Differentially Private Simple Genetic Algorithms

Thomas Humphries
University of Waterloo
thomas.humphries@uwaterloo.ca

Florian Kerschbaum
University of Waterloo
florian.kerschbaum@uwaterloo.ca

ABSTRACT

The differentially private (DP) selection problem is a fundamental building block in the private literature that is commonly solved with the exponential mechanism. It is well known that efficiency is the major drawback of the exponential mechanism, as the utility function must be computed for all elements in the domain. Genetic algorithms (GAs) use the principles of evolution in nature to efficiently search through large domains and select the best candidate. We observe that GAs have many appealing properties for DP Selection. These include being robust to noisy objectives, placing no restriction on the utility function, and efficient runtime for large domains. However, prior work investigating DP GAs has shown poor utility in practice and often gives the highest utility when *zero generations* are conducted (indicating that GA operations are not beneficial under DP). This work provides a new DPGA based on the simple GA that addresses the weaknesses of prior solutions. We reduce the destructive nature of previous GA operators and utilize several techniques to reduce the noise from DP. Our modifications allow us to utilize the GA operators over multiple generations (under DP) and improve the GA's overall utility over zero generation techniques. Our work shows that private GAs are competitive with state-of-the-art general and problem-specific solutions to the DP selection problem, with runtime sublinear in the domain size.

KEYWORDS

genetic algorithms, differential privacy

1 INTRODUCTION

A fundamental problem in data science is selecting a candidate from a set of items that maximizes some objective function over the dataset. Genetic algorithms (GAs) use the principles of evolution in nature to search through large domains and find these optimal candidates efficiently. For example, one might want to choose the most representative features [2] or find the best ML model parameters [43] for a given dataset. However, these datasets often contain sensitive information, such as health care records. Thus, protecting the participants' privacy is crucial. An increasingly popular notion for protecting the privacy of individuals while allowing the computation of aggregate statistics is differential privacy (DP). Differential privacy guarantees that the output of an algorithm is approximately the same, regardless of the participation of any single user. The privacy parameter ϵ defines how similar the outputs must be and determines the trade-off between privacy and utility. The intuitive guarantee, along with a tunable privacy parameter, has led to wide adoption of DP by organizations such as Google [20], Microsoft [12], Apple [11], and the U.S. Census Bureau [38].

We focus on the problem of differentially private selection, which involves selecting an item from a set of candidates that *approximately* maximizes a given objective function, *while preserving differential privacy.* The most popular mechanism for solving this problem is the exponential mechanism [40]. The exponential mechanism can be used as a standalone algorithm for problems like heavy hitter or median [18, 37] calculations. A recent line of work has found that with the right utility function, the exponential mechanism can outperform state-of-the-art solutions for simple problems like mean estimation [6, 23, 44]. The exponential mechanism also acts as a building block for more sophisticated algorithms such as principal component analysis [8], synthetic data generation [53], and empirical risk minimization [5]. A significant weakness of the exponential mechanism is scalability. The exponential mechanism requires that the utility function is evaluated for all possible candidates in the domain. This results in exponential runtime for many problems, such as ML model fitting, k-medians clustering, and mean estimation. One possible solution is the sub-sampled exponential mechanism; however, this technique depends heavily on the utility of the uniform sampling [33]. Other solutions reduce the candidate space in a data-dependent way; however, to maintain privacy, this typically requires a non-negligible probability the algorithm will return no output [17, 44].

Genetic algorithms use meta-heuristics to efficiently search through large domains and find high utility solutions. Like the exponential mechanism, GAs require minimal assumptions about the objective function and can be applied to a wide range of problems. The runtime of the GA depends on hyperparameters such as the number of generations which are typically sublinear in the domain size. GAs also tend to be robust to noisy objective functions [51]. Despite these appealing properties, there have been no promising experimental results applying GAs to the DP selection problem. The fundamental challenge of adding DP to the GA is that each selection from the population incurs a privacy cost that is deducted from the total privacy budget ϵ. This induces a trade-off; the more selections we make, the noisier each selection becomes in order to preserve privacy.

Zhang et al. conducted the first and only investigation into the concept of DP GAs with their algorithm PrivGene [54]. While PrivGene lays an interesting theoretical foundation, it exhibits poor utility in practice [50]. Su et al. speculate that the poor performance is due to the destructive nature of the crossover operator and the fact that GAs need a large number of iterations to converge (each of which consumes privacy budget) [50]. Additionally, we observe that when using the PrivGene algorithm, the highest utility is given when *zero* generations are conducted. That is, PrivGene often uses all of the privacy budget in a single (higher utility) selection, rather than spreading the privacy budget over multiple generations and

Proceedings on Privacy Enhancing Technologies 2023(4), 540–558
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2023-0124

utilizing the GA operations. This suggests that, using current techniques, evolutionary operators are not helpful in the private setting.

In this work, we propose a new DPGA based on the classic *simple GA* [24]. Our approach builds on some of the ideas in PrivGene, such as using truncation selection in conjunction with the exponential mechanism to ensure DP. However, we make several important changes to overcome the drawbacks of PrivGene. First, we prevent the crossover operator from destroying good candidates by using *elitism*, which protects the best candidates by copying them directly to the new population. By using the more popular simple GA, we introduce standard hyperparameters that control the impact of both crossover and mutation operations. Finally, we employ a number of techniques to reduce the noise from DP in order to make more generations feasible while maintaining sufficient selection pressure. After designing our GA, we empirically evaluate the utility of our solution on two example problems: logistic regression (convex) and k-medians (non-convex). We investigate the effect of DP on the hyperparameters of the simple GA and suggest good default values that generalize across the datasets and problems we considered. In particular, we find that our new DPGA performs best with a higher number of generations (always more than one for $\epsilon \in [0.1, 1]$). That is, GA operations *can* be utilized under DP and indeed help find better solutions than applying all the privacy budget in one shot.

We also evaluate our DPGA against both general and problem-specific solutions to the DP selection problems we consider. We find that our DPGA consistently achieves better utility than the sub-sampled exponential mechanism for $\epsilon > 0.32$. Our DPGA is competitive in utility or efficiency when compared against state-of-the-art problem-specific techniques in private logistic regression and k-medians. To summarize, we show that despite previous efforts, GA operations *can* be utilized under the constraints of DP and tend to result in higher utility than previously shown. Furthermore, we show that DP GAs provide a competitive solution to the more general DP selection problem that is sublinear in the size of the candidate space.

2 PRELIMINARIES

2.1 Genetic Algorithms

The Genetic Algorithm (GA) is a metaheuristic that mimics the evolutionary concept of natural selection [22, 24]. The idea was first introduced by John Holland in the 1960s and has since been developed extensively [27]. The foundations of the GA are very simple probabilistic operations, that when applied carefully and repeatedly, enable an effective search through complex solution spaces. To use a GA, we require an encoding of possible solutions, which we call chromosomes, and a utility function. The utility function takes as input a chromosome (encoded solution vector) and returns a real number representing the effectiveness of the chromosome at solving the problem. To guide the search, a GA evaluates this utility function directly and does not require derivatives or any auxiliary knowledge. As such, there are minimal restrictions on the types of utility functions that can be considered.

2.2 The Simple GA

There are many variants of the GA; we focus on the simple GA as defined by Mitchell [42] for this work. The simple GA consists of four main operations: initialization, selection, crossover, and mutation. There are many possible versions of each of these operations. We will describe the most simple example of each. In Algorithm 2.1, we lay out the basic procedure of the simple GA and how each of the four main operations are invoked. As input, the algorithm takes a utility function and a set of four hyperparameters which we will describe as they are used.

The first step in the algorithm (line 1) is to initialize the population with N_p chromosomes. Most commonly, these chromosomes are sampled uniformly at random from the domain of possible chromosomes. We then enter the algorithm's main loop (line 2). Each iteration of this loop represents a generation. For simplicity, we assume a fixed number of generations (N_g). In each generation, the simple GA creates an entirely new population based on the current population. We denote this new population $\mathcal{P}'$ and build this population using the while loop in line 4.

To create the new population, we repeatedly use the remaining three main operators: selection, crossover, and mutation. First, using the selection operator, we select two parent chromosomes x_1 and x_2 (line 5). The most common way to do this is Goldberg's roulette wheel or fitness proportionate selection, which randomly samples parents weighted linearly by their utility. Next (line 6), we use the p_c hyperparameter to decide if we should invoke the crossover operator (line 7) or simply return the parents themselves (line 9). The simplest crossover operator is the single-point crossover operator. For this approach, we choose a random point in the chromosome and create children by taking everything before that point from x_1 and everything after from x_2 and vice versa for the other child.

Once we have the set of children, C, the mutation operator is applied to each child (line 10). The most basic mutation operator iterates through each dimension of the given chromosome and, with a very small probability p_m, replaces the value with a uniformly random one. Finally, after mutation, the children are added to the new population (line 11). This process (selection, crossover, and mutation) repeats until we have a new population $\mathcal{P}'$, which is the same size as the old one, $\mathcal{P}$. This marks the end of a generation. As the final step in each generation, we discard the current population and replace it with the new one (line 12). After all N_p generations have passed, we are left with a population $\mathcal{P}$ containing some of the best chromosomes we have seen so far. Typically, the last step (line 13) is to greedily choose the best chromosome from that population to solve the problem.

2.3 Differential Privacy

We work in what is called the central model of DP, where a user sends their data to a trusted curator who collects the data and runs certain private algorithms on the private data. That is, the curator is assumed to have access to the entire private dataset D in plaintext. We denote a general private algorithm as $\mathcal{M}$ that returns some aggregate statistic $\mathcal{M}(D) \in \mathcal{R}$. More formally, differential privacy can be defined as follows.

Definition 2.1 (Differential Privacy)**.** A randomized algorithm $\mathcal{M} : \mathcal{D} \mapsto \mathcal{R}$ is (ϵ, δ)-DP, if for any pair of neighbouring datasets $D, D' \in \mathcal{D}$, and for any $T \subseteq \mathcal{R}$ we have

$$\Pr[\mathcal{M}(D) \in T] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in T] + \delta. \tag{1}$$

Algorithm 2.1 The Simple GA

```
Inputs: $u$: Utility function.
    $p_c$: Fixed probability we do crossover.
    $p_m$: Fixed probability we do mutation.
    $N_g$: Number of generations.
    $N_p$: Size of the population.
1:  $\mathcal{P}$ = INITIALIZE($N_p$)
2:  for generation in range($N_g$) do
3:      $\mathcal{P}' = \emptyset$
4:      while $|\mathcal{P}'| \leq N_p$ do
5:          $x_1, x_2$ = SELECT($u(\mathcal{P})$)
6:          if Bernoulli($p_c$) then
7:              $C$ = CROSSOVER($x_1, x_2$)
8:          else
9:              $C = x_1, x_2$
10:         $C$ = MUTATE($C, p_m$)
11:         $\mathcal{P}' = \mathcal{P}' \cup C$
12:     $\mathcal{P} = \mathcal{P}'$
13: return $argmax(u(\mathcal{P}))$
```

The parameter ϵ captures how much sensitive information is leaked by the mechanism $\mathcal{M}$. The lower the value of ϵ, the better the privacy. Typically, values less than $\epsilon = 1$ are considered to provide high privacy. The parameter δ makes it easier to satisfy DP by allowing a small chance of failure in the guarantee. If $\delta \neq 0$, then we say that the mechanism provides approximate differential privacy. When $\delta = 0$, it satisfies pure differential privacy. It is typical to choose $\delta < 1/n$, where n is the number of elements in the dataset [18]. We say that two datasets are neighbouring if $|D| = |D'| = n$ and $|D \cap D'| = n - 1$. That is, we allow the replacement of a single data point. This is known as the bounded definition of differential privacy. If we apply a differentially private mechanism(s) sequentially, the resulting privacy parameter is simply the sum of the privacy parameter in each step. We call this the naive composition theorem [18]. Another useful property of differential privacy is the post-processing lemma [18]. This lemma states that once a DP result has been published, we can apply any additional processing we wish without affecting the privacy guarantee.

In this work, we focus on maximizing a given utility function while satisfying DP. We assume a public set of candidate items $\mathcal{R}$ (e.g. weights to a machine learning model or locations of a new facility) and corresponding utility function $u : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$. The objective is to return a candidate item $r \in \mathcal{R}$ that approximately maximizes $u(D, r)$, whilst satisfying differential privacy. In order to prove DP in this scenario, we must first introduce the concept of sensitivity.

Definition 2.2. The sensitivity of a utility function $u : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$ is defined as

$$\Delta^{(u)} = \max_{r \in \mathcal{R}} \max_{D, D' \in \mathcal{D}} |u(D, r) - u(D', r)| \tag{2}$$

where D, D' are neighbouring datasets.

The exponential mechanism, introduced by McSherry and Talwar, is the most common approach for privately maximizing a utility function [40]. The exponential mechanism randomly chooses a candidate following a specific distribution such that the utility function exponentially affects the probability of selecting a given output; the higher the utility, the larger the chance of selection. At the same time, the privacy parameters ϵ and $\Delta^{(u)}$ smooth the distribution to ensure the resulting selection is private (when $\epsilon = 0$ the distribution is uniform).

Definition 2.3 (Exponential Mechanism [40])**.** The *exponential mechanism* defines a probability distribution in which each output r, is sampled with the following probability:

$$Pr[r] = \frac{\exp\left(\frac{\epsilon u(D,r)}{2\Delta^{(u)}}\right)}{\sum_{i \in \mathcal{R}} \exp\left(\frac{\epsilon u(D,i)}{2\Delta^{(u)}}\right)} \tag{3}$$

The exponential mechanism, as defined above, guarantees ϵ-differential privacy [40, Theorem 6].

3 PROBLEM SETUP

3.1 Defining DP Selection

Given a set of (public) candidate items $\mathcal{R}$ and corresponding utility function $u : \mathcal{D} \times \mathcal{R} \rightarrow \mathbb{R}$, the objective is to return a candidate item $r \in \mathcal{R}$ that approximately maximizes $u(D, r)$, whilst satisfying differential privacy. The only assumption we make about a given utility function is that the sensitivity (defined in (2)) is bounded. This paper aims to show that GAs can provide a general purpose, high utility optimization algorithm for solving DP selection problems with large and complex candidate item spaces. We will evaluate this goal experimentally in terms of the utility of the solutions obtained on a variety of problems in Section 5. To satisfy our goal of privacy, a complete proof of privacy is included in Section 4.4.

3.2 Example Problem Definitions

Throughout the paper, we will focus on two specific instances of DP selection, the convex optimization problem of training a logistic regression model and the non-convex problem of k-medians.

Logistic Regression. Let $\mathcal{D}$ represent the data domain, where each entry x is a tuple of dimension d, with an integer label y. We consider a dataset $D \sim \mathcal{D}$. We study the task of creating a classifier $h()$ that, given an input x, can accurately predict the true label y. The model has a d-dimensional vector of weights, α, and a bias β. The goal in this problem is to find a set of parameters, (α, β) such that the following function is maximized

$$u(D, (\alpha, \beta)) = -\frac{1}{|D|} \sum_{x \in D} \mathbb{I}(\lfloor h(x) \rceil \neq y) \tag{4}$$

where $\mathbb{I}$ is an indicator function, $\lfloor h(x) \rceil$ represents rounding $h(x)$ to the nearest integer, and y represents the true label of a point x. This represents the standard zero-one loss, i.e., the fraction of miss-classified instances in the training set.

k-Median / Facility Location. The goal of the k-median problem is to find a set of k cluster centers (or facilities) that minimize the distance from all private data points to their nearest cluster center. The k-median problem has multiple definitions, but informally, the problem definition lies between k-medoids (centers are from

the dataset) and k-means (centers are real-valued vectors). In the privacy literature [26, 33], k-median refers to selecting the cluster centers from a finite domain set V. Almost unanimously, the k-median problem considers the ℓ_1 distance metric. We generalize the previous definition to distinguish between the private set of demand points and a public set of possible facilities. We remark that setting the public set to be the domain V gives the previous definition from the literature. We let V represent the data domain, where each record is a tuple of dimension d. We consider a private dataset $D \subset V$ and a public set of potential medians (facilities) $P \subset V$ such that $P \cap D = \emptyset$.[1] The goal is to choose $M \subset P$ with $|M| = k$ such that the following function is maximized.

$$u(D, M) = -\sum_{x \in D} \min_{y \in M} dist(x, y) \tag{5}$$

where we assume $dist$ represents ℓ_1 distance.

3.3 Datasets

To instantiate our example problems, we use a series of binary classification datasets from the UCI machine learning repository [15]. We use Adult (48842×104), Credit [52](30000×24), Spam (4601×57), and Mushrooms (8214×107). We preprocess these datasets by replacing the missing values with the mean or mode of the column. All data is normalized using min-max scaling such that each column is in the range $[0, 1]$. Categorical attributes are converted to numerical ones using a one-hot or binary encoding. For the logistic regression, we use an $80 : 20$ train to test split and report all evaluations on the test set. For k-median, we disregard the classification label and use the entire dataset as the private set. To create the public set, we sample additional points uniformly from the range of the attributes such that we achieve an $80 : 20$ private to public data ratio.[2]

4 DESIGN

4.1 Investigating PrivGene

We recall that PrivGene [54] is the first and only prior work to investigate DPGAs[3]. To make their GA private, Zhang et al. replace the selection operator of the GA (e.g. roulette wheel) with calls to the exponential mechanism (Definition 2.3). That is, instead of the likelihood of choosing a chromosome x_1 being $u(x_1)$ (as in roulette wheel) it is now $exp(u(x_1)\epsilon/2\Delta^{(u)})$ (following the exponential mechanism) where u is the utility function of the given problem. For example, x_1 could be a set of weights for a machine learning model, and $u(D, x_1)$ could be the accuracy of the model on dataset D. Zhang et al. observe that all other GA operations are completely independent of the private data. Thus, the privacy cost for the GA can be computed by summing the cost of each call to the exponential mechanism (by naive composition). As we mentioned, the fundamental challenge in making a GA private is that the total privacy budget must be divided over all selections. Thus, the more selections the GA makes, the lower the utility of each selection. To this end, PrivGene employs truncation selection [48], where one first selects the top N_s chromosomes from the population, then randomly chooses parents from this set in order to perform the usual breeding, crossover, and mutation operations. This reduces the number of selections from the size of the population to N_s (from 200 to 10 in PrivGene).

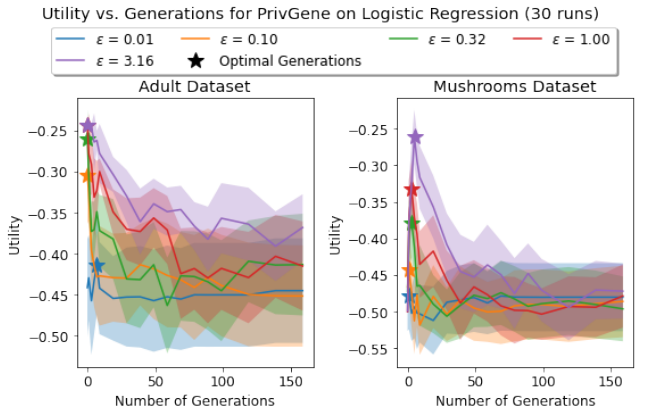


Figure 1: Investigating the utility of PrivGene for different numbers of generations.

While the initial design of PrivGene is promising, recent work has shown poor utility when evaluated in practice [50]. Su et al. point out a number of reasons for the poor performance of PrivGene. The first is that the crossover operator is destructive and often results in poor candidates. PrivGene uses single-point crossover (with a 100% probability of crossover) and a uniform mutation (applied to a single dimension of each solution with 100% probability) with decreasing scale over the generations. We observe that both the mutation and crossover operators allow no chance of the N_s chromosomes making it to the next generation.

Su et al.'s second reason comes back to the fundamental challenge of making a GA private. Despite their efforts using truncation selection, the GA still needs many generations to converge, and each generation requires some of the overall privacy budget to make selections. In order to satisfy DP, the number of generations is fixed to $c \cdot \frac{|D|\epsilon}{N_s} - 1$ where c is a constant that was experimentally set to 1.25×10^{-3} in the paper. We observe that this formula tends to result in zero generations for most reasonable privacy parameters ($\epsilon \leq 3$) and dataset sizes. To confirm this hyperparameter setting is optimal, we run an additional experiment varying the number of generations for various privacy parameters. We plot the results in Figure 1. We run the experiment 30 times for each number of generations and compute the mean utility. The shaded region represents the 95% confidence interval of the mean. We find for all reasonable epsilon values, the optimal utility is achieved at zero generations. That means the GA simply initializes the population and chooses the best (this is equivalent to the sub-sampled exponential mechanism described in Section 5.3). Any application of GA operators only reduces performance over this naive baseline.

To summarize, the main takeaways from our investigation and the work of Su et al. is that 1) the GA operators used in PrivGene do

[1]This can be relaxed when considering $P = V$

[2]We note that if we were to instead split the dataset into a public and private set, the public set would be similarly distributed, and thus one would not need the private set to solve the problem.

[3]Zhang et al. also introduce a variant only using mutation, which we call PrivEEM. We focus on PrivGene for this work. However, we also evaluate PrivEEM in Section 5.2.4.

not allow high utility chromosomes to survive between generations. 2) under PrivGene, the best approach is to apply all of the privacy budget in a single shot rather than use the GA operations. In the following sections, we design a new DPGA that takes advantage of the strengths of PrivGene (using truncation selection) while addressing these issues.

4.2 Our DP Simple GA

We detail our complete solution in Algorithm 4.1. We start with the simple GA using standard crossover and mutation operators and apply the necessary changes to satisfy DP and address the weaknesses of PrivGene. The simple GA introduces a probability that we do not invoke crossover and mutation, which partially addresses the destructive nature of PrivGene's operators. In line 7, we also add *elitism*, which simply copies the best candidates to the next generation to ensure they are preserved. To add privacy, we introduce the standard privacy parameters ϵ, δ, representing the algorithm's total privacy budget. Our algorithm requires a DP selection mechanism `DPSelect` that is used whenever we choose the best candidates from the population to ensure DP. Since our algorithm makes multiple selections, we need to distribute the total privacy budget over all these selections. To compute the privacy budget per selection, we employ a DP composition theorem, `DPComp` in Lines 2 and 3. We do this using a binary search over ϵ_s with n compositions until we reach the desired total privacy budget ϵ (a common approach [25]). We use a binary search as most composition theorems do not have a closed-form inverse. This is possible since $\epsilon = \texttt{DPComp}(\epsilon_s, n, \delta)$ is monotonic with respect to ϵ_s.

Similarly to PrivGene, we employ truncation selection to reduce the privacy cost and help maintain selection pressure. This introduces a new hyperparameter N_s for the number of parent chromosomes. In lines 5 and 6 we choose the top N_s solutions from the current population in a differentially private manner. We then sample parents from these top N_s candidates uniformly at random in line 9. The new population is then created using the same procedure as the non-private simple GA. We note that we fix the number of generations in order to maintain DP. This means the population is not guaranteed to converge to a single solution. Thus, as the last step, we apply `DPSelect` to obtain the best solution from the final population.

To encode a solution to logistic regression, we simply concatenate the weights α and bias β. We restrict each entry to the range $[-1, 1]$ using truncation if necessary. For the k-median problem, we use an integer encoding where each entry represents the index of a unique median in P. We initialize the population randomly but insert 10 zero vectors into the logistic regression population as it is a common starting point for this problem. We use uniform crossover for logistic regression, where each weight is equally likely to come from either parent. That is, instead of single point crossover, we use $(d-1)$-point crossover. We use a modified uniform crossover operator for k-median that ensures unique entries. Specifically, we take two parent solutions x_1, x_2 and obtain the set union of their entries, then uniformly sample a child $c \sim x_1 \cup x_2$ without replacement. We mutate using Gaussian noise with variance 0.1 for logistic regression and use uniform replacement for k-medians (resampling if needed to ensure unique entries).

Algorithm 4.1 DPSGA

```
Inputs: $u$: Utility Function.
$p_c$: Fixed probability we do crossover.
$p_m$: Fixed probability we do mutation.
$N_g$: Number of generations.
$N_p$: Size of the population.
$N_s$: Number of chosen parents.
DPComp, DPSelect: DP composition and selection mechanisms.
$(\epsilon, \delta)$: Privacy Parameters.
1: $\mathcal{P}$ = INITIALIZE($N_p$)
2: $n = N_g * N_s + 1$
3: $\epsilon_s$ = BinarySearch(DPComp, $n, \epsilon, \delta$)
4: for generation in range($N_g$) do
5:     for selection in range($N_s$) do
6:         $\mathcal{B}$ = DPSelect($u(\mathcal{P} - \mathcal{B}), \epsilon_s$)
7:     $\mathcal{P}' = \mathcal{B}$
8:     while $|\mathcal{P}'| \le N_p$ do
9:         $x_1, x_2$ = UniformRandomSelect($\mathcal{B}$)
10:        if Bernoulli($p_c$) then
11:            $C$ = UniformCrossover($x_1, x_2$)
12:        else
13:            $C = x_1, x_2$
14:        $C$ = MutateAll($C, p_m$)
15:        $\mathcal{P}' = \mathcal{P}' \cup C$
16:    $\mathcal{P} = \mathcal{P}'$
17: return DPSelect($u(\mathcal{P}), \epsilon_s$)
```

4.3 Reducing the noise from DP

We designed our algorithm in such a way that we can easily use any DP selection mechanism and composition theorem. We begin by choosing the mechanism and theorem that gives the best performance. In Appendix C, we discuss how to reduce the sensitivity of the loss functions to further increase performance.

4.3.1 DP Selection Mechanisms. We describe the state-of-the-art mechanisms for DP selection: the exponential mechanism [40], report noisy max [18], and permute and flip [39]. For simplicity, we consider mechanisms that return a single element of maximal utility, where the utility function is problem specific (such as those in Section 3.2). To select multiple elements, we will employ a peeling version of each mechanism in a similar style to Durfee and Rogers [17]. That is, we select the best candidate using the chosen mechanism and then "peel off" (remove) this candidate from the pool before selecting again. We define the exponential mechanism in Definition 2.3.

Report Noisy Max. Report noisy max is a simple algorithm originally proposed to return the largest counting query from a list [18]. We consider the more general version that maximizes a given utility function u. The mechanism simply computes the utility of each outcome in $\mathcal{R}$ and adds Laplace noise, $Lap(\frac{2\Delta^{(u)}}{\epsilon})$, then returns the outcome with the largest *noisy* utility. The result ensures ϵ-differential privacy [18, Claim 3.9].

Permute and Flip Mechanism. McKenna and Sheldon recently introduced the permute and flip mechanism [39]. We describe the algorithm in Algorithm 4.2. Intuitively, this mechanism is very similar to the exponential mechanism, with the key difference being the permutation (which mimics sampling without replacement). The algorithm gives the same privacy guarantee of ϵ-differential privacy [39, Theorem 1]. McKenna and Sheldon show that this new mechanism never has a worse utility than the exponential mechanism and can improve upon it by up to a factor of 2 [39, Theorem 2].

Algorithm 4.2 Permute and Flip [39]

```
1: $u_* \leftarrow \max_{r \in \mathcal{R}} u(D, r)$
2: for $r$ in RandomPermutation($\mathcal{R}$) do
3:     $p_r \leftarrow \exp\left(\frac{\epsilon}{2\Delta^{(u)}}(u(D,r) - u_*)\right)$
4:     if $Bernoulli(p_r)$ then
5:         return r
```

Implementation. We note that both the exponential mechanism and permute and flip can be equivalently implemented using a noisy max algorithm. For the exponential mechanism, Durfee and Rogers proved that using noisy max with Gumbel noise is equivalent to the exponential (and peeling exponential) mechanism [17, Lemma 4.2]. Ding et al. recently proved that the permute and flip mechanism is equivalent to the noisy max with exponential noise [13, Theorem 5].[4] We implement both the exponential mechanism and permute and flip with their noisy max counterparts to improve efficiency as well as circumvent precision issues when the loss is very high.

4.3.2 DP Composition Theorems. Now we have defined the selection mechanisms, we investigate the best way to compose these mechanisms in order to use them repeatedly. We consider the case of adaptive composition and let ϵ_s denote the privacy budget of a single run of a specific mechanism. We state the total epsilon ϵ required for each composition theorem for all runs. Each theorem below is significantly tighter than the naive composition theorem of summing ϵ_s. The cost of this tighter composition is relaxing the overall privacy guarantee from pure to approximate DP. We fix δ to be $1/|D|^{1.1}$ to satisfy the rule of thumb that delta should be less than $1/|D|$ [18]. We list the (ϵ, δ)-DP guarantees of each composition technique below.

Advanced Composition. We begin with the most well-known composition theorem (aside from naive composition) introduced by Dwork et al. [19].

Lemma 4.1 (Advanced Composition [19]). *The adaptive composition of a ϵ_s-DP mechanism under n-fold adaptive composition is (ϵ, δ)-DP with*

$$\epsilon = \epsilon_s\sqrt{2n \ln 1/\delta} + n\epsilon_s(e^{\epsilon_s} - 1) \tag{6}$$

Kairouz et al. improved this bound and proved the optimal advanced composition theorem for a general (ϵ, δ)-DP mechanism [30]. We use the simplified theorem that provides a slightly looser bound with a closed-form expression described below.

Lemma 4.2 (Theorem 3.4 [30]). *The adaptive composition of a ϵ_s-DP mechanism under n-fold adaptive composition is (ϵ, δ)-DP with*

$$\epsilon = \min\left\{ n\epsilon_s, \frac{(e^{\epsilon_s} - 1)\epsilon_s n}{(e^{\epsilon_s} + 1)} + \epsilon_s\sqrt{2n \ln\left(e + \frac{\sqrt{n\epsilon_s^2}}{\delta}\right)}, \frac{(e^{\epsilon_s} - 1)\epsilon_s n}{(e^{\epsilon_s} + 1)} + \epsilon_s\sqrt{2n \ln\left(\frac{1}{\delta}\right)} \right\} \tag{7}$$

The Moments Accountant. Concentrated differential privacy (CDP) and Renyi differential privacy (RDP) are popular privacy definitions used for the composition of DP mechanisms that take advantage of the Renyi divergence to give bounds [7, 41]. We focus on the RDP as it tends to give a more accurate analysis [41]. Furthermore, composing mechanisms using RDP is equivalent to the moments' accountant of Abadi et al. [1]. To use RDP as a composition theorem, we must first provide a conversion from pure DP to RDP. We can then use the composition theorems of RDP to compose the mechanisms and finally convert back to approximate DP. We give the details and prove the following lemma in Appendix A

Lemma 4.3. *The adaptive composition of a ϵ_s-DP mechanism under n-fold adaptive composition is (ϵ, δ)-DP with*

$$\epsilon = \min_{\alpha}\left\{ \frac{n}{\alpha - 1} \ln\left(\frac{\sinh(\alpha\epsilon_s) - \sinh((\alpha - 1)\epsilon_s)}{\sinh(\epsilon_s)}\right) + \frac{\ln 1/\delta}{\alpha - 1} \right\}. \tag{8}$$

Bounded Range Composition. Bounded range DP was first introduced by Durfee and Rogers in their work on top-k queries [17].

Definition 4.4 (Range-Bounded [17]). Given a mechanism $\mathcal{M}$ that takes a collection of records in $\mathcal{D}$ to some outcome set $\mathcal{R}$, we say that $\mathcal{M}$ is ϵ-range-bounded (ϵ-BR) if for any $y, y' \in \mathcal{R}$ and any neighbouring databases D, D' we have

$$\frac{\Pr[\mathcal{M}(D) = y]}{\Pr[\mathcal{M}(D') = y]} \leq e^{\epsilon} \frac{\Pr[\mathcal{M}(D) = y']}{\Pr[\mathcal{M}(D') = y']}. \tag{9}$$

This definition says that in addition to the distribution of outputs being similar across neighbouring databases, the mechanism must also offer a similar distribution over the outputs. Bounded range is a general notion of privacy, but it is particularly useful for exponential mechanisms. All ϵ-DP mechanisms satisfy 2ϵ-BR; however, the exponential mechanism enjoys a tighter analysis of ϵ-BR [17, Lemma 4.3].[5] Using this more restrictive form of DP allows one to prove tighter composition bounds. Durfee and Rogers [17] showed this in their initial work, which was later tightened by Dong et al. [14]. We give the version with a closed-form expression that was proven by computing the supremum of the KL divergence.

Lemma 4.5 (Proposition 4 [14]). *The adaptive composition of a ϵ_s-BR mechanism under n-fold adaptive composition is (ϵ, δ)-DP with*

$$\epsilon = \min\left\{ n\epsilon_s, n\left(\frac{\epsilon_s}{1 - e^{-\epsilon_s}} - 1 - \ln\left(\frac{\epsilon_s}{1 - e^{-\epsilon_s}}\right)\right) + \sqrt{\frac{n\epsilon_s^2}{2}\ln(1/\delta)} \right\}. \tag{10}$$

[4]It is currently an open question whether Gaussian noise could be used in a noisy max algorithm. However, lower bounds on the DP selection problem suggest the mechanisms we have presented are essentially optimal [13, 49]

[5]A natural question to ask is if the permute and flip mechanism also enjoys a similar analysis. A recent blog post by Durfee and Rogers shows that is likely not the case as the permute and flip mechanism has a lower bound close to 2ϵ-BR (the value that applies to all ϵ-DP algorithms) [16].

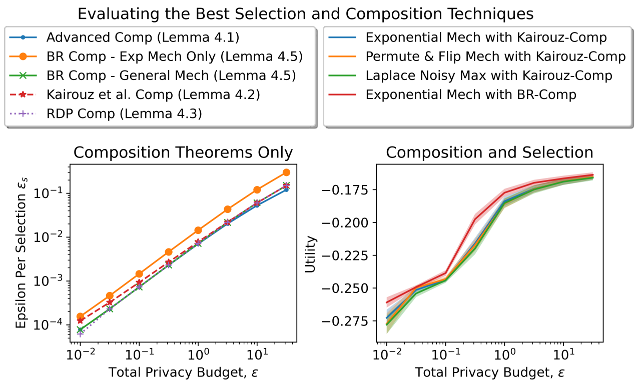


Figure 2: Comparing the various selection mechanisms and composition theorems to choose the best for our DPGA.

4.3.3 Evaluation of the Best Selection and Composition Technique. We have presented several selection mechanisms and composition theorems. However, it is not apparent which combination is best. We first narrow down the list of composition theorems. We plot the per selection privacy budget ϵ_s for a given overall privacy budget ϵ and a typical run of our simple GA. That is we let $n = 1000$ (100 generations, each with 10 selections) and fix $\delta = 10^{-5}$. To compute the corresponding ϵ_s, we simply binary search, evaluating the equation given by each lemma and plot the results in the left plot of Figure 2. We remark that a higher ϵ_s means more privacy budget is available at each step and thus a preferable composition theorem.

We observe BR composition on the exponential mechanism outperforms all other techniques. However, BR composition on a general mechanism (all ϵ-DP mechanisms are 2ϵ-BR) is comparable to RDP. In general, we see a lot of similarities between the other techniques, although the composition from Kairouz et al. offers slight improvement for smaller values of ϵ. Thus, we choose Lemma 4.5 for the exponential mechanism and Lemma 4.2 for all other mechanisms.

Despite choosing the best composition theorems, it is not clear which *combination* of composition theorem and DP mechanism will perform best. For example, the permute and flip mechanism is known to have better utility than the exponential mechanism but does not benefit from bounded range composition [16, 39]. Thus, we evaluate the exponential, permute and flip, and Laplace noisy max mechanisms using the composition theorem of Kairouz et al. (Lemma 4.2). Additionally, we will consider the exponential mechanism under bounded range composition (Lemma 4.5). We give the results in the right plot of Figure 2 using the Adult dataset on Logistic Regression (extended results in Appendix D). We find that using the exponential mechanism with bounded range composition consistently gives the best results. Furthermore, the composition theorem seems to be the dominating factor, as all three mechanisms perform very similarly under advanced composition. Thus, we chose the exponential mechanism as `DPSelect` and bounded range composition as `DPComp` for our work.

4.4 Privacy Guarantee

We state and prove the end-to-end privacy of our algorithm, assuming that we use the exponential mechanism as `DPSelect` and bounded range composition [14, 17] as `DPComp`. We use the utility functions described in Section 3.2.

Theorem 4.6. *If we set the parameter of the exponential mechanism to $\frac{\epsilon_s u(D,r)}{2\Delta^{(u)}}$ as specified in Definition 2.3, Algorithm 4.1 is (ϵ, δ)-DP where*

$$\epsilon = \min\left\{n\epsilon_s, n\left(\frac{\epsilon_s}{1-e^{-\epsilon_s}} - 1 - \ln\left(\frac{\epsilon_s}{1-e^{-\epsilon_s}}\right)\right) + \sqrt{\frac{n\epsilon_s^2}{2}\ln(1/\delta)}\right\}$$

with $n = N_g * N_s + 1$.

At a high level, proving the privacy of our simple GA consists of the following four parts.

(1) Bound the sensitivity of a given utility function.
(2) Prove that each selection (Algorithm 4.1, line 6 and 17) is ϵ-BR (or ϵ-DP).
(3) Prove that all other components of the simple GA incur no additional privacy cost.
(4) Prove that the adaptive composition of all selections is (ϵ, δ)-DP.

We begin by bounding the sensitivity of our example problem utility functions under bounded-DP.

Lemma 4.7. *The sensitivity, $\Delta^{(u)}$ of the zero-one loss for logistic regression (given in (4)) is at most $\frac{1}{|D|}$*

Proof. For a given vector θ changing D for D' changes at most one entry in the summation. This can change the value of the indicator function by at most one. Thus, the sensitivity is equivalent to the normalization term $\frac{1}{|D|}$. □

Lemma 4.8. *The sensitivity, $\Delta^{(u)}$ of the k-median loss function (given in (5)) using the ℓ_1 distance metric is d*

Proof. For a given set M changing D for D' changes at most one entry in the summation. This can change the minimum of that entry by at most the sensitivity of $dist(x, y)$. Assuming we use the ℓ_1-norm and all records in V are normalized[6] onto the interval $[0, 1]$, $dist(x, y) \leq d$. □

We now prove Theorem 4.6.

Proof. We begin by showing that each selection is ϵ_s-BR. This follows from the fact we follow the definition of the exponential mechanism (Definition 2.3) and the result of Durfee and Rogers [17, Lemma 4.3]. Next, we discuss the various steps of the algorithm to show that the selection step is the only step that requires spending any privacy budget. We follow Algorithm 4.1 line by line. First, the random initialization (line 1) is entirely independent of the database, so it incurs no privacy cost. Then for a given generation, we first select the best parents (lines 5-6). This incurs a privacy cost of ϵ_s-BR and is performed $N_g * N_s$ times. The remaining steps in each generation perform post-processing on the output of this step

[6]We note that this is a rather pessimistic bound on the sensitivity that can be improved using domain information. We show one such example in Appendix C.

$\mathcal{B}$ (by Proposition 2.1 [18]). Specifically, line 9 uniformly selects from the pool, line 11 randomly combines the chosen solutions, and line 14 randomly changes the solutions created by crossover. None of these steps require the use of the utility function, and thus the dataset, in any way. Finally, after all generations have passed, we must choose the best solution from the final population (line 17). For this, we simply make one more selection, and thus we add this to our count n. Hence, the privacy budget consumption of our algorithm consists of $n = N_g * N_s + 1$ calls to exponential mechanism. Thus applying the adaptive composition theorem from Lemma 4.5 gives the desired result. □

4.4.1 Additional DP Costs. In addition to the privacy cost described in Section 4.4, there are two other potential sources of leakage in our system: dataset pre-processing and hyperparameter tuning. Following the related work we compare to [26, 28, 33], we assume all datasets have features scaled to the range $[0, 1]$ and have no missing values. However, in practice, we must also ensure that the process of normalization and imputation is done privately. We use min-max scaling, which is optional for logistic regression, but needed to bound the sensitivity of the k-median problem [26]. In many cases, a pessimistic min and max value can be derived publically e.g., age could be bounded between 0 and 125, or max pixel values are well known for an image. Any values outside of the public range can be clipped [28]. Neither of these approaches incurs any additional privacy cost. However, in the worst case, an analyst could assign an arbitrary portion of their privacy budget to compute the min and max using standard differentially private mechanisms. The simplest solution to replace missing values is to drop rows with missing values (and incurs no cost). An alternative is to use known private techniques for imputation [10]. Once again, the amount of privacy budget assigned to this mechanism is arbitrary.

The other potential source of leakage is hyperparameter tuning. First, we note that our "default" solution, given in all experiments, uses folklore hyperparameters that are fixed across all datasets and problems. Thus, there would be no additional cost to use these values, and they perform well in our experiments. Our "optimal hyperparams" solution is obtained using a grid search specific to each dataset and problem (following the related work of Iyengar et al. [28]). This process indeed incurs additional privacy costs in practice. It is used only to ensure a fair comparison where all algorithms (including those from related work) use their optimal hyperparameters. In practice, a data analyst could optimize the hyperparameters on a public or random dataset for the given problem at no cost to privacy. If public data is unavailable, the analyst can use a known random search technique [46] (similar to a grid search, but choosing the number of configurations following a Poisson distribution) on the private data. The cost of this approach would be 2ϵ, where ϵ is the total privacy cost proved in Theorem 4.6 [46]. However, our experiments show that the expected gain in utility would not justify doubling the privacy cost of our default solution.

5 EVALUATION

In this section, we empirically evaluate the utility of our DPGA. First, we tune the hyperparameters of our GA and confirm that our modifications improve the utility over PrivGene. We also evaluate a variation of PrivGene, which we call PrivEEM. Second, we compare our DPGA to the sub-sampled exponential mechanism, the state-of-the-art *general* solution to the DP selection problem. Third, we compare our DPGA to Local Search techniques from the literature. Finally, we consider how our DPGA compares to state-of-the-art problem-specific solutions.

5.1 Setup

In each evaluation, we consider a trivial baseline and a non-private solution to give context to the performance of the algorithms. For logistic regression, the trivial baseline is a solution of all zero vectors. Whereas for k-median, we sample 1000 solutions uniformly at random and plot the average utility. As non-private algorithms, we use Scikit-learn packages. For logistic regression, we use the default logistic regression model with the stochastic average gradient solver.[7] For k-median, we consider the partitioning around medoids (PAM) algorithm of Kaufman and Rousseeuw [32]. We adapt the Scikit-learn implementation[8] to consider distances between a private and public set (rather than distances between points in the same set). We make all source code, including this modification, publicly available.[9]

We repeat each evaluation over 100 runs (unless otherwise specified), varying each run's random seed. We plot the mean utility over all runs and show the 95% confidence interval as a shaded region. For all evaluations, we consider multiple values of the privacy parameter ϵ with a fixed δ. We fix $\delta = 1/|D|^{1.1}$ to satisfy the rule of thumb that delta should be less than $1/|D|$ [18]. Due to space restrictions, we present each evaluation on two of our four datasets. Results on the other datasets can be found in Appendix D. We choose the Adult and Mushrooms datasets for most evaluations as they are the most representative. The only exceptions are Section 5.5.1, where Credit and Mushrooms better represent the performance and Section 5.4.2, where we could only use the smaller Spam and Mushrooms datasets.

5.2 Evaluating our DPGA

We start by studying the effect of hyperparameters on the utility of our DPGA. We are particularly interested in the relationship between the hyperparameter values and the privacy budget ϵ.

5.2.1 Number of Generations. As we have mentioned, the number of generations parameter is the most affected by the privacy budget as the more generations we conduct, the more DP degrades the selections made in each generation. To investigate this parameter, we fix all other hyperparameters to the following defaults (used as the default in all experiments). $p_c = 0.5$, $p_m = \frac{1}{|c|+1}$ where $|c|$ is the length of the chromosome, $N_p = 200$, and $N_s = 10$. In Figure 3, we vary the number of generations for various values of ϵ (similar to Figure 1 for PrivGene).

We see that, on average, for $\epsilon \geq 0.1$, the optimal utility is reached using more than zero generations, showing that our modifications enable the use of GA operations (and they improve utility). As expected, the lower the privacy budget, the fewer feasible generations,

[7]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[8]https://scikit-learn-extra.readthedocs.io/en/stable/generated/sklearn_extra.cluster.KMedoids.html
[9]https://gitlab.uwaterloo.ca/t3humphr/dp-simple-ga

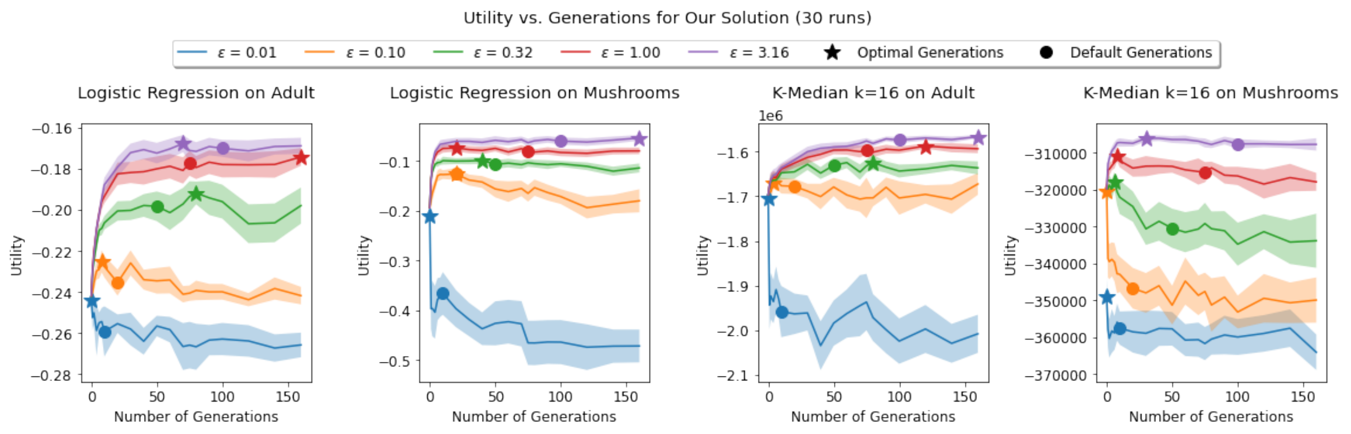


Figure 3: Investigating the utility of our DPGA for different numbers of generations.

and for extremely low values ($\epsilon = 0.01$), it is impossible to run multiple generations. We also observe a relatively flat trend after 30 generations for higher epsilon values. This seems to indicate an equilibrium between the improvement of adding more generations vs. the decrease of utility for each selection.

We take the average optimal values over all datasets and problems to construct a list of default generations per epsilon. Specifically, we choose $N_g = \{10, 10, 20, 50, 75, 100, 120, 120\}$ for $\epsilon \in \{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}, 10^{0.5}, 10^{1}, 10^{1.5}\}$ respectively. The performance of these values are highlighted in Figure 3. We observe that the defaults tend to give similar utility to the optimal number of generations (except when $\epsilon = 0.01$).

5.2.2 *Grid Search over Remaining Hyperparameters.*

We further conduct a full grid search over 2016 configurations of the 4 most influential parameters: $N_g \in \{2, 10, 20, 50, 75, 100, 120\}$,$N_s \in \{5, 10, 20\}$, $p_c \in \{0.35, 0.5, 0.65\}$, and $p_m \in \{\frac{2}{|c|+1}, \frac{1.5}{|c|+1}, \frac{1}{|c|+1}, \frac{0.75}{|c|+1}\}$ The remaining hyperparameter N_p is fixed to its default value of 200. We run each configuration for various epsilons to study the effect of privacy on hyperparameter tuning. The full experiment results are given in Appendix B. Similar to the generations parameter, we see N_s exhibits an approximate trend of decrease with lower epsilons. However, this parameter's impact on utility is minor compared to the generations parameter, so we keep the default of $N_s = 10$. There was no clear relationship between epsilon and p_m or p_c, so we posit that they can be tuned following techniques from the non-private literature. We choose to maintain their default values of $p_c = 0.5$ and $p_m = \frac{1}{|c|+1}$ as we observe reasonable utility.

We further use the grid search results to evaluate the optimal performance of the DPGA when all hyperparameters are tuned. As mentioned in Section 4.4.1, we do this to ensure a fair comparison to related work, as well as to evaluate if hyperparameter tuning is worth the extra privacy cost it would incur in practice. Specifically, we compute the optimal utility over the 30 runs of each configuration, then re-run the optimal configuration for an additional 70 runs. This optimal performance is plotted alongside the performance using the default parameters in our evaluations against related work in Section 5.3 (e.g. Figure 6). On average, we observe only a slight advantage to hyperparameter tuning and conclude that hyperparameter tuning is helpful but not necessary for our algorithm in the use cases we have considered.

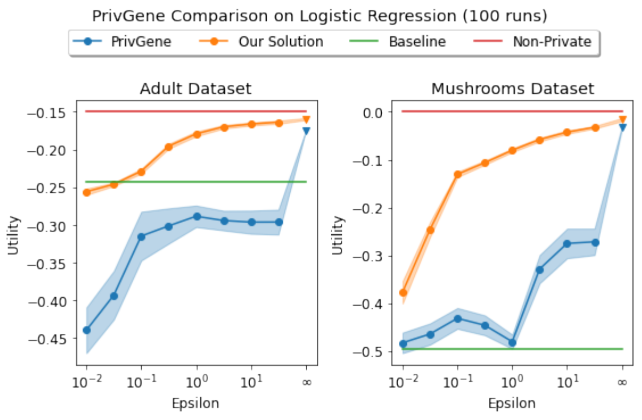


Figure 4: Comparing the utility of our DPGA to PrivGene [54].

5.2.3 *Comparison to PrivGene.*

To confirm our modifications strictly improve over PrivGene, we plot the utility of our solution compared with theirs in Figure 4. On the y-axis, we give the utility of the candidate returned by the GA (higher is better). On the x-axis, we evaluate various levels of privacy ϵ (lower is more private). We also include a data point for $\epsilon = \infty$ representing the non-private performance of the algorithms. We observe that our solution consistently improves over PrivGene (both privately and non-privately). We remark that the performance we observe for PrivGene is in line with the results of Su et al. [50, Figure 6].

5.2.4 *Evaluation of PrivEEM.*

To address the limitations of their DPGA, Zhang et al. also introduce a variant of PrivGene, which we call PrivEEM, where the chosen size (N_s) is reduced to 1. This removes the crossover operator completely, making PrivEEM an evolutionary strategy that only uses mutation. This modification

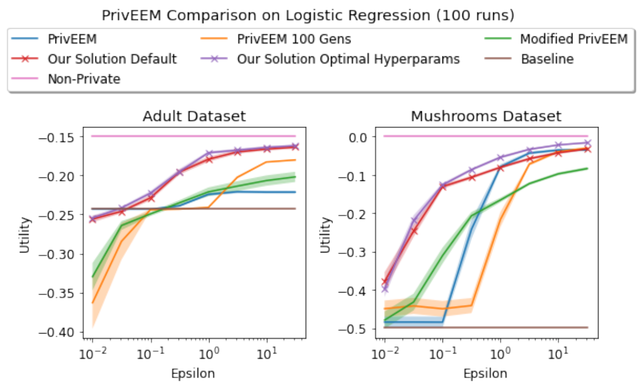


Figure 5: A comparison of our DPGA with PrivEEM [54].

allows the use of the enhanced exponential mechanism (EEM), a mechanism introduced by Zhang et al. with tighter sensitivity analysis than the exponential mechanism. We note that this only applies to logistic regression, and thus (similar to Zhang et al. [54][Section 5.3]), we only evaluate PrivEEM on logistic regression.

We set all parameters to the defaults specified by Zhang et al. [54]. We note that PrivEEM uses the same formula for the number of generations as PrivGene. That is, $c \cdot \frac{|D|\epsilon}{N_s} - 1$ where c is a constant that was experimentally set to 1.25×10^{-3} in the paper. We recall that this formula often leads to zero generations. To evaluate the potential of this approach, we also investigate several improvements to PrivEEM. First, we consider setting the number of generations to 100 so that the mutation operator will be invoked at lower epsilons. Second, we consider a modified version applying our tighter privacy analysis by using bounded range composition and the lower sensitivity zero-one loss function.[10] We give the results in Figure 5. We observe that the DPGA outperforms all variants of PrivEEM for $\epsilon > 0.1$ (except on Mushrooms where we tie). Furthermore, we always beat PrivEEM in the most important region of $\epsilon \in [0.1, 1]$. We find that increasing the generations to 100 has mixed success depending on the dataset. Our modified version outperforms 100 generations as expected; however, for larger ϵ values, the 100 generations solution can still be best. This is likely due to the enhanced exponential mechanism. We leave further investigation into this evolutionary strategy to future work.

5.3 Sub-Sampled Exponential Mechanism

Since our work is not problem-specific, we evaluate against another general DP selection technique. We consider the sub-sampled exponential mechanism from Lantz et al. [33]. This technique first takes a random sample from the space of possible solutions before applying the standard exponential mechanism. We evaluate the sub-sampled exponential mechanism on both logistic regression and the k-median problem. The original paper by Lantz et al. also considered the k-median problem and showed competitive performance compared to the techniques of Gupta et al. [33] (evaluated in Section 5.4.2).

We choose the sample size to be 1000 and run the sub-sampled exponential mechanism 100 times. In Figure 6, we see that the utility appears to plateau at a specific value much less than the non-private baseline. For logistic regression, both our default and optimal parameter solution always outperforms the sub-sampled exponential mechanism when $\epsilon > 0.1$. For the k-median problem, we tend to outperform the sub-sampled exponential mechanism for $\epsilon >= 0.32$ (except on the Mushrooms dataset). This indicates our solution provides a much more stable solution in that offers significantly better utility when $\epsilon > 0.32$ ($\epsilon > 0.032$ for logistic regression). Furthermore, this means our algorithm gives the most promising alternative to date for the exponential mechanism on large solution spaces.

5.4 Local Search Algorithms

A local search algorithm is an alternate iterative search algorithm that takes a small greedy step towards a better solution in each iteration. Both of our example problems have been solved with local searches in the literature, so we compare these solutions to our DPGA.

5.4.1 PrivLocal. Su et al. first observed that the source code linked in the PrivGene paper did not implement PrivGene [50]. The source code actually implements a local search algorithm that did not appear in the PrivGene paper. Su et al. named this local search algorithm PrivLocal, and give a detailed description of it. In their evaluation, Su et al. show that PrivLocal performs much better than PrivGene. Thus, we also evaluate this local search algorithm. We confirmed that the source code publicly available for PrivGene[11] implements PrivLocal (and not PrivGene). We also confirmed that work by Lee and Kifer [34] that claims to evaluate PrivGene actually evaluated PrivLocal (which explains the difference in performance). We use Lee and Kifer's Python implementation of PrivLocal [12] in this evaluation.

Similar to our evaluation of PrivEEM, we also investigate improved variants with 100 generations and a modified version with our privacy analysis. We give the results in Figure 7. While PrivLocal gives much better performance than PrivGene, we once again observe very dataset-dependent performance. We significantly outperform PrivLocal on our first three test datasets over all values of ϵ. PrivLocal performs better on the Mushrooms dataset, slightly outperforming our default solution when $\epsilon > 0.3$. However, our optimized solution can consistently outperform PrivLocal. We see that 100 generations tend to decrease the performance of PrivLocal; however, our modified version drastically improves performance. Despite this increase, we observe the DPGA tends to give the best utility in the important region of $\epsilon \in [0.1, 1]$.

5.4.2 Gupta et al.'s k-Median. The first work in private k-medians by Gupta et al. is also a local search algorithm [26]. The algorithm is based on the non-private search of Arya et al. [3], simply replacing each selection with the exponential mechanism. To evaluate this algorithm, we implement it from scratch, introducing our notion of public and private set; however, in the original algorithm, $P = V$. We found that the algorithm is rather inefficient for large public set

[10] In this case, we bound the sensitivity using Lemma 4.7. We leave the tighter analysis of the enhanced exponential mechanism to future work.

[11] https://sourceforge.net/p/privgene/
[12] https://github.com/ppmlguy/DP-AGD

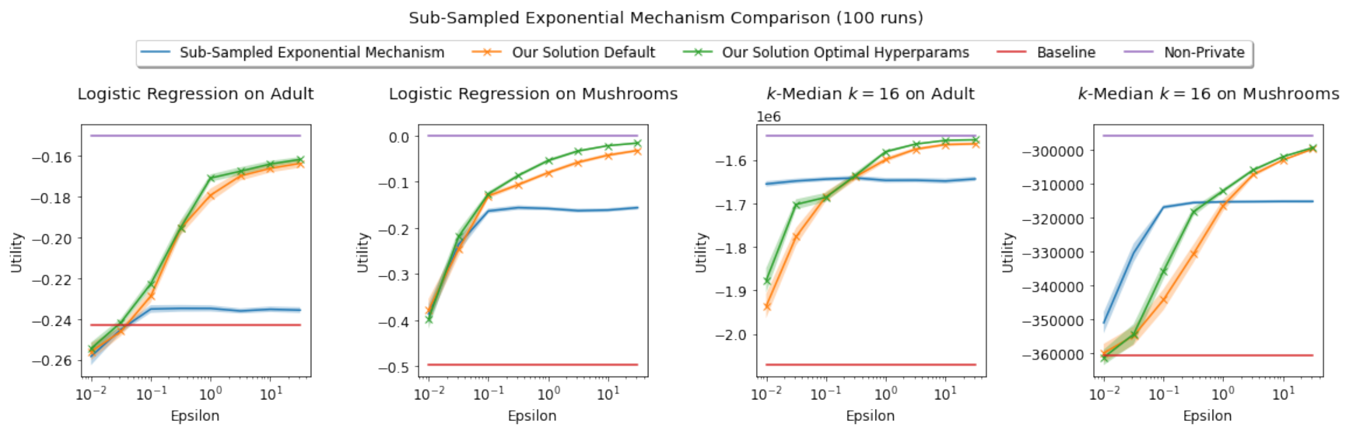


Figure 6: Comparing the utility of the DPGA vs. the sub-sampled exponential mechanism [33].

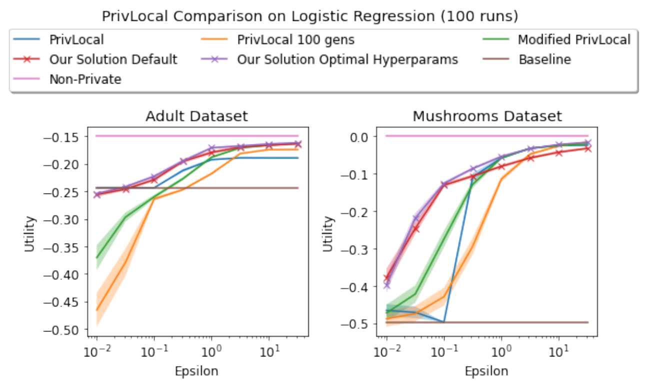


Figure 7: A comparison of our DPGA to PrivLocal [50, 54].

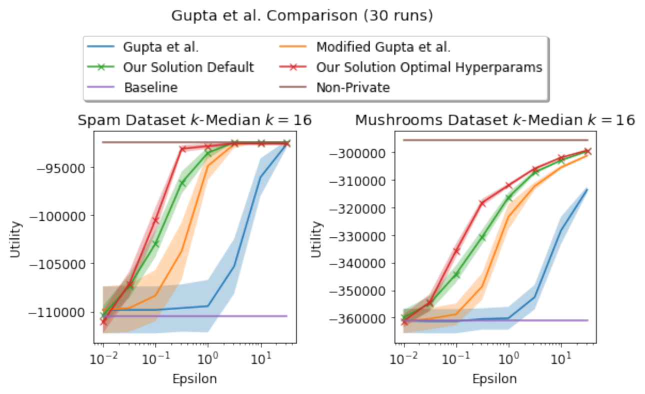


Figure 8: A comparison of our DPGA to Gupta et al.'s *k*-Median [26].

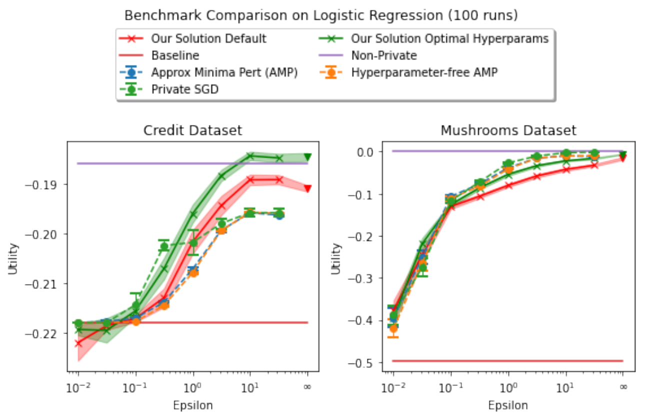


Figure 9: A comparison of our DPGA to the benchmark of private convex optimization techniques from Iyenger et al. [28].

sizes due to the large number of swaps considered. For this reason, we only consider our smaller datasets and a smaller number of runs. In Figure 8, we see that our solution drastically outperforms Gupta et al.'s solution. We conclude that in both utility and runtime, our solution is superior to that of Gupta et al. [26]. Our modified version once again greatly improves the performance of Gupta et al.'s solution. However, the DPGA still gives the best utility.

Our experiments on local search techniques show that they have more potential than has been previously shown in the literature. Applying our tighter privacy analysis can greatly improve their performance. However, in all experiments, we find that our DPGA offers the most promising result to date.

5.5 Problem Specific Techniques

In this section, we consider the state-of-the-art solutions specific to each of our example problems.

5.5.1 Logistic Regression. We consider the benchmark of state-of-the-art algorithms in DP convex optimization by Iyengar et al. [28]. The code was made publicly available[13], which allows us to simply run their benchmark, unchanged, on our own datasets. We note that the code performs a grid search over the hyperparameters of the various algorithms and reports the best in a similar manner

[13]https://github.com/sunblaze-ucb/dpml-benchmark

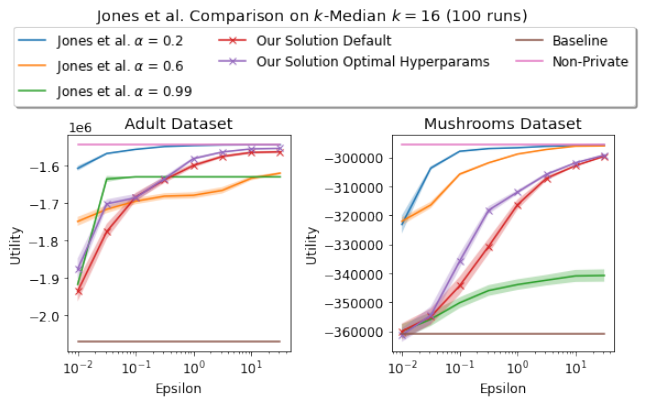


Figure 10: Comparing the utility of our DPGA against the state-of-the-art k-medians solution given by Jones et al. [29].

to our optimal hyperparameter solution. We focus on AMP and SGD as they tend to perform best, but we give the complete results in Appendix D.9. For this evaluation, we give the results for the Credit and Mushrooms dataset as they better reflect our best and worst performance, respectively (results on all datasets are given in Appendix D.9). In Figure 9, we observe that our optimal hyperparameter solution is very rarely outperformed, with a strong performance on the Credit dataset in particular.[14] We conclude that even against problem-specific solutions in private convex optimization, our general technique provides a competitive solution.

5.5.2 k-Median. We conduct the first empirical evaluation of work by Jones et al. [29], the theoretical state-of-the-art for the k-median problem, achieving smaller additive error than any other work. The algorithm first creates and publishes a DP weighted coreset, then applies an off-the-shelf non-private k-median algorithm. The algorithm has a single hyperparameter $\epsilon \in (0, 1)$, which we will rename α to avoid confusion. This parameter controls the number of iterations and plays a major role in the utility and runtime of the protocol (the lower the value, the better utility but the longer the runtime). For our experiments, we chose $\alpha \in \{0.2, 0.6, 0.99\}$. We did not choose $\alpha < 0.2$ as the runtime was too large, and the utility gains were insignificant. We implement the algorithm from scratch in Python using our problem definition of public and private sets. We use our non-private Scikit-learn baseline for the final step.

In Figure 10, we see that for higher values of α, our algorithm often outperforms Jones et al. significantly. Furthermore, our solution is significantly faster than the solution of Jones et al. in all experiments. However, for smaller values of α, Jones et al. gives the best utility (but is approximately twenty times slower on the Adult dataset). We note that the computation complexity of Jones et al. is $O(k|P||D| \log_{1+\alpha}(|D|) \ln(1/\alpha))$ as opposed to our solution that is $O(kN_pN_g|D|)$ where α and $N_p * N_g$ are technically constants, but, we include them as they have a significant effect in practice. In particular, as α approaches zero, $\log_{1+\alpha}(|D|)$ and $\ln(1/\alpha)$ approach infinity. We remark that our algorithm removes this additional log dependence on the private dataset and has no dependence on the size of the public set. Since our solution still offers reasonable utility with a significant improvement in efficiency, we believe it is indeed competitive with this problem-dependent approach.[15]

6 RELATED WORK

Differentially Private Selection. We have studied the exponential mechanism [40], the permute and flip mechanism [39] and the report noisy max mechanism [18]. All of these solutions suffer prohibitively high computation complexity as they require computing the utility of every candidate. One attempt to address this issue was the sub-sampled exponential mechanism, which we evaluated and found high-quality solutions are often discarded in the sampling [33]. In a similar light, Durfee and Rogers investigate solving the top-k selection problem for counting queries without considering the entire domain [17]. The limitations of this work are that they assume the domain of candidates is sorted in order of utility, and there is a non-trivial chance that no solution is returned. A recent line of work investigates privately tuning the hyperparameters of DP machine learning algorithms [35, 46]. While their work gives a method to apply the exponential mechanism on a subset of the domain, it does so randomly (like the sub-sampled exponential mechanism) and is best suited to the case of DP selection from DP candidates.

Differentially Private Nature-Inspired Algorithms. Very few works have focused on nature-inspired algorithms under the constraints of differential privacy. While PrivGene [54] has been the only work investigating differentially private genetic algorithms, other works have made use of swarm intelligence and evolutionary computation alongside differential privacy [36, 55]. However, these works either did not make the nature-inspired algorithms private [55], or focus on census protocols [36] instead of DP selection.

Problem Specific Differentially Private Solutions. Most works are not specific to DP logistic regression. Instead, they use it as one of the multiple examples to evaluate the effectiveness of their optimization algorithm. Recent work by Iyengar et al. [28] performs an extensive performance benchmark of state-of-the-art algorithms from the three major areas of work in differentially private convex optimization (output, objective, and gradient perturbation). We use this benchmark to evaluate the performance of our techniques in Section 5.5.1. The first work to define the k-median problem in a private setting was Gupta et al. [26]. Following up on this work, Lantz et al. used the k-median problem to show the effectiveness of the sub-sampled exponential mechanism [33]. The most recent work in this space is that of Jones et al. [29] (evaluated in Section 5.5.2). Other notable works solve variants of this problem such as DP k-means [4, 31, 45], the facility location problem [21, 26], and k-tuples clustering [9].

[14]We include the non-private GA performance in this plot to explain why the private GA outperforms the non-private baseline. For this dataset, the GA outperforms the stochastic average gradient (SAG) algorithm we use as the non-private baseline. We posit that this is due to the GA optimizing zero-one loss directly, whereas SAG uses cross-entropy loss (as zero-one loss is not differentiable).

[15]We remark that Jones et al.'s solution would suffer a severe drop in utility if we consider the k-medoid problem. This is because publishing the coreset when the centers are not public would require a significant amount of additional privacy budget. Our DPGA would not suffer the same loss as we only publish the index within the population and not the private set.

7 CONCLUSION

As the public becomes more privacy savvy, we are starting to see an increased effort from organizations to use privacy-preserving mechanisms. We show that despite previous misconceptions, genetic algorithms can offer competitive solutions to optimization problems *under the constraints of differential privacy*. A crucial trade-off influencing the adoption of privacy-preserving mechanisms is privacy vs. utility vs. efficiency. Our work provides a better compromise in terms of these objectives. First, the DPGA is proven to satisfy differential privacy. Second, the runtime scales polynomially with the size of the private dataset and does not require iterating over the whole domain of solutions. Finally, the utility of the DPGA is competitive with both general and problem-specific related work on the DP selection problem. We believe genetic algorithms are well suited to privacy-sensitive problems for these reasons.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council (NSERC) for grants RGPIN-05849 and IRC-537591,the NSERC Postgraduate Scholarship-Doctoral program, and the Royal Bank of Canada for funding this research.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] M Anbarasi, E Anupriya, and NCSN Iyengar. 2010. Enhanced prediction of heart disease with feature subset selection using genetic algorithm. *International Journal of Engineering Science and Technology* 2, 10 (2010), 5370–5376.

[3] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. 2004. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM J. Comput.* 33, 3 (2004), 544–562. https://doi.org/10.1137/S0097539702416402

[4] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. 2017. Differentially Private Clustering in High-Dimensional Euclidean Spaces. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 322–331.

[5] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. 464–473. https://doi.org/10.1109/FOCS.2014.56

[6] Gavin Brown, Marco Gaboardi, Adam Smith, Jonathan Ullman, and Lydia Zakynthinou. 2021. Covariance-aware private mean estimation without private covariance estimation. *Advances in Neural Information Processing Systems* 34 (2021), 7950–7964.

[7] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*. Springer-Verlag, Berlin, Heidelberg, 635–658. https://doi.org/10.1007/978-3-662-53641-4_24

[8] Kamalika Chaudhuri, Anand D Sarwate, and Kaushik Sinha. 2013. A Near-Optimal Algorithm for Differentially-Private Principal Components. *Journal of Machine Learning Research* 14 (2013).

[9] Edith Cohen, Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. 2021. Differentially-Private Clustering of Easy Instances. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 2049–2059. https://proceedings.mlr.press/v139/cohen21c.html

[10] Soumojit Das, Jorg Drechsler, Keith Merrill, and Shawn Merrill. 2022. Imputation under Differential Privacy. https://doi.org/10.48550/ARXIV.2206.15063

[11] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf

[12] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[13] Zeyu Ding, Daniel Kifer, Sayed M. Saghaian N. E., Thomas Steinke, Yuxin Wang, Yingtai Xiao, and Danfeng Zhang. 2021. The Permute-and-Flip Mechanism is Identical to Report-Noisy-Max with Exponential Noise. arXiv:2105.07260 [cs.CR]

[14] Jinshuo Dong, David Durfee, and Ryan Rogers. 2020. Optimal Differential Privacy Composition for Exponential Mechanisms. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 2597–2606. http://proceedings.mlr.press/v119/dong20a.html

[15] Dheeru Dua and Casey Graff. 2021. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[16] David Durfee and Ryan Rogers. 2021. One-shot DP Top-k mechanisms. DifferentialPrivacy.org. https://differentialprivacy.org/one-shot-top-k/.

[17] David Durfee and Ryan M Rogers. 2019. Practical Differentially Private Top-k Selection with Pay-what-you-get Composition. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/b139e104214a08ae3f2ebcce149cdf6e-Paper.pdf

[18] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (Aug. 2014), 211–407. https://doi.org/10.1561/0400000042

[19] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. 2010. Boosting and Differential Privacy. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS '10)*. IEEE Computer Society, USA, 51–60. https://doi.org/10.1109/FOCS.2010.12

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[21] Yunus Esencayi, Marco Gaboardi, Shi Li, and Di Wang. 2019. Facility Location Problem in Differential Privacy Model Revisited. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/41c576a3bac4220845f9427b002a2a9d-Paper.pdf

[22] David B Fogel. 2006. *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons.

[23] Jennifer Gillenwater, Matthew Joseph, Andres Munoz, and Monica Ribero Diaz. 2022. A Joint Exponential Mechanism For Differentially Private Top-k. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 7570–7582. https://proceedings.mlr.press/v162/gillenwater22a.html

[24] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[25] Google. 2023. Google's Differential Privacy Library. https://github.com/google/differential-privacy/.

[26] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially Private Combinatorial Optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Austin, Texas) *(SODA '10)*. Society for Industrial and Applied Mathematics, USA, 1106–1125.

[27] John Henry Holland et al. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press.

[28] Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. 2019. Towards Practical Differentially Private Convex Optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*. 299–316. https://doi.org/10.1109/SP.2019.00001

[29] Matthew Jones, Huy L. Nguyen, and Thy D Nguyen. 2021. Differentially Private Clustering via Maximum Coverage. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 13 (May 2021), 11555–11563. https://ojs.aaai.org/index.php/AAAI/article/view/17375

[30] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The Composition Theorem for Differential Privacy. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1376–1385. https://proceedings.mlr.press/v37/kairouz15.html

[31] Haim Kaplan and Uri Stemmer. 2018. Differentially Private k-Means with Constant Multiplicative Error. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 5436–5446.

[32] Leonard Kaufman and Peter J. Rousseeuw. 1990. *Partitioning Around Medoids (Program PAM)*. John Wiley & Sons, Ltd, Chapter 2, 68–125. https://doi.org/10.1002/9780470316801.ch2 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470316801.ch2

[33] Eric Lantz, Kendrick Boyd, and David Page. 2015. Subsampled Exponential Mechanism: Differential Privacy in Large Output Spaces. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security* (Denver, Colorado, USA)

(AISec '15). Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/2808769.2808776

[34] Jaewoo Lee and Daniel Kifer. 2018. Concentrated Differentially Private Gradient Descent with Adaptive Per-Iteration Privacy Budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1656–1665. https://doi.org/10.1145/3219819.3220076

[35] Jingcheng Liu and Kunal Talwar. 2019. Private Selection from Private Candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (Phoenix, AZ, USA) *(STOC 2019)*. Association for Computing Machinery, New York, NY, USA, 298–309. https://doi.org/10.1145/3313276.3316377

[36] Qishuai Liu and Qing Hui. 2018. The Bat-Inspired Consensus Protocols with Differential Privacy. In *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. 829–834. https://doi.org/10.1109/ICCA.2018.8444327

[37] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proc. VLDB Endow.* 10, 6 (Feb. 2017), 637–648. https://doi.org/10.14778/3055330.3055331

[38] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. 2008. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*. IEEE, IEEE, Cancun, Mexico, 277–286. https://doi.org/10.1109/ICDE.2008.4497436

[39] Ryan McKenna and Daniel R Sheldon. 2020. Permute-and-Flip: A new mechanism for differentially private selection. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 193–203. https://proceedings.neurips.cc/paper/2020/file/01e00f2f4bfcbb7505cb641066f2859b-Paper.pdf

[40] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*. IEEE Computer Society, USA, 94–103. https://doi.org/10.1109/FOCS.2007.41

[41] Ilya Mironov. 2017. Rényi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 263–275. https://doi.org/10.1109/CSF.2017.11

[42] Melanie Mitchell. 1998. *An introduction to genetic algorithms.* MIT press.

[43] David J Montana, Lawrence Davis, et al. 1989. Training feedforward neural networks using genetic algorithms.. In *IJCAI*, Vol. 89. 762–767.

[44] Shyam Narayanan, Vahab Mirrokni, and Hossein Esfandiari. 2022. Tight and Robust Private Mean Estimation with Few Users. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 16383–16412. https://proceedings.mlr.press/v162/narayanan22a.html

[45] Huy L. Nguyen, Anamay Chaturvedi, and Eric Z Xu. 2021. Differentially Private k-Means via Exponential Mechanism and Max Cover. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 10 (May 2021), 9101–9108. https://ojs.aaai.org/index.php/AAAI/article/view/17099

[46] Nicolas Papernot and Thomas Steinke. 2022. Hyperparameter Tuning with Renyi Differential Privacy. In *International Conference on Learning Representations*. https://openreview.net/forum?id=-70L8lpp9DF

[47] Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. University of California Press, 547–561.

[48] Dirk Schlierkamp-Voosen and H Mühlenbein. 1993. Predictive models for the breeder genetic algorithm. *Evolutionary Computation* 1, 1 (1993), 25–49.

[49] Thomas Steinke and Jonathan Ullman. 2017. Tight lower bounds for differentially private selection. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 552–563.

[50] Dong Su, Jianneng Cao, Ninghui Li, and Min Lyu. 2018. PrivPfC: Differentially Private Data Publication for Classification. *The VLDB Journal* 27, 2 (April 2018), 201–223. https://doi.org/10.1007/s00778-017-0492-3

[51] T.W. Then and E.K.P. Chong. 1994. Genetic algorithms in noisy environment. In *Proceedings of 1994 9th IEEE International Symposium on Intelligent Control*. 225–230. https://doi.org/10.1109/ISIC.1994.367813

[52] I-Cheng Yeh and Che hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2, Part 1 (2009), 2473–2480. https://doi.org/10.1016/j.eswa.2007.12.020

[53] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Trans. Database Syst.* 42, 4, Article 25 (Oct. 2017), 41 pages. https://doi.org/10.1145/3134428

[54] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. 2013. PrivGene: Differentially Private Model Fitting Using Genetic Algorithms. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2463676.2465330

[55] Ezgi Zorarpacı and Selma Ayşe Özel. 2020. Differentially private 1R classification algorithm using artificial bee colony and differential evolution. *Engineering Applications of Artificial Intelligence* 94 (2020), 103813. https://doi.org/10.1016/j.engappai.2020.103813

A RDP COMPOSITION THEOREM

We detail the building blocks and proof for our RDP composition theorem stated in Section 4.3.2. We begin by defining RDP. To define RDP, we must first recall Renyi divergence.

Definition A.1. Renyi Divergence [47] For two probability distributions P and Q defined over $\mathcal{R}$, the Renyi divergence of order $\alpha > 1$ is

$$D(P\|Q) = \frac{1}{1-\alpha} \ln \mathop{\mathbb{E}}_{x\sim Q} \left(\frac{P(x)}{Q(x)}\right)^{\alpha} \tag{11}$$

where $P(x)$ is the density of P at x.

Definition A.2. (RDP [41]) An algorithm A is said to be (α, ϵ)-RDP if for all neighbouring databases $D, D' \in \mathcal{D}$

$$D(A(D)\|A(D')) \le \epsilon. \tag{12}$$

The first step is to convert the arbitrary DP mechanism to the RDP definition. To do this, we take advantage of an intermediary result from Bun and Steinke [7, Proposition 19]. Specifically,

Lemma A.3 (From DP to RDP [7]). *If a mechanism satisfies ϵ_s-DP, it also satisfies $(\epsilon(\alpha), \alpha)$-RDP with*

$$\epsilon(\alpha) = \frac{1}{\alpha-1} \ln\left(\frac{\sinh(\alpha\epsilon_s) - \sinh((\alpha-1)\epsilon_s)}{\sinh(\epsilon_s)}\right) \tag{13}$$

After we have converted to RDP, we can take advantage of the following composition theorem (similar to naive composition for DP).

Proposition A.4 (Proposition 1 [41]). Consider two mechanisms F and G. If F is (α, ϵ_1)-RDP and G is (α, ϵ_2)-RDP then $F(G(x))$ is $(\alpha, \epsilon_1 + \epsilon_2)$-RDP

Finally, we need to convert the final result to approximate DP as follows.

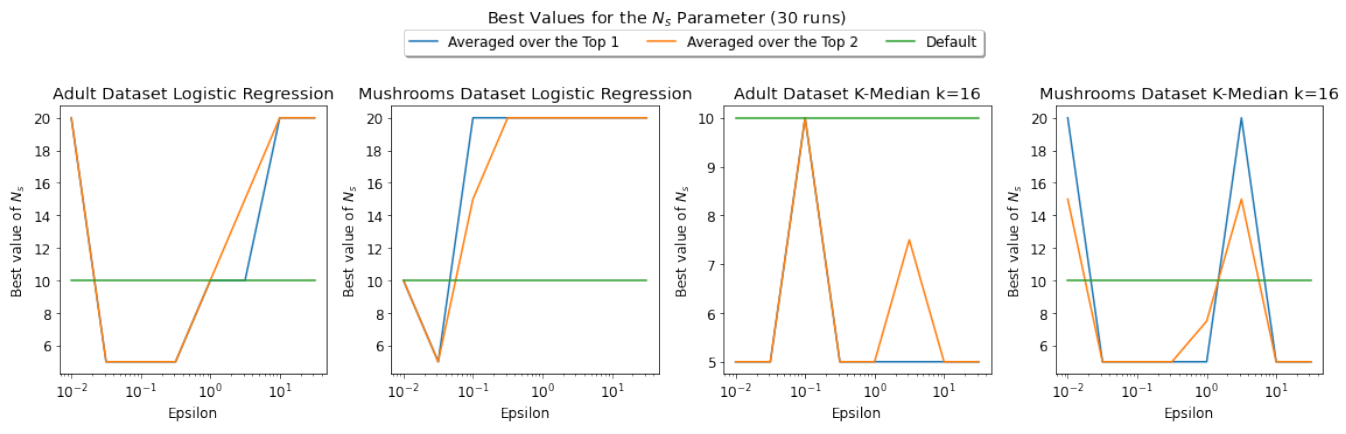
Proposition A.5 (Proposition 3 [41]). If a mechanism satisfies (α, ϵ)-RDP, then it is also $(\epsilon + \frac{\ln 1/\delta}{\alpha-1}, \delta)$-DP for any $\delta \in (0, 1)$.

Mirnov showed that in practice, it suffices to consider a restricted set of alphas and compute that α that approximately minimizes ϵ [41]. In our implementation, we take this approach, brute-forcing over a small set of alphas to find the empirical minimum. Putting all of these steps together, we obtain the following end-to-end composition theorem restated for convenience.

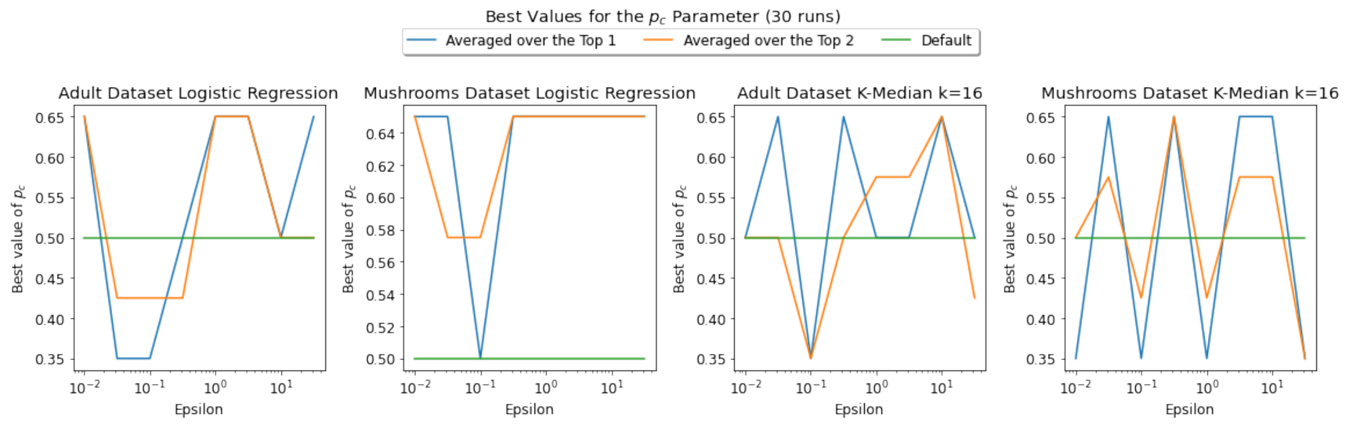
Lemma A.6. *The adaptive composition of a ϵ_s-DP mechanism under n-fold adaptive composition is (ϵ, δ)-DP with*

$$\epsilon = \min_{\alpha} \left\{ \frac{n}{\alpha-1} \ln\left(\frac{\sinh(\alpha\epsilon_s) - \sinh((\alpha-1)\epsilon_s)}{\sinh(\epsilon_s)}\right) + \frac{\ln 1/\delta}{\alpha-1} \right\} \tag{14}$$

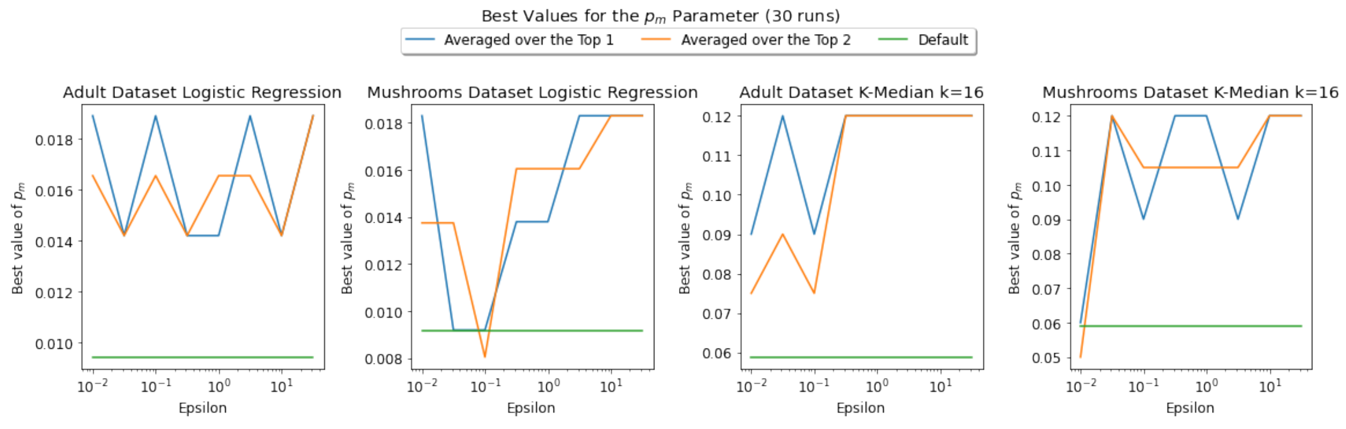
Proof. The result follows by first applying Lemma A.3, then multiplying by n as per Proposition A.4. Finally, we apply Proposition A.5 and minimize over α. □



(a)



(b)



(c)

Figure 11: The best settings for the various hyperparameters found from the grid search.

B HYPERPARAMETER SEARCH

In Section 5.2.2, we describe our hyperparameter search over various parameters. Specifically, we conduct a full grid search over 2016 configurations of the 4 most influential parameters: $N_g \in \{2, 10, 20, 50, 75, 100, 120\}$,$N_s \in \{5, 10, 20\}$, $p_c \in \{0.35, 0.5, 0.65\}$, and $p_m \in \{\frac{2}{|c|+1}, \frac{1.5}{|c|+1}, \frac{1}{|c|+1}, \frac{0.75}{|c|+1}\}$ The remaining hyperparameter N_p is fixed to its default value of 200. We present the best hyperparameter values for various epsilons in Figure 11. We omit the generations parameter as we studied it in Section 5.2.1, and the results here are similar. Similar to the generations parameter, we see an approximate trend of increase in the number of selections parameter with epsilon. We investigated setting default values in a similar fashion to the generations parameter. However, we found that the increase in performance was relatively small. Thus, we maintain the default value of 10.

The probability of crossover parameter does not have an obvious trend across datasets or problems. Thus, we also maintain the default value of 0.5. Finally, the probability of mutation parameter similarly has no obvious trend with epsilon. However, we generally see values higher than our default do perform better. We investigated increasing the mutation rate but once again found only a slight performance increase, so for simplicity, we maintain our default value of $\frac{1}{|c|+1}$.

C REDUCING SENSITIVITY

Outside of the choice of `DPSelect` and `DPComp`, another major contributor to the amount of random noise introduced by DP is the choice of the utility function. The lower the sensitivity of the function, the less noise must be added, and thus the utility improves. A recent line of work has found that by choosing a lower sensitivity utility function, the exponential mechanism can outperform state-of-the-art problem-specific techniques [6, 23, 44]. When training a logistic regression model, PrivGene uses standard cross-entropy loss. However, cross-entropy loss has a sensitivity of $O(d)$ [54]. Conversely, we use zero-one loss as it has a sensitivity of $O(1)$ (as shown in Section 4.4) and significantly increases the performance of the DPGA. We note that the zero-one loss is not typically used in the non-private literature since it is not differentiable. However, since our GA does not use gradients, we can take advantage of this loss function to decrease the impacts of DP.

For the k-median problem, the classic utility function (5) has sensitivity $O(d)$. We found that using alternatives such as the Silhouette index or the ℓ_∞ norm reduces the sensitivity but does not sufficiently represent the original loss function. Thus, we maintained the original loss function for all experiments in the paper. We did find that improvements can be made if we modify the existing utility function by carefully clipping the distance metric to reduce the sensitivity while preserving utility. In the unmodified version, the distance between two points is in $[0, d]$ (the width of this range gives the sensitivity). We instead use a distance metric where the distance between two points is in $[d/4, d/2]$. We do this by simply truncating any distance values outside this range. In the datasets we study, we find this to have minimal effect on utility. Few distances are below $d/4$, and distances above $d/2$ (approx. the mean) are not helpful when looking for a solution that minimizes distance. This technique reduces the sensitivity by a factor of 4 and significantly increases GA performance.

Figure 13 shows the effect on utility. We can see that, in general, this gives us a significant boost in utility across all datasets and values of epsilon. We note that this interval does not generalize to all datasets or distributions of public vs. private data. The credit dataset for higher epsilon values is one such example since the min is slightly higher than $d/4$. A simple approach to setting the clipping parameters, in such a case, would be first to compute the DP min and mean values of the distance matrix and then clip accordingly. However, we find that $[d/4, d/2]$ is a reasonable choice without tailoring to each specific dataset.

D ADDITIONAL EXPERIMENTS

In this section, we show additional figures that we could not fit into the paper. The majority of these are the same figures as the main paper on the datasets missing from the paper.

D.1 Selection and Composition Experiment

Figure 12 shows the results of the comparison described in Section 4.3.3 on additional datasets. We confirm that using the exponential mechanism with bounded range composition consistently gives the best results.

D.2 Utility vs. Generations for PrivGene

This corresponds to Figure 1 of the main paper. We give the results for the other datasets in Figure 14. We observe similar trends on the missing datasets, with zero generations being the optimal solution.

D.3 Utility vs. Generations for Our Solution

This corresponds to Figure 3 of the main paper. We give the results for the other datasets in Figure 16. We observe similar results to the main paper except for the Spam dataset on k-median. In particular, we see that for $\epsilon > 0.32$, the GA converges for any number of generations. We find it is very easy to solve the k-medians problem on the Spam dataset. This pattern is visible in all evaluations of this problem and dataset. For example, the sub-sampled exponential mechanism (equivalent to zero generations of the GA) also obtains the optimal non-private solution for $\epsilon > 0.32$ on this problem (Figure 17).

D.4 PrivGene Comparison

This corresponds to Figure 4 of the main paper. We give the results for the other datasets in Figure 15. We observe very similar results to the main paper, corroborating the observation that we improve over PrivGene, both privately and non-privately.

D.5 PrivEEM Comparison

This corresponds to Figure 5 of the main paper. We give the results for the other datasets in Figure 18. We observe very similar results to the main paper with Credit and Spam showing similar trends to the Adult dataset.

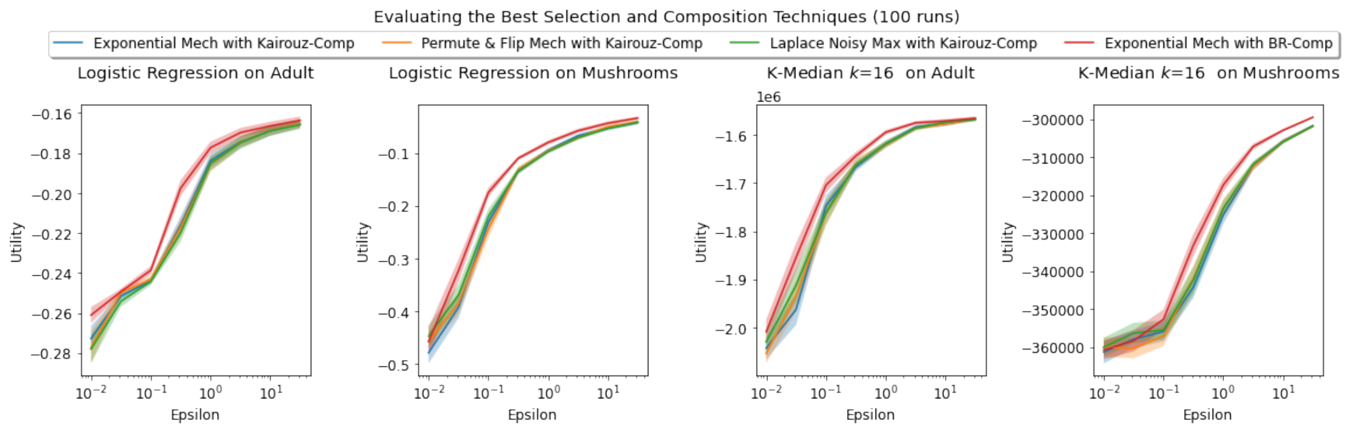


Figure 12: An evaluation of the various selection techniques and composition theorems discussed in this section.

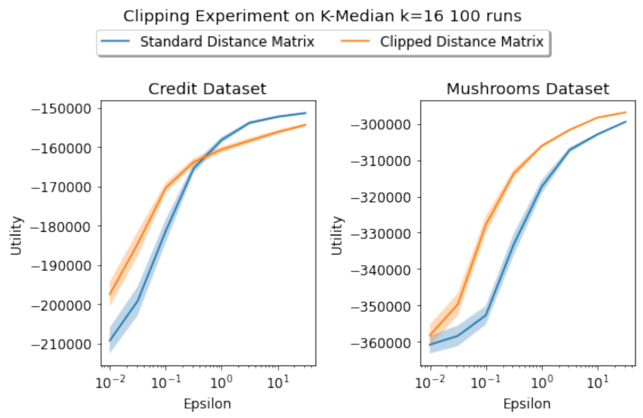


Figure 13: Evaluating the performance of distance matrix clipping.

D.6 Sub-sampled Exponential Mechanism Comparison

This corresponds to Figure 6 of the main paper. We give the results for the other datasets in Figure 17. We again see that the utility appears to plateau at a specific value much less than the non-private baseline (except k-median on Spam as discussed in Section D.3). Our optimal parameter solution always outperforms the sub-sampled exponential mechanism when $\epsilon > 0.1$.

D.7 PrivLocal Comparison

This corresponds to Figure 7 of the main paper. We give the results for the other datasets in Figure 19. We observe very similar results to the main paper, with even larger performance gaps in Spam and Credit.

D.8 Gupta et al. Comparison

This corresponds to Figure 8 of the main paper. We recall that we only used the two smallest datasets, Spam and Mushrooms, for this experiment due to the large runtime of this algorithm. We give the results for $k = 4$ in Figure 21. We observe very similar results to the main paper, with the modified version performing slightly better on this problem. Our conclusion remains the same the DPGA still gives the best utility and runtime.

D.9 Logistic Regression Benchmark Comparison

This corresponds to Figure 9 of the main paper. We give the results for all approaches in the benchmark on all of our datasets in Figure 20. We see that our solution performs comparably on the Adult and Spam datasets and that the excluded algorithms typically perform worse than the ones in the paper. Thus, we can still conclude that even against problem-specific solutions in convex optimization, our general technique provides a competitive solution.

D.10 Jones et al. Comparison

This corresponds to Figure 10 of the main paper. We give the results for the other datasets in Figure 22. We observe very similar trends on the missing datasets, with the Jones et al. solution performing best for higher values of α. However, for smaller values, where the runtime of Jones is significantly higher (approximately ten times our solution on Adult), the solution of Jones et al. is the best.

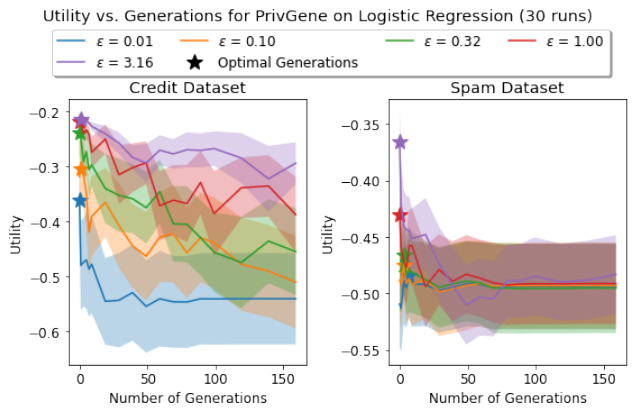


Figure 14: Investigating the utility of PrivGene for different numbers of generations.

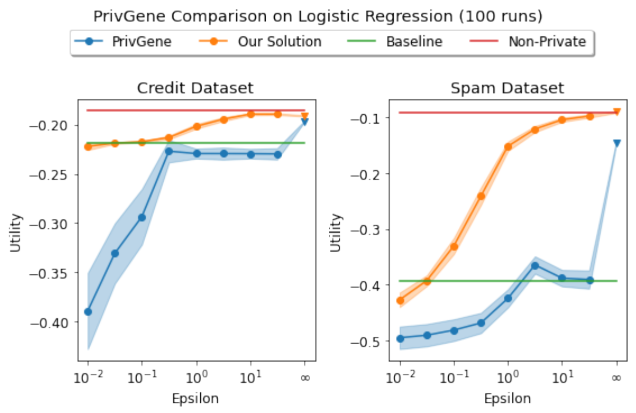


Figure 15: Comparing the utility of our DPGA to PrivGene [54].

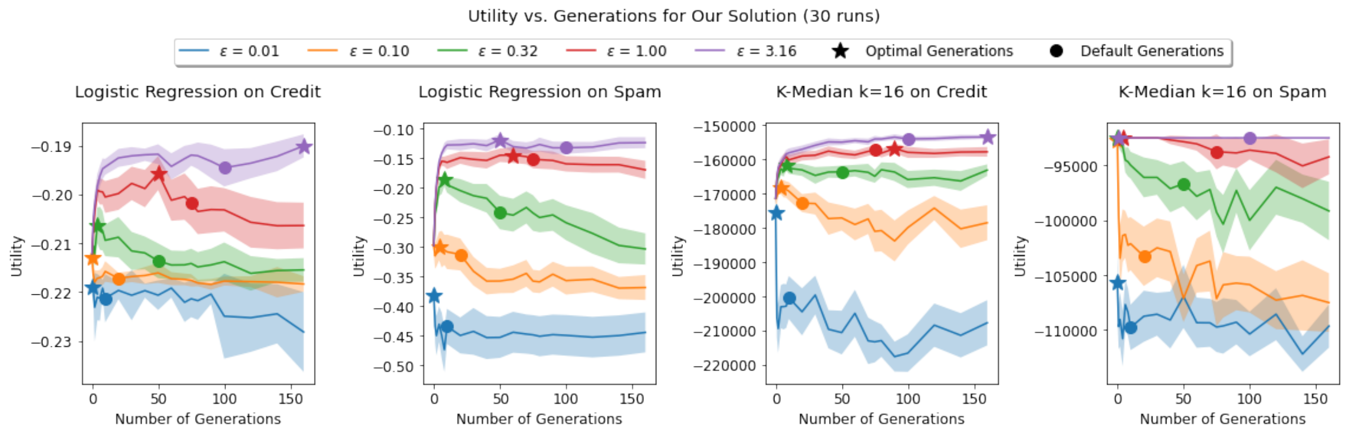


Figure 16: Investigating the utility of our DPGA for different numbers of generations.

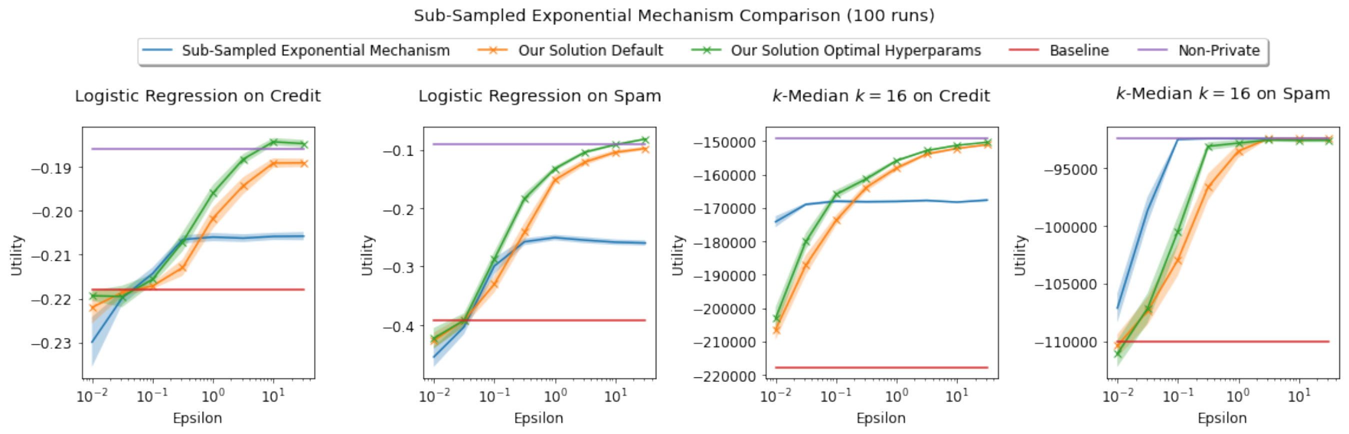


Figure 17: Comparing the utility of the DPGA vs. the sub-sampled exponential mechanism [33].

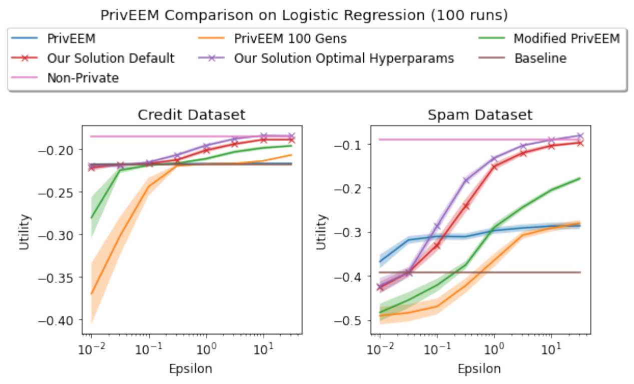


Figure 18: Comparing the utility of our DPGA to PrivGene [54].

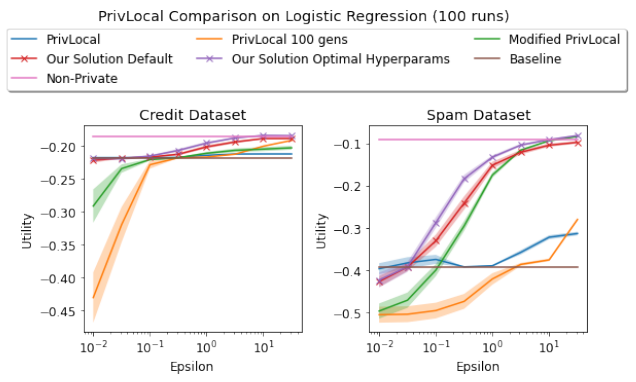


Figure 19: Comparing the utility of our DPGA to PrivLocal [50, 54].

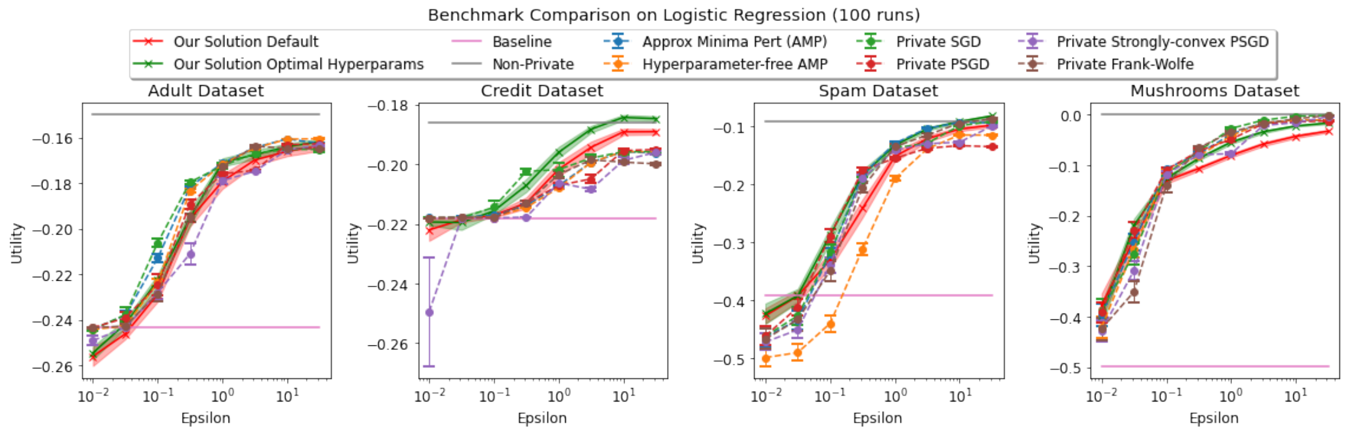


Figure 20: A comparison of our DPGA to the benchmark of private convex optimization techniques from Iyenger et al. [28].

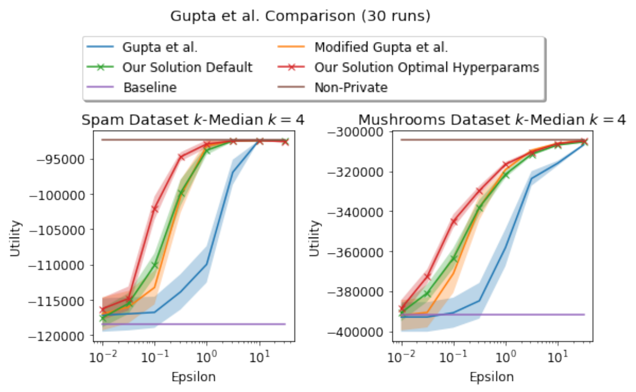


Figure 21: Comparing the utility of our DPGA to the local search of Gupta et al. [26].

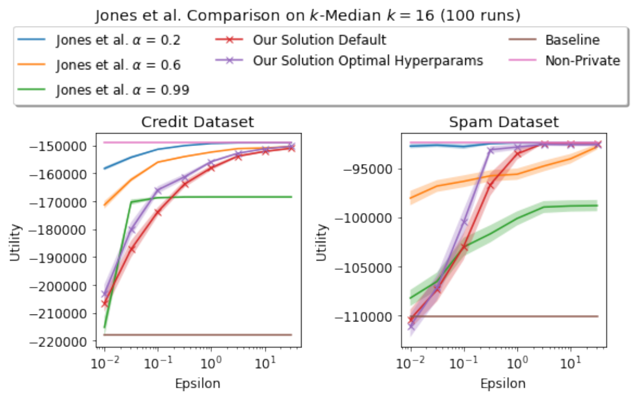


Figure 22: Comparing the utility of our DPGA against the state-of-the-art k-medians solution given by Jones et al. [29].