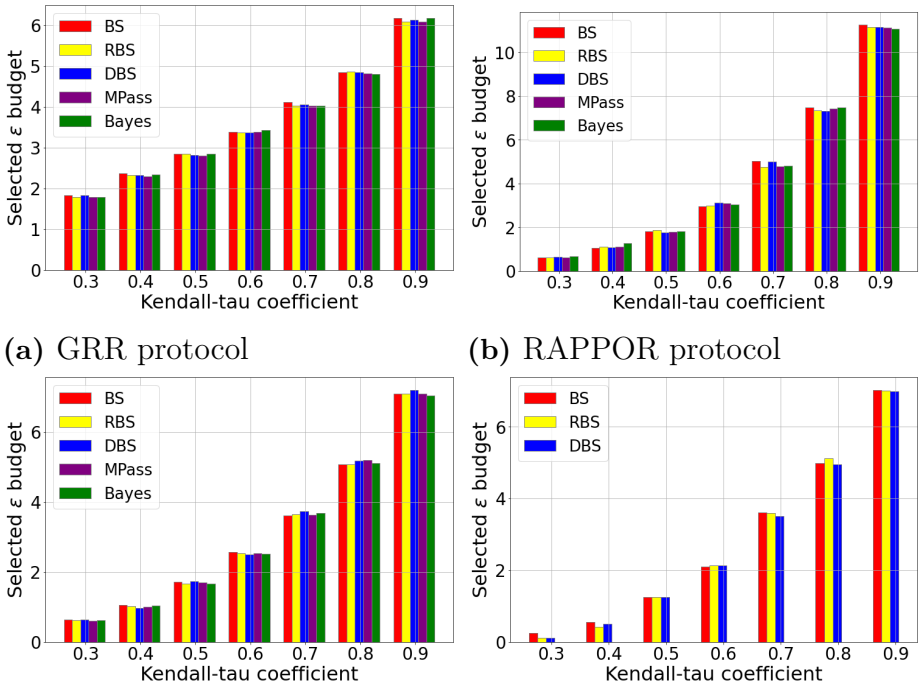
[1] `http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data`



(a) GRR protocol **(b)** RAPPOR protocol

(c) OUE protocol **(d)** BLH protocol

Figure 5.1 Comparison of BS, RBS, DBS, MPass, and Bayes methods using four protocols (GRR, RAPPOR, OUE, BLH) in terms of the selected ε budgets. MSNBC dataset with varying Kendall-tau coefficients between 0.1 and 0.5 are used.



(a) GRR protocol **(b)** RAPPOR protocol

(c) OUE protocol **(d)** OLH protocol

Figure 5.2 Comparison of BS, RBS, DBS, MPass, and Bayes methods using four protocols (GRR, RAPPOR, OUE, OLH) in terms of the selected ε budgets. Emoji dataset with varying Kendall-tau coefficients between 0.3 and 0.9 are used.

may indicate that this method did not converge properly. Overall, based on

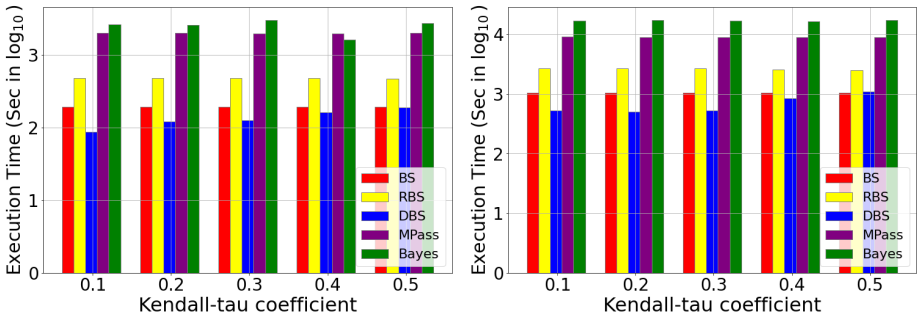
the results reported in Figures 5.1 and 5.2, we observe that the resulting ε values of all methods are similar to each other. The results in Figure 5.2 are particularly similar. In general, the difference between any two methods is at most 0.1, which indicates that the methods are in agreement. We also observe from Figure 5.1 that BS can sometimes yield lower or higher ε compared to other methods, which supports our motivation for proposing alternative solutions such as RBS, DBS, and MPass.

A final observation we make from Figures 5.1 and 5.2 is that the selected ε is dependent on the protocol and dataset. For example, the ε values found for GRR are different from RAPPOR, OUE, BLH, and OLH. Also, the ε values for the MSNBC dataset are typically less than 0.4, whereas the ε values for the Emoji dataset are much larger, e.g., 2, 4, or 6. These trends are because different protocols have different utility losses, and the domains V are different from dataset to dataset. Collectively, these results demonstrate that incorporating V and PROT into the optimization problem formulated in Section 3.4 is indeed necessary when selecting the ε budgets.

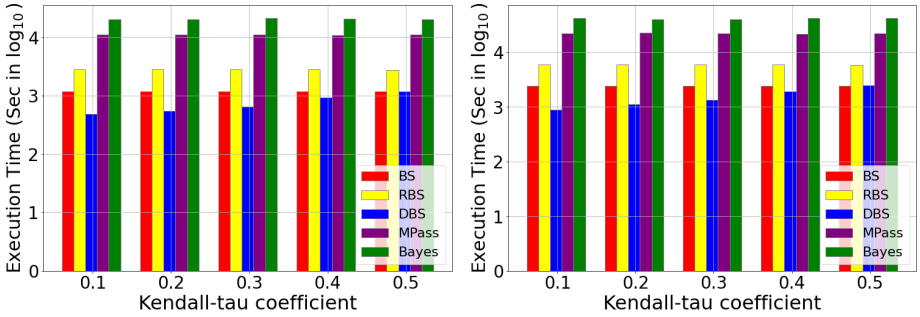
5.1.3 Execution Time Analysis

In addition to accuracy and consistency, an important aspect in optimized ε selection is *speed*. It is desirable to use a method that finds the optimized ε as quickly as possible. Hence, in this section, we compare BS, RBS, DBS, MPass, and Bayes in terms of execution time. In Figures 5.3 and 5.4, we present their execution times on the MSNBC and Emoji datasets, respectively. Execution times are internally dependent on the speed of the underlying LDP protocol; therefore, results with each different protocol are drawn in a separate plot. For example, among the four protocols (GRR, RAPPOR, OUE, BLH), GRR is the fastest since it does not perform encoding on the clients' data. Since RAPPOR and OUE both use bitvectors of length $|V|$ for encoding and perturbation, they have similar execution times, which are higher than GRR. Finally, BLH has the highest execution time.

Comparing BS, RBS, DBS, MPass, and Bayes across all LDP protocols, we observe that Bayes and MPass consistently have the highest execution times. We note that although Bayes has the highest execution time, its execution time can be reduced by using lower *iter_num* and *init_points*. However, lower *iter_num* and *init_points* can also reduce the accuracy of Bayesian

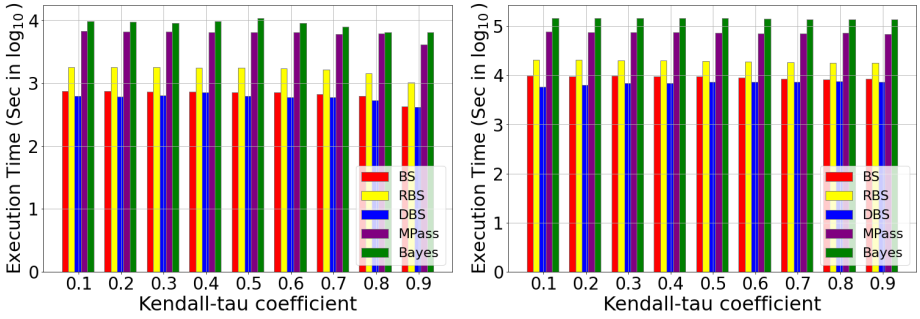


(a) GRR protocol (b) RAPPOR protocol

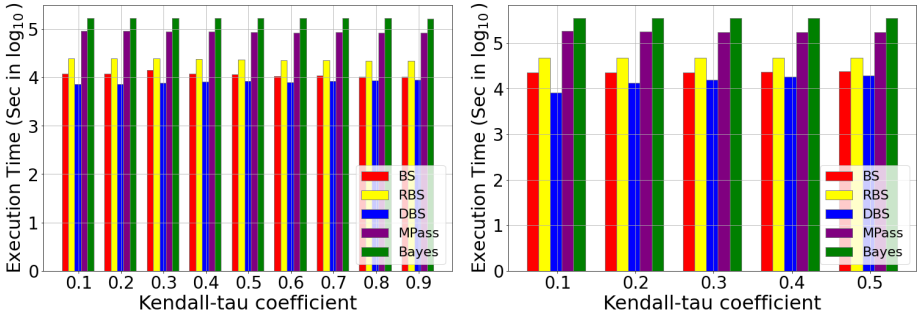


(c) OUE protocol (d) BLH protocol

Figure 5.3 Comparison of BS, RBS, DBS, MPass, and Bayes in terms of execution time under varying protocols and Kendall-tau coefficients (MSNBC dataset). Note that the y axis is in logarithmic scale.



(a) GRR protocol (b) RAPPOR protocol



(c) OUE protocol (d) BLH protocol

Figure 5.4 Comparison of BS, RBS, DBS, MPass, and Bayes in terms of execution time under varying protocols and Kendall-tau coefficients (Emoji dataset). Note that the y axis is in logarithmic scale.

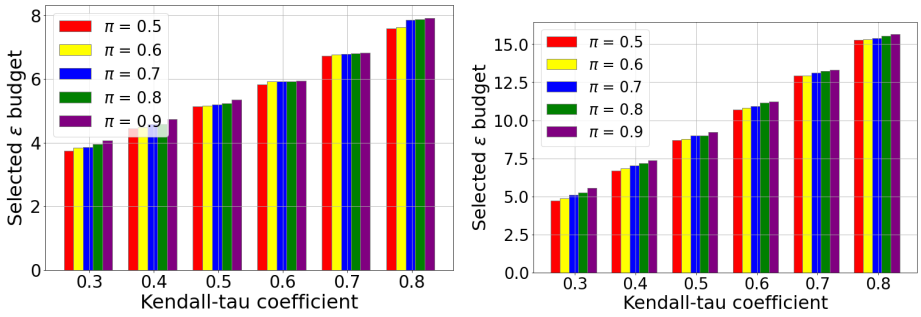
optimization. Hence, this approach should be used carefully (more analysis of the accuracy-efficiency tradeoff in Bayes is given in Figure 5.6). On the other hand, MPass performs multiple passes and evaluates multiple ε_{cur} in each pass by design. Furthermore, the number of ε_{cur} can be low in the first passes, but it needs to be increased in subsequent passes to achieve higher precision. Hence, in total, MPass typically performs a larger number of LDP simulations compared to BS, RBS, and DBS; therefore, it has higher execution time.

Comparing BS, RBS, and DBS, we observe that DBS has the lowest execution time. This shows that DBS's strategy of using a smaller $\mathcal{N}$ in clear-cut scenarios but larger $\mathcal{N}$ in challenging scenarios is effective in reducing the overall execution time, even compared to BS. In contrast, RBS has a higher execution time compared to BS and DBS. This is caused by RBS's strategy of adding or subtracting a *step* size each time the search space is narrowed down. Since the search space narrows down slower in RBS compared to BS and DBS, the overall execution time of RBS is higher.

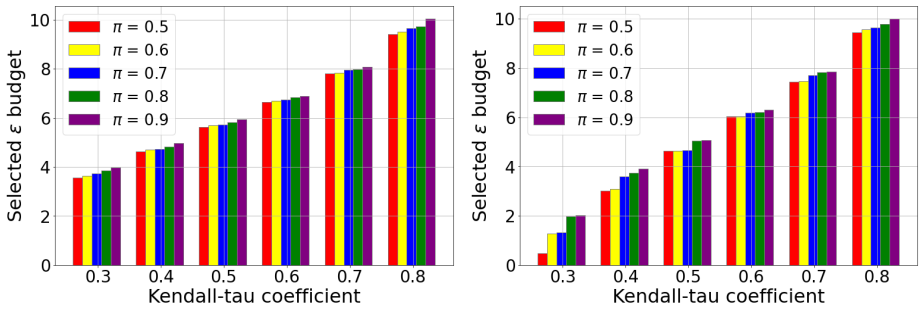
5.1.4 Impacts of the π Parameter

A key component of the optimization problem formulated in Section 3.4 is the probability threshold parameter π. This parameter requires that the utility constraint be satisfied with at least π probability. In Figure 5.5, we vary π between 0.5 and 0.9 and plot the resulting ε budgets that are selected by the BS solution under four different protocols.

As π is increased from 0.5 to 0.9, the utility constraint becomes more challenging to satisfy. We observe from Figure 5.5 that this causes the resulting ε budgets to increase. For example, consider the OUE protocol and the Kendall-tau coefficient = 0.3. The selected ε budgets are approximately 3.6 and 3.7 when $\pi = 0.5$ and 0.6. In contrast, when $\pi = 0.9$, the selected ε budget is 4. Similar changes in ε budgets can be observed in other protocols as well. In many cases, while the increase in ε is noticeable, it is not extremely large (i.e., within 0.5 or 1), which is beneficial for stability. An exception to this observation is the OLH protocol (Figure 5.5d) especially when the Kendall-tau coefficient is small (such as 0.3). In this case, the selected ε budget can increase several folds as π is increased from 0.5 to 0.9.



(a) GRR protocol **(b)** RAPPOR protocol



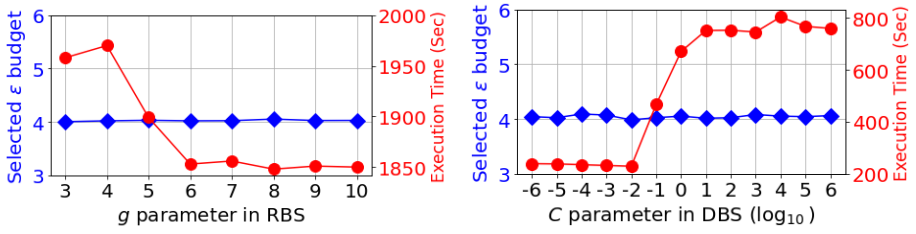
(c) OUE protocol **(d)** OLH protocol

Figure 5.5 Impact of varying the π parameter on BS (Emoji dataset).

5.1.5 Hyperparameter Analysis

Some solution methods in Optimus have internal parameters which affect their convergence and execution times. In this section, we individually analyze the impacts of these parameters using the Emoji dataset, GRR protocol, and Kendall-tau coefficient $\tau = 0.7$. The results are shown in Figure 5.6.

Impact of g in RBS: In RBS, the g parameter is used to determine the *step* size. Smaller g implies larger *step* size, therefore the search space remains larger. Hence, we would expect the execution time to increase. The results in Figure 5.6a confirm this intuition. Execution times are high when $g = 3$ and 4, they decrease as g increases to 5 and 6, and remain similar when g is further increased from 7 to 10. On the other hand, large g implies smaller *step* size, therefore the search space is reduced quickly. This can cause inaccuracy or instability in the convergence of RBS. We indeed observe an example of this when $g = 8$. Usually, the selected ε budget is close to 4; however, it deviates slightly when $g = 8$. As a result, we obtain a tradeoff in the use of g: Small g is good for stability but incurs high execution time, whereas large g may bring instability but execution time is lowered. Overall, based on our experiments, we recommend a value of $g = 5$ or 6 to achieve both stable convergence and low execution time. We also note that the differences between execution times with varying g are not too large, e.g., the execution



(a) g in RBS **(b)** C in DBS

(c) *iter_num* in Bayes **(d)** *init_points* in Bayes

Figure 5.6 Impacts of the different solutions' internal parameters on the selected ε budgets and execution times. Kendall-tau correlation = 0.7, Emoji dataset, and GRR protocol is used in all figures.

time difference between $g = 4$ and $g = 10$ is less than 3 minutes. Hence, if execution time and computation power are not of concern, smaller g values such as $g = 2$, 3, or 4 can be preferred.

Impact of C in DBS: In DBS, the C parameter is used to determine the desired number of simulations, $\mathcal{N}_{desired}$. Smaller C yields fewer simulations, which reduces execution time but can also hurt stability. In Figure 5.6b, we experiment with a large range of C values between 10^{-6} and 10^{6}. We observe from Figure 5.6b that the execution times are low when $C \leq 10^{-2}$, they increase as C is increased from 10^{-2} to 10^{2}, and they remain high after C exceeds 10^{2}. This is an intuitive result since small C yields fewer simulations (hence the execution times are low) whereas high C yields many simulations (hence the execution times increase as C is increased). The reason why execution times remain stable between $10^{-6} \leq C \leq 10^{-2}$ and $10^{2} \leq C \leq 10^{6}$ is due to the way $\mathcal{N}_{desired}$ is computed (Equation 17). When $10^{-6} \leq C \leq 10^{-2}$, the resulting $\mathcal{N}_{desired}$ values become lower than $\mathcal{N}_{min}$, therefore no new simulations need to be performed. When $10^{2} \leq C \leq 10^{6}$, the resulting $\mathcal{N}_{desired}$ values become upper bounded by $\mathcal{N}_{max}$ per Equation 17. Hence, the total simulation numbers do not change in these ranges, resulting in stable execution times. Another observation from Figure 5.6b is that the resulting ε budgets remain close to 4 despite a large range of C values. Nevertheless, small variations in ε between $10^{-5} \leq C \leq 10^{-2}$ should be noted. Overall, the stability of DBS is not too much affected by C, which is beneficial since it shows that DBS is not overly sensitive to its internal parameters.

Impact of *iter_num* and *init_points* in Bayes: Finally, we study the impacts of the *iter_num* and *init_points* parameters in Bayes in Figures 5.6c and 5.6d. In Bayes, *init_points* determines the number of initial data points used in training the initial surrogate model. High *init_points* is beneficial to increase the accuracy of the initial surrogate model, but it also increases the execution time since *init_points* number of function evaluations need to be performed (each function evaluation requires multiple LDP simulations). We observe from Figure 5.6d that the execution times steadily increase as *init_points* is increased from 1 to 50. In addition, when *init_points* is less than 15, we observe that the selected ε budgets may deviate from 4 substantially. This shows that too few *init_points* can be detrimental because it causes the initial surrogate model to be too inaccurate. Hence, we recommend to use $init\_points \geq 20$ in practice.

On the other hand, *iter_num* determines the number of iterations of Bayesian optimization. More iterations are beneficial from an accuracy perspective since the search space is further explored, the surrogate model is updated, and the *history* is enhanced with more samples in each iteration. Yet, more iterations cause higher execution time for the Bayes algorithm. We indeed observe this increase in execution time in Figure 5.6c. We also observe that low *iter_num* such as 1 or 5 can cause inaccuracies, e.g., ε is significantly different from 4. Yet, ε values converge to 4 as $iter\_num \geq 15$. Hence, we recommend to use $iter\_num \geq 15$ or 20 in practice.

5.2 Experiments on CAPRI

5.2.1 Experiment Setup for CAPRI

We implemented CAPRI in Python. In this section, we perform experiments with CAPRI on real-world and synthetic datasets using three LDP protocols, varying constraints, thresholds, TTC amounts, and capacities. The following default parameters are used throughout the experiments: $\varepsilon_{min} = 0.001$, $\varepsilon_{max} = 10$, and $\varphi = 0.01$.

We used AguaH obtained from Kaggle[2] as our real-world dataset. This dataset contains monthly water consumption per client (in cubic meters) from 2009 to 2016 in a city. From the original dataset, we removed those clients whose data contained missing values. In addition, we observed that there were very few instances in which the consumption was above 40 m^3. To alleviate the impact of such anomalous readings, we limited the max consumption to 40 m^3. After all pre-processing, the final dataset used in our experiments contains N = 98,511 clients and TTC = 1,598,489.

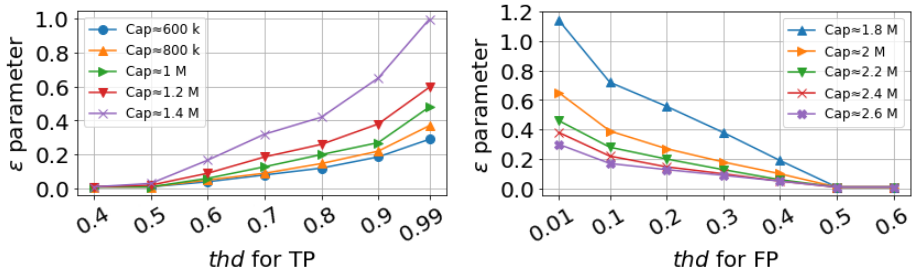
5.2.2 Impact of Thresholds and Capacities

In Figure 5.7, we show the relationship between the optimized (selected) ε budgets versus the TP and FP thresholds τ under varying *cap*. First, we study the TP scenarios (Figures 5.7a and 5.7c). As the TP threshold increases from 0.4 to 0.99, the selected ε budget also increases. This is an intuitive result since increasing threshold implies that Pr[TP] $\geq \tau$ for an increasing τ. Considering that $TTC > cap$, there is an increasing need for $ETC > cap$ to also hold. Hence, the risk of underestimating ETC should be lowered as the threshold increases. This can be achieved by increasing the selected ε budget since higher budgets yield lower estimation error. As a result, we observe that ε is allowed to be low (such as below 0.2 in Figure 5.7a) when the threshold is 0.6; however, it must be increased several folds (as high as 1) when the threshold is strict such as 0.9 or 0.99.

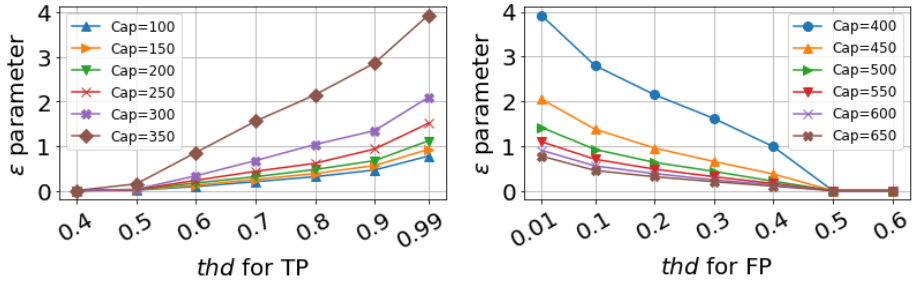
Next, we study the FP scenarios (Figures 5.7b and 5.7d). In these scenarios, as the threshold increases, the goal becomes less strict (more relaxed), e.g., it is easier to achieve Pr[FP] < 0.4 rather than Pr[FP] < 0.01. Consequently, we observe a decrease in the selected ε budget as thresholds increase. For example, according to Figure 5.7b, when cap = 1.8 million, $\varepsilon = 1.2$ should be selected as the optimized budget when τ is as strict as 0.01. However, as τ becomes less strict such as 0.3 and 0.4, the optimized budget can drop to 0.4 and 0.2, because higher estimation error and more FPs can be tolerated.

Finally, we study the impact of varying *cap*. We observe that the impact of *cap* is different with respect to TPs and FPs. In Figures 5.7a and 5.7c, when *cap* increases, the selected ε budget also increases, meaning that estimation

[2]https://www.kaggle.com/datasets/marcomolina/water-consumption-in-a-median-size-city



(a) Real, varying TP threshold **(b)** Real, varying FP threshold



(c) Syn, vary TP threshold **(d)** Syn, vary FP threshold

Figure 5.7 Relationship between optimized ε versus TP and FP thresholds under varying *cap*. Real = real-world AguaH dataset, Syn = synthetic dataset. GRR protocol is used in all figures.

error is less tolerated. The reason for this can be explained as follows. Assume that all parameters and TTC are the same. Since Figures 5.7a and 5.7c are considering TPs, we know that $TTC > cap$. When *cap* is increased, it becomes more difficult to satisfy $ETC > cap$ since ETC must be closer to TTC, i.e., there is less room for ETC to decrease due to negative estimation error. Thus, in Figures 5.7a and 5.7c, there is a positive correlation between *cap* and ε. On the other hand, in Figures 5.7b and 5.7d, since we consider FPs, there is a negative correlation between *cap* and ε. To explain, let us again assume that all parameters and TTC are the same. Since we are considering FPs, we know that $TTC \leq cap$, but $ETC > cap$. If *cap* is increased, ETC needs to exceed an even higher *cap* than before. This means that estimation error shall increase, and estimation error increases when ε decreases. Thus, in FP scenarios, there is a negative correlation between *cap* and ε.

5.2.2.1 Impact of Number of Runs

In Figure 5.8, we examine the impact of the num_runs parameter on the convergence behavior of CAPRI by re-executing the same optimization process with varying num_runs values: 500, 1000, 5000, 15000, 30000 and 50000. We

repeat the experiment with each *num_runs* value 100 times and record the final outputs of CAPRI (the selected ε budgets) in each case. Each ε value that was selected by CAPRI at least once and its count (how many times it was selected out of 100) are reported in Figure 5.8.

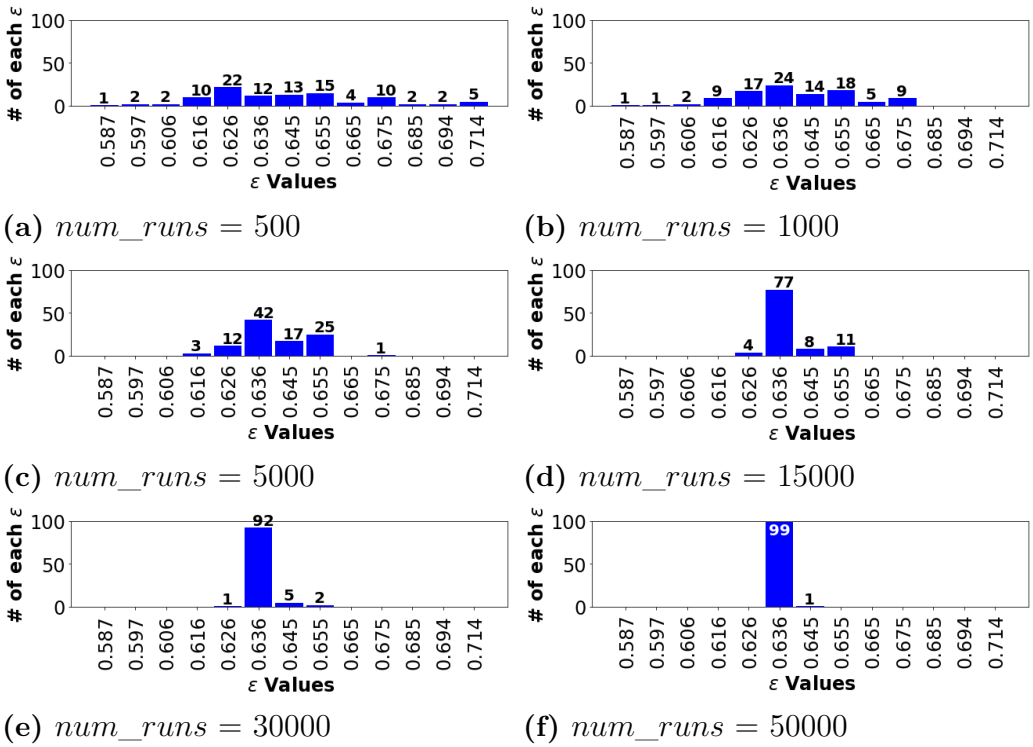


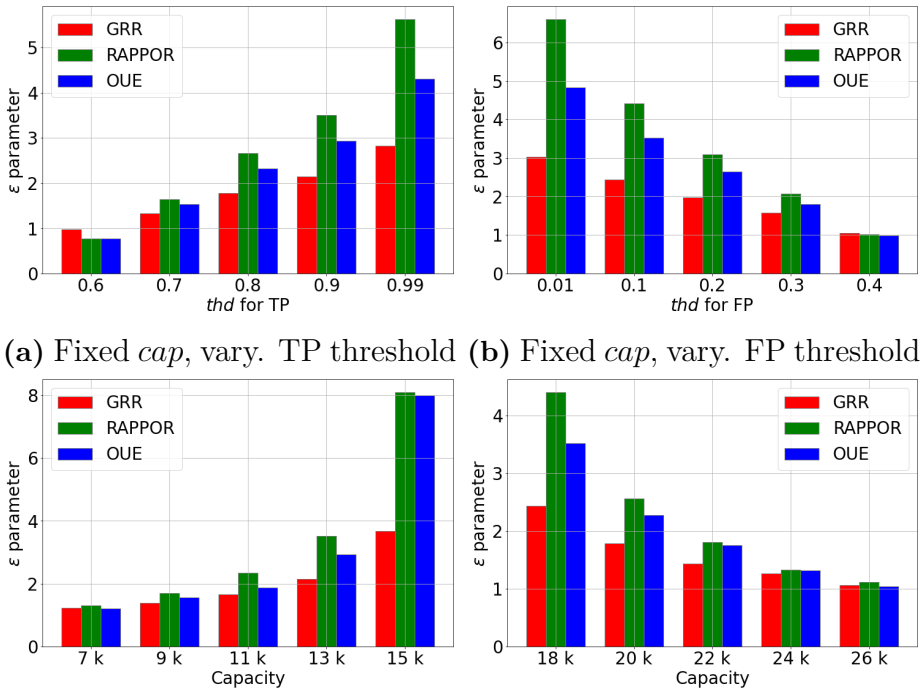
Figure 5.8 Impact of *num_runs* parameter on the convergence behavior of CAPRI.

When *num_runs* = 500, we observe that 13 different ε values varying from 0.587 to 0.714 were selected by CAPRI. There is a fairly even distribution of counts among ε values within the range [0.616, 0.675]. As we increase *num_runs* to 1000, 5000 and 15000, we observe that the uniformity of the distribution drops quickly. Instead, the most dominant ε value (which is ε = 0.636) becomes more and more heavily selected. For high values of *num_runs* such as 30000 and 50000, the most dominant ε value is selected in 92% and 99% of the cases, respectively. Even if the most dominant ε is not selected, it is highly likely that adjacent ε values will be selected. Overall, these results show that when *num_runs* is low, there is higher variance in terms of selected ε: different ε values may be selected and they can originate from a larger range. As *num_runs* increases, the probability of the most dominant ε being selected increases; in addition, even when the most dominant ε is not selected, a close value will be selected. Therefore, we can conclude that when *num_runs* is low such as 500, 1000 or 5000, CAPRI's convergence is not always stable. On the other hand, when *num_runs* is higher such as 30000 or 50000, CAPRI is very likely to converge to the same ε and therefore its convergence is stable.

Hence, higher num_runs is indeed better to remove the impact of randomness in CAPRI and enable more stable convergence.

5.2.2.2 Impact of LDP Protocols

We analyze the impacts of the LDP protocols in Figure 5.9. To improve efficiency and show higher variability, 10% of the real dataset is used in this set of experiments, such that $TTC = 15620$. In Figure 5.9a, we fix $cap = 13000$ in order to have true capacity exceedances. In Figure 5.9b, we fix $cap = 18000$ in order to not have any capacity exceedances (hence, all predicted exceedances will be false positives). In Figures 5.9c and 5.9d, we fix the TP and FP thresholds respectively, and vary the cap values. Note that $TTC > cap$ for all choices of cap in Figure 5.9c (between 7000 - 15000) and $TTC < cap$ for all choices of cap in Figure 5.9d (between 18000 - 26000).



(a) Fixed cap, vary. TP threshold **(b)** Fixed cap, vary. FP threshold

(c) Fixed TP threshold = 0.9 **(d)** Fixed FP threshold = 0.1

Figure 5.9 Comparison of GRR, RAPPOR, OUE protocols in terms of the selected ε budgets under varying cap and varying TP or FP thresholds.

In general, the results show that the selected ε budgets under different LDP protocols can be different. For example, the ε selected for the RAPPOR protocol is usually higher than GRR and OUE. This is because of the different expected estimation errors of the different protocols. For example, OUE typi-

cally provides lower estimation error than RAPPOR Wang et al. (2017). Under the same ε budget, OUE may satisfy a TP requirement but RAPPOR may not. Thus, keeping the TP/FP requirement constant, ε that must be selected for RAPPOR should be higher than OUE so that RAPPOR's estimation error can be similar to OUE's estimation error.

The results in Figure 5.9 can also be used to guide LDP protocol selection for practitioners. For example, consider the requirement: $\Pr[\text{TP}] \geq 0.8$. According to Figure 5.9, GRR is the protocol that enables the usage of lowest ε; the ε budgets that must be used for RAPPOR and OUE are comparatively higher. Hence, from the perspective of maximizing clients' privacy by enforcing the lowest ε possible, the GRR protocol seems to be the best choice. Here, it should be noted that the best choice protocol can change under different settings. For example, OUE is the best choice when TP threshold is 0.6, but GRR is the best choice when TP threshold is 0.8.

6. DISCUSSION

6.1 Discussion and Possible Extensions on CAPRI

6.1.1 Static vs Dynamic Use of CAPRI

As shown in Algorithm 7, CAPRI does not require clients' individual true values but rather population-level aggregate statistics, i.e., it takes as input $X(v)$ for $v \in V$. To learn $X(v)$, CAPRI can make use of publicly available resources or historical aggregate statistics for the client population. If such data is not available, then a small ε budget can be dedicated to performing frequency estimation with LDP so that $X(v)$ can be learned while satisfying ε-LDP. Afterward, the learned $X(v)$ can be fed into Algorithm 7.

Our current implementation of CAPRI assumes the availability of $X(v)$, which makes it a planning tool to determine the optimized ε^* budget for the $X(v)$ available in hand. However, CAPRI can be extended and used in dynamic scenarios as well. For example, say that the data collector (i.e., capacity planner) wishes to perform planning at consecutive timestamps t, $t+1$, $t+2$, and so forth. Let $X_t(v)$ denote the aggregate statistics collected via LDP at timestamp t. Then, the planner can use $X_t(v)$ as input to CAPRI for selecting the budget at time $t+1$. Similarly, the planner can use the resulting $X_{t+1}(v)$ as input to CAPRI for selecting the budget at time $t+2$. This way, CAPRI can be used dynamically and longitudinally, such that the privacy budget that will be used in each future timestamp is determined using aggregate statistics from the previous collection.

6.1.2 Privacy Impact of num_runs Parameter

We would like to highlight that once the aggregate statistics $X(v)$ are available, the rest of CAPRI is performed offline on the server side. In other words, CAPRI does not determine ε^* by interacting with clients or collecting their data multiple times. On the contrary, it performs simulations on the server side using $X(v)$ to simulate the result of "hypothetical" data collections with different ε. Hence, the value of the *num_runs* parameter does not impact the privacy loss of the clients.

6.1.3 Privacy Impacts of Selected ε

Considering that the selected ε can have different impacts on privacy in different contexts, it is important to evaluate the impacts of varying ε in our context. Towards this end, inspired by recent works on the privacy evaluation of LDP protocols Arcolezi et al. (2023); Gadotti et al. (2022); Gursoy et al. (2022), we propose two concrete privacy evaluation metrics for our setting. In both metrics, similar to the assumptions in Gadotti et al. (2022); Gursoy et al. (2022), the adversary is assumed to observe the output of the LDP protocol, and aims to make predictions regarding the client's true value v_i using Bayesian inference. For the implementation of Bayesian inference, we use the techniques from Gursoy et al. (2022). Let $\mathcal{A}$ denote the adversary, v_i denote the client's true consumption value, and o_i denote the LDP protocol output sent from the client to the server. The Bayesian adversary $\mathcal{A}$ predicts the client's true value as:

$$v_i^p = \arg\max_{v \in V} \Pr[v|o_i]$$

Our first metric is the Attack Success Rate (ASR), which measures the ratio of clients whose true consumptions are correctly predicted by $\mathcal{A}$:

$$ASR = \frac{\#\text{ of clients } u_i \in P \text{ such that } v_i = v_i^p}{|P|}$$

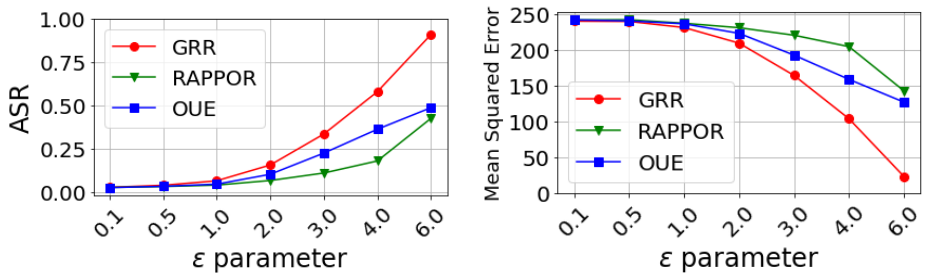


Figure 6.1 ASR and MSE results of various ε values.

Our second metric is Mean Squared Error (MSE), which is measured as:

$$MSE = \frac{\sum_{u_i \in P} (v_i - v_i^p)^2}{|P|}$$

We experimentally find ASR and MSE using the same real-world dataset and experiment setup as in Section 5.2. Figure 6.1 shows the ASR and MSE of different ε values. As expected, as ε increases, privacy is more relaxed and therefore the adversary can make more accurate inferences. Thus, as ε increases, ASR increases and MSE decreases. We also observe that typically, GRR's ASR is higher and MSE is lower compared to other protocols. In contrast, RAPPOR typically has the lowest ASR and highest MSE. Overall, these experiments demonstrate that the privacy impacts of different ε can be measured under various protocols and metrics using the aforementioned Bayesian inference strategy.

We also perform experiments with a third metric (F-score). The goal of this approach is to measure whether the adversary $\mathcal{A}$ can correctly predict if the client's consumption is above (or below) a threshold η. For example, η can be selected as a small value in practice, such that $v_i < \eta$ implies the household has not consumed much water in a given month; therefore, it is likely that the residents are on vacation. We define this attack's true positives, true negatives, false positives, and false negatives as follows. (We use the notation ATP, ATN, AFP, AFN here to differentiate between these notions and the TP, TN, FP, FN notations used in previous sections.)

- ATP: $v_i \leq \eta$ and $v_i^p \leq \eta$
- ATN: $v_i > \eta$ and $v_i^p > \eta$
- AFP: $v_i > \eta$ and $v_i^p \leq \eta$
- AFN: $v_i \leq \eta$ and $v_i^p > \eta$

Then, precision, recall and F-score metrics are defined as:

$$\text{Precision} = \frac{ATP}{ATP + AFP} \quad \text{Recall} = \frac{ATP}{ATP + AFN}$$
$$\text{F-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

We experimentally measure the precision, recall, and F-scores using the same dataset and experiment setup as those used before. We repeat the experiments for different η values: η = 0, 5, 10, 20. The results are given in Figures 6.2, 6.3, 6.4 and 6.5, respectively. We observe that the values of all three metrics increase as ε values increase. The reason why precision, recall and F-score are low in case of low ε is because inferring v_i^p is more difficult, hence ATPs tend to be low. Low ATPs negatively affect both precision and recall. Another interesting remark is that as η increases; precision, recall and F-scores increase, especially in case of lower ε values. This is because when η is small (such as η = 0), not only are there fewer predictions that satisfy $v_i^p \leq \eta$, but also there are fewer true values that satisfy $v_i \leq \eta$. Hence, by nature, we have fewer ATPs and higher AFPs and AFNs when η is small. Finally, the comparison between protocols (GRR vs RAPPOR vs OUE) in Figures 6.2-6.5 generally agree with the ASR and MSE results in Figure 6.1. Precision, recall and F-scores of GRR are generally highest, followed by OUE, and finally by RAPPOR.

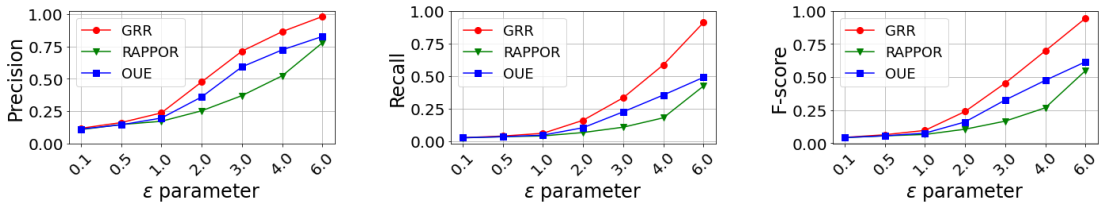


Figure 6.2 Precision, recall and F-score results of various ε values (η = 0).

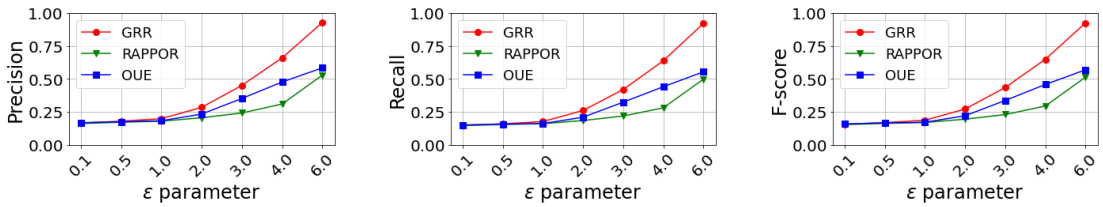


Figure 6.3 Precision, recall and F-score results of various ε values (η = 5).

6.1.4 Frequency Estimation vs Mean Estimation

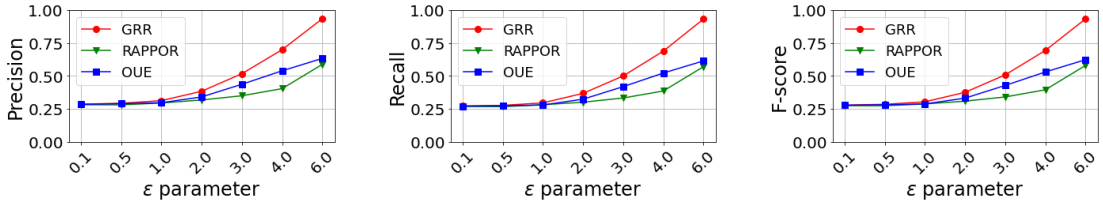


Figure 6.4 Precision, recall and F-score results of various ε values ($\eta = 10$).

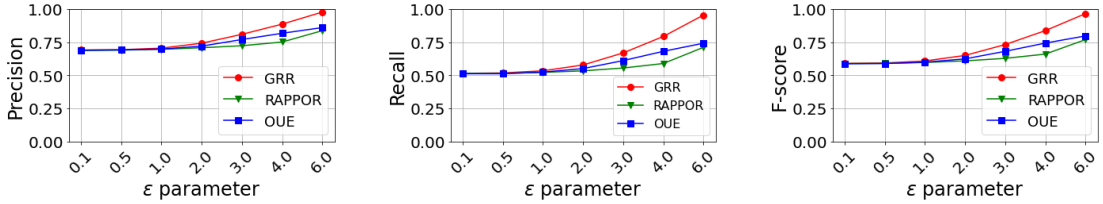


Figure 6.5 Precision, recall and F-score results of various ε values ($\eta = 20$).

As shown in Section 3.6 and Algorithm 9, to compute ETC, we first find frequency estimates $\bar{X}(v)$ for all $v \in V$ and then calculate: $ETC \leftarrow \sum_{v \in V} v \cdot \bar{X}(v)$. Instead of this frequency estimation-based approach, it is also possible to use a mean estimation-based approach. For this, an LDP mean estimation protocol can be integrated into CAPRI (such as Wang et al. (2019)), in which the server first estimates the mean consumption of the population under LDP and then multiplies the mean by $|P|$ to obtain ETC.

The frequency estimation-based (FE) and mean estimation-based (ME) approaches have advantages and disadvantages. FE has the following advantages. First, although we used capacity constraints based on TP, FP, TN and FN in this thesis, there can also be other constraints, such as: "large v values should occur fewer than a certain number of times". FE enables such constraints to be enforced since it estimates the frequency of each v. However, such a constraint cannot be used under ME since frequencies are not estimated. Thus, FE has the advantage of accommodating a wider selection of constraint types. Second, there are a larger number of FE protocols compared to ME protocols in the LDP literature. Since CAPRI's implementation supports arbitrary FE protocols, new FE protocols can easily be integrated into CAPRI. We note that ME protocols can also be integrated into CAPRI by modifying Algorithm 3 in future work. Third, the use of FE enables the server to learn values' frequency distribution under LDP protection. This can be used as a side benefit by the server in downstream tasks and applications.

On the other hand, ME also has some advantages. First, FE requires the server to estimate the frequency of each $v \in V$, so there is an $\mathcal{O}(v)$ cost. When V is large, this cost may become significant. Since ME does not incur this cost, it can be more time-efficient and scalable especially when V is large. Second, if

frequencies are not needed (e.g., they are not going to be used in downstream tasks or applications), ME saves time and memory by not computing and storing them. Overall, considering the advantages and disadvantages of FE and ME, we chose to implement an FE-based approach in CAPRI since our computational resources were sufficient to handle our datasets and V comfortably.

6.1.5 Impact of Outliers

As mentioned before, after obtaining the AguaH dataset from Kaggle, we preprocessed it to limit the max consumption by 40. This is because we observed that although consumption values higher than 300 exist when outliers are not removed, 95% of the consumptions are between 0-40. Consumptions > 40 are rare and consumptions > 80 are exceedingly rare (e.g., between 0-2 clients in a population of tens of thousands).

Including outlier consumption values in the computation increases the domain size V, which has two main impacts on CAPRI. First, it causes LDP protocols' client-side perturbation and server-side estimation procedures to have higher execution times, as well as CAPRI's remaining computations to also have higher execution times. Second, due to the noise added by LDP, rare consumption values would be estimated as having non-zero frequencies (i.e., erroneous $\bar{X}(v)$ estimations) which would increase the total amount of error. The increased error may potentially have a negative impact on utility and the search for ε.

In short, outlier removal provides advantages in terms of improved efficiency and utility. We note that if outliers are not removed, CAPRI would still execute properly, and it would be expected to provide correct results since LDP protocols are unbiased. However, its execution times would have been longer and its convergence stability during the search for ε could have been lower.

The next question is how outliers can be compensated in CAPRI. To compensate for the efficiency aspect (increased execution times), we note that CAPRI's execution time is directly dependent on the execution time of the underlying LDP protocol in use. Some LDP protocols' execution times are inherently negatively affected by large V, e.g., RAPPOR and OUE's client-side perturbations rely on bitvectors with lengths equal to $|V|$. In contrast, other

LDP protocols, such as GRR, are less dependent on V, and therefore they can remain fast despite larger V. Thus, to compensate for the efficiency aspect, we recommend the use of protocols that are faster and do not have execution times dependent on the size of V. To compensate for the utility aspect, we note that CAPRI's main algorithm (Algorithm 1) utilizes a strict decision rule when determining the search space in each iteration (lines 16-19), i.e., if the constraint is satisfied then search for smaller ε; if the constraint is not satisfied then search for larger ε. Since increased utility loss may cause potentially erroneous decisions to be made at this point, this decision rule may be relaxed (weakened) to compensate for the increased error. For example, line 17 can be revised as: $\varepsilon_{max} \leftarrow \varepsilon + \delta$ and line 19 can be revised as: $\varepsilon_{min} \leftarrow \varepsilon - \delta$, where δ is a relaxation parameter. This change would make the optimization algorithm more error-tolerant by eliminating a smaller portion of the search space in each iteration.

6.1.6 CAPRI vs Other Optimization Approaches

We experimentally compared CAPRI with other optimization approaches to confirm its benefits. To do so, we designed the following experiment. We enlarged our synthetic dataset to contain 100.000 clients by repeating the values in the dataset (this enlargement was done so that efficiency comparison between CAPRI and other optimization approaches yields meaningful results). We experimentally compared CAPRI with CAPRI-R and Bayesian optimization:

- CAPRI-R is a relaxed variant of CAPRI, in which Algorithm 1 is modified by revising line 17 as: $\varepsilon_{max} \leftarrow \varepsilon + \delta$ and line 19 as: $\varepsilon_{min} \leftarrow \varepsilon - \delta$. This is the relaxation we described at the end of the previous section.
- Bayesian optimization is a prominent approach to black-box function optimization. We used Bayesian optimization to solve the optimization problem given in Section 3.7.

Comparisons between CAPRI, CAPRI-R and Bayesian optimization were done to analyze: (i) the consistency of selecting the optimized ε, (ii) execution times. Results regarding ε are given in Figure 6.6 and results regarding execution times are given in Figure 6.7.

It can be observed from Figure 6.6 that the selected ε budgets are similar in all

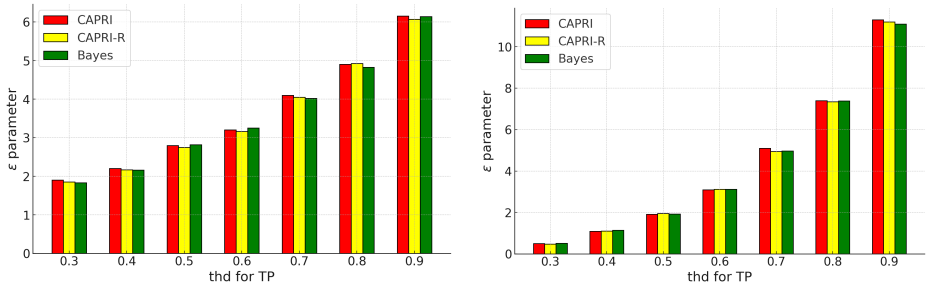


Figure 6.6 Comparison of CAPRI, CAPRI-R, and Bayesian optimization in terms of the selected ε budgets under varying TP thresholds. GRR protocol on the left, RAPPOR protocol on the right.

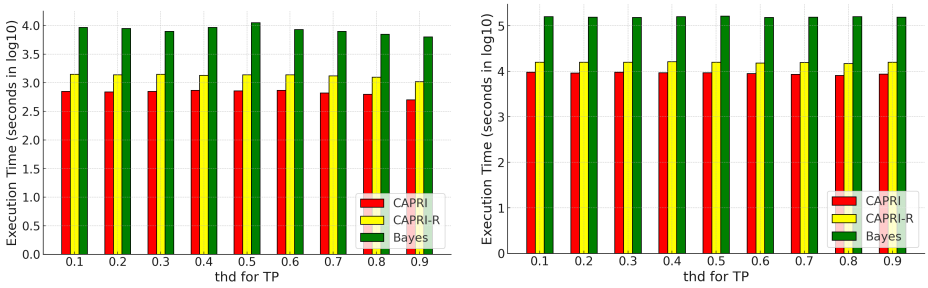


Figure 6.7 Comparison of CAPRI, CAPRI-R, and Bayesian optimization in terms of execution time under varying TP thresholds. GRR protocol on the left, RAPPOR protocol on the right.

three approaches, i.e., they are consistent with one another. Yet, the execution time of CAPRI is lower than CAPRI-R and much lower than Bayesian optimization according to Figure 6.7. Hence, we conclude that CAPRI succeeds in selecting the optimized budget value faster than the other optimization approaches. Producing a consistent result in a faster manner compared to other optimization approaches confirms the benefit of CAPRI.

7. CONCLUSION

In this dissertation, we present two software systems: one designed for general ε budget selection in local differential privacy (LDP) under utility constraints, and another specifically developed for optimized budget selection and capacity planning within the LDP framework. At first, we proposed and developed Optimus, an optimization-based approach for ε budget selection in LDP under utility constraints. Given a utility constraint, we formulated the problem of budget selection as an optimization problem and proposed five solution methods: BS, RBS, DBS, MPass, and Bayes. We experimentally compared these methods using multiple LDP protocols, datasets, and utility constraints. We found that the budgets selected by the different methods are generally consistent and in agreement. DBS is the fastest whereas MPass and Bayes have substantially longer execution times. Overall, RBS offers a good tradeoff between efficiency and consistency, in addition to addressing BS's occasional incorrect search space eliminations caused by LDP randomness.

Secondly, we proposed a novel software system called CAPRI for optimized budget selection and capacity planning under LDP. To the best of our knowledge, CAPRI is the first system to enable optimized budget selection for capacity planning under LDP. CAPRI aims to maximize clients' privacy while a desired utility constraint (e.g., TP, FP, TN, FN constraint) is met. Towards this end, CAPRI utilizes an optimization-based problem formulation and a novel solution that relies on LDP simulations. CAPRI's solution is validated both theoretically by comparing with upper bounds derived from Chebyshev's inequality, as well as experimentally using multiple datasets, capacities, constraints, and LDP protocols. Furthermore, multiple adversarial approaches are proposed for evaluating the impacts of the chosen ε values on clients' privacy in practice. In future, these systems can be implemented as a web-based graphical user interface (GUI) in order to make them extensible and easy-to-use interfaces for non-experts.

BIBLIOGRAPHY

(2020). Apple – learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf.

Anderson, B., Lin, S., Newing, A., Bahaj, A., & James, P. (2017). Electricity consumption and household characteristics: Implications for census-taking in a smart metered future. *Computers, Environment and Urban Systems*, *63*, 58–67.

Arachchige, P. C. M., Bertok, P., Khalil, I., Liu, D., Camtepe, S., & Atiquzzaman, M. (2019). Local differential privacy for deep learning. *IEEE Internet of Things Journal*, *7*(7), 5827–5842.

Arcolezi, H. H., Couchot, J.-F., Al Bouna, B., & Xiao, X. (2021). Random sampling plus fake data: Multidimensional frequency estimates with local differential privacy. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, (pp. 47–57).

Arcolezi, H. H., Couchot, J.-F., Al Bouna, B., & Xiao, X. (2022). Improving the utility of locally differentially private protocols for longitudinal and multidimensional frequency estimates. *Digital Communications and Networks*.

Arcolezi, H. H. & Gambs, S. (2024). Revealing the true cost of locally differentially private protocols: An auditing perspective. *Proceedings on Privacy Enhancing Technologies*, *2024*(4), 123–141.

Arcolezi, H. H., Gambs, S., Couchot, J.-F., & Palamidessi, C. (2023). On the risks of collecting multidimensional data under local differential privacy. *Proceedings of the VLDB Endowment*, *16*(5), 1126–1139.

Bao, E., Yang, Y., Xiao, X., & Ding, B. (2021). Cgm: an enhanced mechanism for streaming data collection with local differential privacy. *Proceedings of the VLDB Endowment*, *14*(11), 2258–2270.

Bassily, R. & Smith, A. (2015). Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, (pp. 127–135).

Beckel, C., Sadamori, L., & Santini, S. (2013). Automatic socio-economic classification of households using electricity consumption data. *Proceedings of the 4th International Conference on Future Energy Systems*.

Bullek, B., Garboski, S., Mir, D. J., & Peck, E. M. (2017). Towards understanding differential privacy: When do people trust randomized response technique? In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, (pp. 3833–3837).

Chen, K.-C., Yu, C.-M., Tai, B.-C., Li, S.-C., Tsou, Y.-T., Huang, Y., & Lin, C.-M. (2017). Data-driven approach for evaluating risk of disclosure and utility in differentially private data release. In *IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, (pp. 1130–1137). IEEE.

Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., & Wang, T. (2018). Privacy at scale: Local differential privacy in practice. In *Proceedings of*

the 2018 International Conference on Management of Data, (pp. 1655–1658).

Cormode, G., Kulkarni, T., & Srivastava, D. (2018). Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, (pp. 131–146).

Cormode, G., Maddock, S., & Maple, C. (2021). Frequency estimation under local differential privacy. *Proceedings of the VLDB Endowment*, *14*(11), 2046–2058.

Cummings, R., Kaptchuk, G., & Redmiles, E. M. (2021). I need a better description: An investigation into user expectations for differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 3037–3052).

Czétány, L., Vámos, V., Horváth, M., Szalay, Z., Mota-Babiloni, A., Deme-Bélafi, Z., & Csoknyai, T. (2021). Development of electricity consumption profiles of residential buildings based on smart meter data clustering. *Energy and Buildings*, *252*, 111376.

da Costa Filho, J. S. & Machado, J. C. (2023). Felip: A local differentially private approach to frequency estimation on multidimensional datasets. In *EDBT*, (pp. 671–683).

Ding, B., Kulkarni, J., & Yekhanin, S. (2017). Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, *30*.

Du, Y., Hu, Y., Zhang, Z., Fang, Z., Chen, L., Zheng, B., & Gao, Y. (2023). Ldptrace: Locally differentially private trajectory synthesis. *Proceedings of the VLDB Endowment*, *16*(8), 1897–1909.

Duan, J., Ye, Q., & Hu, H. (2022). Utility analysis and enhancement of ldp mechanisms in high-dimensional space. In *38th IEEE International Conference on Data Engineering (ICDE)*, (pp. 407–419).

Duan, J., Ye, Q., Hu, H., & Sun, X. (2024). Ldptube: Theoretical utility benchmark and enhancement for ldp mechanisms in high-dimensional space. *IEEE Transactions on Knowledge and Data Engineering*.

Eibl, G., Burkhart, S., & Engel, D. (2019). Insights into unsupervised holiday detection from low-resolution smart metering data. In *International Conference on Information Systems Security and Privacy (ICISSP)*, (pp. 281–302). Springer.

Eibl, G. & Engel, D. (2014). Influence of data granularity on smart meter privacy. *IEEE Transactions on Smart Grid*, *6*(2), 930–939.

Erlingsson, Ú., Pihur, V., & Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1054–1067). ACM.

Franzen, D., Nuñez von Voigt, S., Sörries, P., Tschorsch, F., & Müller-Birn, C. (2022). Am i private and if so, how many? communicating privacy guarantees of differential privacy with risk communication formats. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1125–1139).

Fu, Y., Ye, Q., Du, R., & Hu, H. (2023). Collecting multi-type and correlation-constrained streaming sensor data with local differential privacy. *ACM Transactions on Sensor Networks*.

Gadotti, A., Houssiau, F., Annamalai, M. S. M. S., & de Montjoye, Y.-A. (2022). Pool inference attacks on local differential privacy: Quantifying the privacy guarantees of apple's count mean sketch in practice. In *31st USENIX Security Symposium (USENIX Security 22)*, (pp. 501–518).

Gai, N., Xue, K., Zhu, B., Yang, J., Liu, J., & He, D. (2022). An efficient data aggregation scheme with local differential privacy in smart grid. *Digital Communications and Networks*, *8*(3), 333–342.

Gao, W. & Zhou, S. (2023). Privacy-preserving for dynamic real-time published data streams based on local differential privacy. *IEEE Internet of Things Journal*.

Gough, M. B., Santos, S. F., AlSkaif, T., Javadi, M. S., Castro, R., & Catalão, J. P. (2021). Preserving privacy of smart meter data in a smart grid environment. *IEEE Transactions on Industrial Informatics*, *18*(1), 707–718.

Gursoy, M. E. (2024). Longitudinal attacks against iterative data collection with local differential privacy. *Turkish Journal of Electrical Engineering and Computer Sciences*, *32*(1), 198–218.

Gursoy, M. E., Liu, L., Chow, K.-H., Truex, S., & Wei, W. (2022). An adversarial approach to protocol analysis and selection in local differential privacy. *IEEE Transactions on Information Forensics and Security*, *17*, 1785–1799.

Gursoy, M. E., Tamersoy, A., Truex, S., Wei, W., & Liu, L. (2019). Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing*, *18*(5), 2365–2378.

Hong, D., Jung, W., & Shim, K. (2022). Collecting geospatial data under local differential privacy with improving frequency estimation. *IEEE Transactions on Knowledge and Data Engineering*.

Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B. C., & Roth, A. (2014). Differential privacy: An economic method for choosing epsilon. In *IEEE 27th Computer Security Foundations Symposium*, (pp. 398–410). IEEE.

Huang, Y., Xue, K., Zhu, B., Wei, D. S., Sun, Q., & Lu, J. (2024). Joint distribution analysis for set-valued data with local differential privacy. *IEEE Transactions on Information Forensics and Security*.

Islam, S., Badsha, S., Sengupta, S., Khalil, I., & Atiquzzaman, M. (2022). An intelligent privacy preservation scheme for ev charging infrastructure. *IEEE Transactions on Industrial Informatics*, *19*(2), 1238–1247.

John, M. F. S., Denker, G., Laud, P., Martiny, K., Pankova, A., & Pavlovic, D. (2021). Decision support for sharing data using differential privacy. In *2021 IEEE Symposium on Visualization for Cyber Security (VizSec)*, (pp. 26–35). IEEE.

Karegar, F., Alaqra, A. S., & Fischer-Hübner, S. (2022). Exploring {User-Suitable} metaphors for differentially private data analyses. In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, (pp. 175–193).

Kazan, Z. & Reiter, J. P. (2023). Prior-itizing privacy: A bayesian approach to setting the privacy budget in differential privacy. *arXiv preprint*

arXiv:2306.13214.

Kleiminger, W., Beckel, C., & Santini, S. (2015). Household occupancy monitoring using electricity meters. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, (pp. 975–986).

Lee, J. & Clifton, C. (2011). How much is enough? choosing ε for differential privacy. In *14th International Conference on Information Security (ISC)*, (pp. 325–340). Springer.

Li, X., Liu, W., Lou, J., Hong, Y., Zhang, L., Qin, Z., & Ren, K. (2024). Local differentially private heavy hitter detection in data streams with bounded memory. *Proceedings of the ACM on Management of Data*, *2*(1), 1–27.

Liu, G., Tang, P., Hu, C., Jin, C., Guo, S., Stoyanovich, J., Teubner, J., Mamoulis, N., Pitoura, E., & Mühlig, J. (2023). Multi-dimensional data publishing with local differential privacy. In *EDBT*, (pp. 183–194).

Marks, J., Montano, B., Chong, J., Raavi, M., Islam, R., Cerny, T., & Shin, D. (2021). Differential privacy applied to smart meters: a mapping study. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, (pp. 761–770).

Mockus, J. (1974). On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, (pp. 400–404).

Murakami, T. & Takahashi, K. (2020). Toward evaluating re-identification risks in the local privacy model. *arXiv preprint arXiv:2010.08238*.

Nanayakkara, P., Smart, M. A., Cummings, R., Kaptchuk, G., & Redmiles, E. (2023). What are the chances? explaining the epsilon parameter in differential privacy. *arXiv preprint arXiv:2303.00738*.

Ou, L., Qin, Z., Liao, S., Li, T., & Zhang, D. (2020). Singular spectrum analysis for local differential privacy of classifications in the smart grid. *IEEE Internet of Things Journal*, *7*(6), 5246–5255.

Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., & Ren, K. (2016). Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 192–203).

Radovanovic, D., Unterweger, A., Eibl, G., Engel, D., & Reichl, J. (2022). How unique is weekly smart meter data? *Energy Informatics*, *5*.

Razavi, E., Arefi, A., Smith, D., Ledwich, G., Nourbakhsh, G., & Minakshi, M. (2022). Privacy-preserved framework for short-term probabilistic net energy forecasting. *IEEE Transactions on Industrial Informatics*.

Ren, X., Shi, L., Yu, W., Yang, S., Zhao, C., & Xu, Z. (2022). Ldp-ids: Local differential privacy for infinite data streams. In *Proceedings of the 2022 International Conference on Management of Data*, (pp. 1064–1077).

Ren, X., Yu, C.-M., Yu, W., Yang, S., Yang, X., McCann, J. A., & Philip, S. Y. (2018). Lopub: high-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, *13*(9), 2151–2166.

Seyedkazemi, S., Gursoy, M. E., & Saygin, Y. (2024). Capacity planning under local differential privacy with optimized budget selection. *IEEE Transactions on Industrial Informatics*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, *25*.

Tire, E. & Gursoy, M. E. (2024). Answering spatial density queries under local differential privacy. *IEEE Internet of Things Journal*.

Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., & Wei, W. (2020). Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, (pp. 61–66).

Vatsalan, D., Bhaskar, R., & Kaafar, M. A. (2022). Local differentially private fuzzy counting in stream data using probabilistic data structures. *IEEE Transactions on Knowledge and Data Engineering*.

Wang, B., Chen, Y., Jiang, H., & Zhao, Z. (2023). Ppefl: Privacy-preserving edge federated learning with local differential privacy. *IEEE Internet of Things Journal*, *10*(17), 15488–15500.

Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S. C., Shin, H., Shin, J., & Yu, G. (2019). Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, (pp. 638–649). IEEE.

Wang, S., Li, Y., Zhong, Y., Chen, K., Wang, X., Zhou, Z., Peng, F., Qian, Y., Du, J., & Yang, W. (2023). Locally private set-valued data analyses: Distribution and heavy hitters estimation. *IEEE Transactions on Mobile Computing*.

Wang, T., Blocki, J., Li, N., & Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, (pp. 729–745).

Wang, T., Chen, J. Q., Zhang, Z., Su, D., Cheng, Y., Li, Z., Li, N., & Jha, S. Continuous release of data streams under both centralized and local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 1237–1253).

Wang, T., Li, N., & Jha, S. (2018). Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, (pp. 127–143). IEEE.

Wang, T., Li, N., & Jha, S. (2021). Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, *18*(02), 982–993.

Wang, T., Yang, X., Ren, X., Yu, W., & Yang, S. (2019). Locally private high-dimensional crowdsourced data release based on copula functions. *IEEE Transactions on Services Computing*, *15*(2).

Wang, T., Zhao, J., Hu, Z., Yang, X., Ren, X., & Lam, K.-Y. (2021). Local differential privacy for data collection and analysis. *Neurocomputing*, *426*, 114–133.

Wang, X., Jin, Y., Schmitt, S., & Olhofer, M. (2023). Recent advances in bayesian optimization. *ACM Computing Surveys*, *55*(13s), 1–36.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, *60*(309), 63–69.

Xiong, A., Wang, T., Li, N., & Jha, S. (2020). Towards effective differential pri-

vacy communication for users' data sharing decision and comprehension. In *2020 IEEE Symposium on Security and Privacy (SP)*, (pp. 392–410). IEEE.

Xiong, A., Wu, C., Wang, T., Proctor, R. W., Blocki, J., Li, N., & Jha, S. (2022). Using illustrations to communicate differential privacy trust models: An investigation of users' comprehension, perception, and data sharing decision. *arXiv preprint arXiv:2202.10014*.

Yang, M., Guo, T., Zhu, T., Tjuawinata, I., Zhao, J., & Lam, K.-Y. (2023). Local differential privacy and its applications: A comprehensive survey. *Computer Standards & Interfaces*, 103827.

Zhang, K., Ni, J., Yang, K., Liang, X., Ren, J., & Shen, X. S. (2017). Security and privacy in smart city applications: Challenges and solutions. *IEEE Communications Magazine*, *55*(1), 122–129.

Zhang, Z., Wang, T., Li, N., He, S., & Chen, J. (2018). Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, (pp. 212–229).

Zheng, S., Apthorpe, N., Chetty, M., & Feamster, N. (2018). User perceptions of smart home iot privacy. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 1–20.

Zhu, Y., Cao, Y., Xue, Q., Wu, Q., & Zhang, Y. (2023). Heavy hitter identification over large-domain set-valued data with local differential privacy. *IEEE Transactions on Information Forensics and Security*.