This is a pre-print version of the paper accepted for publication in Computers & Security, DOI: https://doi.org/10.1016/j.cose.2020.101753

Optimization-Based k-Anonymity Algorithms

Yuting Liang
McMaster University, Computing and Software
Hamilton, Ontario, Canada
liangy42@mcmaster.ca

Reza Samavi
McMaster University, Computing and Software
Hamilton, Ontario, Canada
samavir@mcmaster.ca

ABSTRACT

In this paper we present a formulation of k-anonymity as a mathematical optimization problem. In solving this formulated problem, k-anonymity is achieved while maximizing the utility of the resulting dataset. Our formulation has the advantage of incorporating different weights for attributes in order to achieve customized utility to suit different research purposes. The resulting formulation is a Mixed Integer Linear Program (MILP), which is NP-complete in general. Recognizing the complexity of the problem, we propose two practical algorithms which can provide near-optimal utility. Our experimental evaluation confirms that our algorithms are scalable when used for datasets containing large numbers of records.

KEYWORDS

anonymization; optimization; privacy; security; Mixed Integer Linear Program

1 INTRODUCTION

Big Data applications are increasingly related to privacy sensitive information, such as medical records, financial transactions and location footprints. Therefore, mining of such information requires special treatment of data to minimize privacy risks ideally without sacrificing the knowledge extraction value of the data mining task. A common practice to address the privacy concerns of data mining is the k-anonymity model that guarantees each individual record in the dataset is indistinguishable from at least $k-1$ other records, so the probability of re-identification is always less than or equal to $1/k$ [Samarati and Sweeney, 1998, Sweeney, 2002]. While other disclosure limitation techniques such as adding noise [Chawla et al., 2005], statistical obfuscation [Ardagna et al., 2011, Burridge, 2003], data perturbation [Liu et al., 2006] and more recently differential privacy [Dwork, 2011] have been developed to decrease privacy risks, k-anonymity model is relevant and desirable when individual records must remain truthful to their origins after the privacy preservation technique is applied. As a real use case, the co-author of this paper is a collaborator of a multi-disciplinary research team that is currently developing an active classifier to predict patients' cancer risk from medical imaging radiations [Sutton et al., 2018]. The diagnostic imaging data and health records, in this project, are originated from multiple medical institutions. The current practice for data sharing allows each institution to only share anonymized records with pseudo-identifiers and the burden on linking the health records, using the assigned identifiers, is on the research team. In such a scenario, the record linkage cannot be fruitfully applied if each source shares differentially private records. The truthfulness characteristic and the fact that the model was recommended in different privacy legislation and guidelines (such as HIPAA [U.S. Department of Health & Human Services, 2015] and FIPPA [Information and Privacy Commissioner of Ontario, 2016]) contributed to the wide adoption of k-anonymity and multiple algorithms (e.g., [Bayardo and Agrawal, 2005, Byun et al., 2007, Doka et al., 2015, El Emam and Dankar, 2008, Lee et al., 2017, Zhang et al., 2015]) have been devised for application of the technique to privacy-sensitive dataset prior to release.

The process of k-anonymization involves applying two major operations: generalization and suppression. In generalization the value of an individual attribute is replaced with a broader category (e.g., Age: 54 will be replaced by Age: 50-60). When generalization is not applicable or will not achieve k-anonymity, record-level or attribute-level suppression will be applied, where the entire record or the cell value will be deleted, respectively. The operation of generalization will impact the utility of the anonymized dataset variably depending on the degree of generalization (and in a worst case suppression). For example, Age: 54 can be generalized to Age: 50-60 or Age: 50-55, where more information loss is incurred in the former than in the latter. Thus, it is desirable that while k-anonymity is guaranteed, the objective of maximal utility be built into the anonymization process. The objective of this paper is to formulate the anonymization process as a mathematical optimization problem, where we seek to maximize data utility subject to k-anonymity constraints. Although in general the k-anonymization process is NP-hard [Meyerson and Williams, 2004], an optimal formulation is valuable as it can provide insights into the complexity of the problem, serve as a basis for developing heuristics and as a benchmark tool for future algorithms.

In this paper, we focus on the generalization operation of k-anonymity and define information loss for each attribute as the ratio between the range of anonymized values and the range of the possible values of the attribute. For example, if the permissible upper and lower bounds for Age is (0, 100), then generalization of the attribute value to 50-55 leads to an information loss value of 0.05 $(= (55-50)/(100-0))$ compared to information loss value of 0.1 when the attribute value is generalized to 50-60. Thus, the former generalization is favoured by the objective function. We also introduce a weight vector W for the attributes involved in the generalization operation such that the relative importance of each attribute in information loss can be customized by the user. We formulate the model as a Mixed Integer Linear Program (MILP). Given the complexity of MILP, we then propose two practical algorithms based on the intuition that rows of data that are *closer* to each other are likely to end up as equivalent rows. For example, if the dataset has just a single attribute with numerical values (e.g., Age), then neighbouring rows of the sorted column are relatively closer to each other and the operation of generalization will only involve grouping consecutive records. We capture the concept of *closeness* with a preprocessing step of sorting the dataset for numerical values. When we are facing multiple attributes, we use variance

for determining the relative order of the sorting among attributes. Although we are not able to provide a rigorous proof in this paper, we observe empirically that an attribute with less variance incurs more information loss when generalized compared to an attribute with larger variance. Therefore, the attribute with less variance will be sorted first.

Our first heuristic model is based on splitting and carrying over records between the split subsets (Split & Carry algorithm). The splitting allows us to solve a sequence of smaller sub-problems with MILP. By carrying over a portion of records from each sub-problem we provide a linkage between the otherwise disjoint sub-problems. Although the complexity of this algorithm is linear in the number of the records, it is very sensitive to the number of attributes or what is called *the curse of dimensionality* [Aggarwal, 2005]. To address the dimensionality problem, we develop a Greedy Search algorithm inspired by the algorithm presented in [Doka et al., 2015], however, with the difference that our algorithm generates anonymized datasets consistent with the general definition of k-anonymity (not based on the customized definition in [Doka et al., 2015]). We experimentally demonstrate that our practical algorithms can be used to anonymize datasets containing large numbers of records more efficiently with comparable information loss. According to the results of the experiments, we recommend the Split & Carry algorithm when the dataset is very large (millions of records) but the value of k is small (less than 8) and the number of attributes is also small (less than 6); otherwise we recommend our Greedy Search algorithm.

The contributions and structure of the paper is as follows. In Section 2 we introduce a mathematical formulation for the k-anonymization process, followed by discussions on several aspects of the model including complexity, weighting attributes, treatment of categorical data and the utility metric. We then present two practical algorithms for k-anonymity in Section 3, with test results in Section 4 to demonstrate their optimality and performance. In Section 4 we also present some benchmarking results with similar algorithms and scalability measure for our algorithms to show they can be used for anonymizing large datasets. We discuss a number of related work in Section 5. Finally we conclude the paper and provide potential research directions for future work in Section 6.

2 GENERAL OPTIMAL MODEL

In this section we first provide a preamble to the optimal model including the mathematical definitions of k-anonymity and the necessary mechanics for model formulation (Section 2.1). We then formulate k-anonymity as a general Mixed Integer Linear Program (MILP) and discuss its complexity in Section 2.2. In Section 2.3 we discuss an initial feasible solution which is the basis of our heuristic algorithms. Two additional aspects in our optimal model formulation, weighting the attributes and treating categorical attributes are discussed in Section 2.4 and Section 2.5, respectively.

2.1 Definitions

The following definitions can be found throughout various k-anonymity literature; here we state them in the context of our general optimal model.

DEFINITION 1: Quasi-identifiers are subsets of attributes of a dataset which can be used to deduce the identity of an individual. Note: In this paper all attributes are considered as quasi-identifiers.

DEFINITION 2: k-anonymity is achieved in a dataset if each record of the dataset cannot be distinguished from at least $k-1$ other records by quasi-identifiers.

DEFINITION 3: k-anonymization is the procedure of applying generalization and suppression to the quasi-identifiers of the dataset in order to achieve k-anonymity.

DEFINITION 4: Let x be an $n \times m$ matrix of records with each column corresponding to a quasi-identifier and each row corresponding to a record of some subject. Let x' be the matrix obtained from x by generalization, and x'_{ij} denote the generalized value of an entry x_{ij} of x for attribute $j \in J := \{1, 2, ..., m\}$ on record $i \in I := \{1, 2, ..., n\}$. We can write x'_{ij} as:

$$x'_{ij} = [y_{ij}, z_{ij}], \tag{1}$$

where y_{ij} and z_{ij} are the lower and upper bounds for the generalized values of x_{ij} and $y_{ij} \le x_{ij} \le z_{ij}$. Then the degree of information loss, D_{ij}, for the entry x_{ij}, is defined to be:

$$D_{ij} = \frac{z_{ij} - y_{ij}}{U_j - L_j}, \tag{2}$$

where U_j , L_j are the minimum and maximum permissible values of attribute j. Note that:

$$0 \le \frac{z_{ij} - y_{ij}}{U_j - L_j} \le 1. \tag{3}$$

When $y_{ij} = x_{ij} = z_{ij}$, information is not generalized and the entry has 0 information loss. When $y_{ij} = L_j$, $z_{ij} = U_j$, all information about this entry is lost, i.e., the attribute is suppressed.

LEMMA 1: Let v_{il} be a binary variable, let y_{ij}, z_{ij} be the lower and upper bounds that represent the generalized entry x'_{ij}, for $1 \le i, l \le n$, $i \ne l$, and $1 \le j \le m$. The following set of constraints guarantees k-anonymity in the matrix x':

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} y_{ij} \\ y_{lj} \end{pmatrix} \le M^y(i,l,j)(1 - v_{il}) \tag{4}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} y_{ij} \\ y_{lj} \end{pmatrix} \ge m^y(i,l,j)(1 - v_{il}) \tag{5}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{lj} \end{pmatrix} \le M^z(i,l,j)(1 - v_{il}) \tag{6}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{lj} \end{pmatrix} \ge m^z(i,l,j)(1 - v_{il}) \tag{7}$$

$$\sum_{l \ne i} v_{il} = k - 1, v_{il} \in \{0, 1\} \; \forall i \ne l \tag{8}$$

where the $M^y(i,l,j)$, $M^z(i,l,j)$ are upper bounds and $m^y(i,l,j)$, $m^z(i,l,j)$ are lower bounds on the pair $(y_{ij} - y_{lj}, z_{ij} - z_{lj})$

Proof. We provide a construction of the constraints. Recognizing that the process of k-anonymization involves selecting k rows and generalizing the entries in such a way that the k rows are indistinguishable from each other, we need to be able to represent equivalence between two rows. We want to show that v_{il} is the

binary variable that represents whether rows i and l are equivalent. Let y_{ij}, z_{ij} be the lower and upper bounds that represent the generalized entry x'_{ij}, and y_{lj}, z_{lj} represent those of x'_{lj}. We want to impose the relation that:

$$(v_{il} = 1) \rightarrow (y_{ij} = y_{lj} \,\&\, z_{ij} = z_{lj}). \tag{9}$$

Or equivalently for all $j \in J$:

$$(y_{ij} \neq y_{lj} \vee z_{ij} \neq z_{lj}) \rightarrow (v_{il} = 0). \tag{10}$$

In other words, rows i, l are considered equivalent if each attribute of row i has the same generalized lower and upper bounds as those of row l. We can formulate equation (9) as:

$$(v_{il} = 1) \rightarrow \left(\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} y_{ij} \\ y_{lj} \end{pmatrix} = 0 \,\&\, \begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{lj} \end{pmatrix} = 0 \right). \tag{11}$$

Using the ideas from [Mosek, 2018], which is in turn based on the books [Nemhauser and Wolsey, 1988, Williams, 1993], the above relation can be further decomposed as equations (4) - (7). It is also easy to see that relation (9) is equivalent to equations (4) - (7); when $v_{il} = 0$, we have:

$$m^y \leq y_{ij} - y_{lj} \leq M^y, \tag{12}$$

$$m^z \leq z_{ij} - z_{lj} \leq M^z, \tag{13}$$

which are true by definitions of m^y, m^z, M^y and M^z. When $v_{il} = 1$:

$$0 \leq y_{ij} - y_{lj} \leq 0, \tag{14}$$

$$0 \leq z_{ij} - z_{lj} \leq 0, \tag{15}$$

i.e. when $v_{il} = 1$, we have $y_{ij} = y_{lj}$ and $z_{ij} = z_{lj}$.

Note that the constraint for the other direction

$$\{(y_{ij} = y_{lj} \,\&\, z_{ij} = z_{lj}) \mid \forall j \in J\} \rightarrow (v_{il} = 1) \tag{16}$$

is not necessary. To see this, suppose we have two sets of equivalent rows S^1, S^2 such that:

$$v_{i_1 l_1} = 1 \,\forall i_1, l_1 \in S^1,\ v_{i_2 l_2} = 1 \,\forall i_2, l_2 \in S^2, \tag{17}$$

$$|S^1| = k - 1 = |S^2|. \tag{18}$$

Possibly there exists $l \in S^1 \cap S^2$, but $v_{i_1 i_2} \neq 1$ for $i_1 \in S^1 \setminus \{l\}$ and $i_2 \in S^2 \setminus \{l\}$, i.e., in x' we actually observe a larger set of equivalent rows:

$$S^1 \cup S^2,\ |S^1 \cup S^2| > k - 1, \tag{19}$$

but in this case, the model interprets the situation as two non-disjoint equivalent sets.

We have shown that the binary variables v_{il} as constructed represent equivalence between two rows. Then we can see immediately that for any row i to be equivalent to $k - 1$ other rows, all such binary variables associated with row i must sum to at least $k - 1$. Equation (8) provides the equality constraint, and the preceding discussion in equations (16) - (19) implies the inequality. □

DEFINITION 6: Let S^1, S^2 be two equivalence classes in x' such that $S^{12} := S^1 \cap S^2 \neq \emptyset$. We say the records in $S^1 \setminus S^{12}$ and $S^2 \setminus S^{12}$ are indirectly equivalent.

DEFINITION 7: Finally, a general optimization problem can be mathematically defined as of the form:

$$\begin{aligned} &\min_{x \in X} f(x) \\ &\text{Subject to } g(x) \leq 0. \end{aligned} \tag{20}$$

In our context, $f(x)$ is the objective function (information loss) and $g(x)$ is the set of k-anonymity and generalization range validity constraints.

2.2 Model Formulation

Using Definition 7 and Lemma 1, we formally state the optimization problem for k-anonymization as follows:

$$\min_{(y,z,v)} \sum_{i \in I, j \in J} \frac{z_{ij} - y_{ij}}{U_j - L_j} \tag{21}$$

Subject to

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} y_{ij} \\ y_{lj} \end{pmatrix} \leq M^y(i,l,j)(1 - v_{il}) \tag{22}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} y_{ij} \\ y_{lj} \end{pmatrix} \geq m^y(i,l,j)(1 - v_{il}) \tag{23}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{lj} \end{pmatrix} \leq M^z(i,l,j)(1 - v_{il}) \tag{24}$$

$$\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{lj} \end{pmatrix} \geq m^z(i,l,j)(1 - v_{il}) \tag{25}$$

$$\sum_{l \neq i} v_{il} \geq k - 1, v_{il} \in \{0,1\} \,\forall i \neq l \,\&\, i, l \in I \tag{26}$$

and

$$y_{ij}, z_{ij} \in \mathbb{Z},\ \forall j \in S^D \subseteq J \tag{27}$$

$$y_{ij}, z_{ij} \in \mathbb{R},\ \forall j \in S^C \subseteq J. \tag{28}$$

Where:
y_{ij}, z_{ij}: upper and lower bounds in generalizing attribute j.
v_{il}: binary variable for whether row i is equivalent to row l.
U_j, L_j: min and max permissible values of attribute j.
$S^C \subseteq J$: the set of indices for continuous attributes.
$S^D \subseteq J$: the set of indices for discrete value attributes.

We choose the bound constants for the i^{th} and l^{th} record of attribute j as follows:

$$M^y(i,l,j) = \max(x_{ij}, x_{lj}) - L_j \geq y_{ij} - y_{lj}, \tag{29}$$

$$m^y(i,l,j) = L_j - \max(x_{ij}, x_{lj}) = -M^y(i,l,j) \leq y_{ij} - y_{lj}, \tag{30}$$

$$M^z(i,l,j) = U_j - \min(x_{ij}, x_{lj}) \geq z_{ij} - z_{lj}, \tag{31}$$

$$m^z(i,l,j) = \min(x_{ij}, x_{lj}) - U_j = -M^z(i,l,j) \leq z_{ij} - z_{lj}. \tag{32}$$

Remark: Note that by our choice of lower bound constants, it might seem that inequalities (5) and (7) are redundant as $m^y = -M^y$ and $m^z = -M^z$; in fact both constraints are needed in order to ensure relation (9) is satisfied. Suppose we remove inequality (5). When $v_{il} = 1$, to satisfy inequality (4) we can actually have $y_{ij} < y_{ij}$, which would not satisfy relation (9).

We formulated the model described above as a Mixed Integer Linear Program (MILP) which is hard in general [Papadimitriou, 1981, von zur Gathen and Sieveking, 1978]. There exist subclasses of MILPs which have better complexity, but measuring the complexity of any MILP itself is a difficult task and relies on heuristics and conditions on the constraint matrix [Genova and Guliashki, 2011]. Meyerson and Williams showed that the general k-anonymization

Name	Age	Sex	Zipcode	Disease
Mary	37	F	22071	Pneumonia
Alice	35	F	22098	Diabetes
Betsy	36	F	23061	Anemia
David	61	M	55107	Pneumonia
Tom	63	M	55099	Diabetes
James	66	M	55324	Diabetes
Eric	63	M	55229	Diabetes

(a) Original dataset ([Lee et al., 2017])

Age	Sex	Zipcode
[35-37]	[0-0]	[22071-23061]
[35-37]	[0-0]	[22071-23061]
[35-37]	[0-0]	[22071-23061]
[61-66]	[1-1]	[55099-55324]
[61-66]	[1-1]	[55099-55324]
[61-66]	[1-1]	[55099-55324]
[61-66]	[1-1]	[55099-55324]

(b) 3-anonymized outcome

Table 1: Example of anonymized Electronic Health Record using our model

problem (using suppression) is hard using Graph Theory techniques [Meyerson and Williams, 2004].

MILPs are best solved using Branch-and-Cut (Branch-and-Bound with Cutting Planes) methods [Genova and Guliashki, 2011]. Therefore, it is not surprising that many existing algorithms aim to solve a relaxed version of the problem using some variant of Branch-and-Bound [Bayardo and Agrawal, 2005, Lee et al., 2017]. Although MILPs are hard in general, there are commercial implementations of Branch-and-Cut which are quite efficient (e.g., Gurobi [1] or CPLEX [2]); there are also heuristics for selecting *better* nodes at which to branch out (e.g., [Bayardo and Agrawal, 2005]).

To confirm whether commercial solvers can provide any breakthrough to the complexity of our optimal formulation, we implemented our MILP model in Python with state-of-the-art optimization solver Gurobi, which implements the Branch-and-Cut method in parallel [Gurobi Optimization, LLC., 2018]. Gurobi and CPLEX are arguably the best available MILP solvers [Mittelmann, 2018]. We have also implemented the general model with CPLEX and open source solver CBC [3]; we ran a few simple tests. We take a small example from [Lee et al., 2017] in Table 1a to demonstrate the outcome of our model in Table 1b, where we anonymized three quasi-identifiers (Age, Sex, and Zipcode) using our optimal MILP model.

Although we found comparable performance between Gurobi and CPLEX which both are 30x faster than CBC, even the fastest solver struggles to solve the k-anonymity MILP on a dataset with the size of 100 records and 8 attributes in a reasonable amount of time. Therefore, devising practical algorithms is necessary for solving k-anonymity problems.

2.3 Initial Feasible Solution

In light of the complexity of the optimal model, we provide an initial feasible solution to the solver to prune some of the sub-optimal nodes early on in the search. The idea behind our initial feasible solution is that rows of records that are *closer* to each other are likely to end up as equivalent rows. For example, if we have just a single column of numbers, then neighbouring rows of the sorted column are close together. However, when we have multiple columns, we need to determine the order of the attributes with which we compare the tuples.

The individual information loss component of attribute j

$$D_j = \sum_{i=1}^{n} D_{ij} = \sum_{i=1}^{n} \frac{z_{ij} - y_{ij}}{U_j - L_j} \tag{33}$$

[1] http://www.gurobi.com/

[2] https://www.ibm.com/products/ilog-cplex-optimization-studio

[3] https://projects.coin-or.org/Cbc

index	AGE	SEX	INJ_SEV	DRINKING
0	64	2	4	0
1	29	1	0	0
2	42	2	0	0
3	41	2	4	1
4	53	1	2	1
5	59	1	4	0
6	49	1	4	0
7	59	1	2	0
8	80	1	4	0
9	50	1	4	0
10	64	1	3	0
11	55	1	0	0
12	25	1	0	0
13	42	1	4	0
14	33	1	2	0
15	31	2	2	0
16	68	1	3	0
17	20	2	4	0
18	40	1	4	1
19	18	1	4	0

(a) Original dataset

index	AGE	SEX	INJ_SEV	DRINKING
12	25	1	0	0
1	29	1	0	0
11	55	1	0	0
14	33	1	2	0
7	59	1	2	0
10	64	1	3	0
16	68	1	3	0
19	18	1	4	0
13	42	1	4	0
6	49	1	4	0
9	50	1	4	0
5	59	1	4	0
8	80	1	4	0
2	42	2	0	0
15	31	2	2	0
17	20	2	4	0
0	64	2	4	0
4	53	1	2	1
18	40	1	4	1
3	41	2	4	1

(b) Sorted by variance

Table 2: FARS dataset on 4 attributes

depends largely on the upper-lower bound gap $U_j - L_j$ where the information loss is inversely proportional to the gap. Thus, for columns with smaller upper-lower bound gaps, we would like their entries to appear as sorted as possible in the matrix; however, it is likely that we have multiple columns with the same gaps (e.g., when we have multiple binary attributes). For this reason, we opted to use ascending variance for determining the order. In general a larger upper-lower bound difference contributes to a larger variance in the column. Moreover, for attributes with the same upper-lower bound gaps, because the values of their entries lie within intervals of the same length, a larger variance implies the values are more dispersed on the interval; whereas a smaller variance implies there are more points centred around the mean, which leads to larger information loss when we try to bring such points to equivalent sets with those points near the boundaries. Therefore, attributes with smaller variances will be sorted first and attributes with larger variances last. We demonstrate the impact of sorting based on variance with the following example.

Consider an example of a dataset from the FARS [US Department of Transportation, 2016] database containing traffic accidents data. This dataset contains 20 records with 4 attributes (AGE, SEX, INJ_SEV (injury severity) and DRINKING). In Table 2a we have the original dataset, and in Table 2b we show the dataset sorted with comparison order determined by increasing variance. The actual equivalence classes determined by our optimal MILP are {0,2,15,17}, {1,11,12}, {3,4,18}, {5,6,8,9}, {7,10,16}, and {13,14,19}. Notice most of the equivalence classes appear as consecutive rows in the sorted matrix.

2.4 Weighting Attributes

Another observation made is that in our current optimal modelling of k-anonymity, all attributes are treated equally in their impact on information loss encoded as the objective function. However, generally speaking utility is a subjective measure and depending on the use, some researchers might prefer to have less information loss in certain attributes even if it is at the expense of other attributes. By placing weights on the attributes, the requester of data has some control over which attributes he/she would want to have tighter generalized bounds. We use W to assign weight w_j for all $j \in J$ (the

set of indices of all attributes). W is used to adjust the likelihood of an attribute being generalized/suppressed in the anonymization process, where a larger weight means the attribute is less likely to be generalized/suppressed. We provide an example in Appendix A to demonstrate the effect of applying unequal weights to the attributes. Then the objective function in equation (21) is adjusted as follows to include the attribute weights:

$$\min_{(y,z,v)} \sum_{i\in I, j\in J} \left[w_j \frac{z_{ij} - y_{ij}}{U_j - L_j} \right]. \tag{34}$$

We have to also add the following additional constraint to the model:

$$\sum_{j\in J} w_j = 1. \tag{35}$$

Applying weights to the initial feasible solution (Section 2.3) requires an additional step. When we have a vector of unequal weights, we need to scale the variances of the columns to reflect the weights in the objective function. Variances for attributes with smaller weights need to be scaled by a larger amount than those with larger weights. Therefore, we can adjust the variance function with

$$wVar(c_j) := \frac{Var(c_j)}{w_j^2}, \tag{36}$$

for column c_j with weight $w_j, j \in J := \{1, ..., m\}$.

2.5 Categorical Data

The objective function in our formulation suffers from one drawback. The information loss calculated as in Definition 4 would only make sense if the subtraction between two permissible values is defined, i.e., a clear distance metric is defined for the attribute. This is also a problem for the initial feasible solution in Section 2.3 where sorting is undoubtedly based on a distance metric. For binary data, there is a natural distance metric; however, for some categorical data with more than two permissible values, the distance metric is not clear. One way to mitigate the problem is to number the permissible values of a categorical attribute in a way that implies a sense of distance. For example, if we have Countries as an attribute, we can number the countries based on their geographical distances to some reference country (e.g. Canada) or language similarity. If the attribute is important to the researcher, and a weak sense of distance is not adequate, then we can split the categorical data into binary data where each new attribute corresponds to one permissible value of the original categorical attribute. When we are converting categorical data to binary data, we recommend adjusting the weight by multiplicative factor of $1/|H|$, where H is the set of permissible values of the categorical attribute. This puts the sum of utility loss of the constituent binary attributes to be within the range [0,1] and thus the original categorical attribute still satisfies equation (3) of Definition 4. This treatment of categorical data is similar to the Global Certainty Penalty (GCP) utility function in [Doka et al., 2015, Wong et al., 2010] and in fact there is a natural one-to-one correspondence from one to the other. Please see Appendix B for detailed definition of GCP.

2.6 On Utility Metric

For anonymization on relational datasets (or any dataset where each record can be represented as a tuple of attributes), there are utility metrics which are more general and applicable to different data types, such as the Discernibility Metric (DM) [Bayardo and Agrawal, 2005] and Classification Metric (CM) [Iyengar, 2002]; however, these metrics do not measure the amount of information loss contributed by the individual records which might be important to the user of the anonymized data. Information loss metrics are used in various literature [Doka et al., 2015, Ghinita et al., 2009, Wong et al., 2010], especially when generalization is an operation used to anonymize the data. In the formulation of our model in Section 2.2, we have defined our utility metric to be the sum of the degree of information loss over each anonymized entry x'_{ij} as we aim to minimize the total amount of information loss in our anonymization process, i.e., maximize the utility of the anonymized data. The linearity of this metric is also desirable as linear optimization problems are among the easiest subclass of optimization problems in terms of complexity and availability of tools. We understand that the usefulness of this utility function might be limited in some application domains. As described in Section 2.4, we introduced W to help customize the utility function but this can still be an issue when a more complicated utility function beyond an optimized generalization is being sought. In theory our general optimal formulation can be adapted to different utility metrics; however, since sorting is an important part of our practical algorithms, ideally the utility metric should respect the ordering in the permissible values of each attribute, i.e., for permissible values $a_1 < a_2 < a_3$ the metric should favour anonymized outcome $[a_1, a_2]$ to $[a_1, a_3]$.

3 PRACTICAL ALGORITHMS

We have implemented the general model with the best available MILP solver; however, as we have discussed in the previous section, the complexity of the model quickly leads to undesirable performance as we increase the number of records, making the model unsuitable for practical use despite offering optimal utility. We also provide an initial feasible solution which can help in pruning some of the sub-optimal nodes early on in the search. In general it is not easy to see how much performance improvement such initial solutions can provide; moreover as the size of the problem increases, the majority of the time for the solver is spent trying to improve the bounds between the so-far best feasible node and the relaxed optimum. Even if we feed in an initial solution that is optimal, the solver might still need to traverse many nodes and solve many relaxed problems before determining that the solution was already optimal. Therefore, in practice we cannot rely on solving the general model to anonymize any reasonably sized dataset. In this section, we use the initial feasible solution to devise two different practical algorithms with improved utility over the initial feasible solution.

3.1 Split & Carry Algorithm

The first practical algorithm based on the initial feasible solution is the Split & Carry algorithm. We use the initial solution to split the original problem into smaller sub-problems with manageable sizes, i.e. sub-problems that are solvable using the general MILP.

As discussed in Section 2.3, consecutive rows in a sorted matrix are likely to be equivalent rows in an optimally k-anonymized dataset. Thus, we expect the rows that are far apart in a sorted matrix are unlikely to be equivalent, and can be placed into different sub-problems to be solved by the general model. This idea is the basis of our Split & Carry algorithm as described in Algorithm 1.

The input values to the algorithm are:

(1) $arrType$ an array describing the data type of each column (from {Integer, Continuous}, used in Gurobi solver)
(2) $x = [x_{ij}]_{i \in I, j \in J}$, the dataset in a matrix form
(3) $U = [U_j]_{j \in J}$, $L = [L_j]_{j \in J}$, upper and lower bounds of each column
(4) $W = [w_j]_{j \in J}$, weights for each column
(5) k, minimum number of equivalent rows desired
(6) S, minimum number of k-sets[4] in each sub-problem ($S \geq 2$).

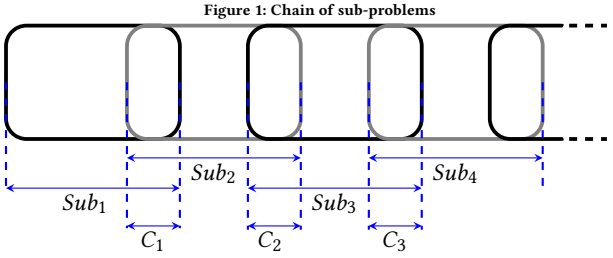
Data: $x, U, L, W, k, S, arrType$
Result: $\mathbf{A}, \mathbf{f}$
1 $VAR = [VAR]_{j \in J} \leftarrow$ compute variances of all attributes;
2 $\tilde{x} \leftarrow sort(x, VAR)$;
3 $C_0 \leftarrow \emptyset$;
4 $\mathbf{f} \leftarrow 0$;
5 **for** $m := 1$ *to* $\lceil \frac{n}{k \cdot S} \rceil$ **do**
6 $\quad \mathbf{Sub_m} \leftarrow C_{m-1}$ + Read the next $k \times S$ rows from $\tilde{x}$;
7 $\quad (\mathbf{A_m}, \mathbf{f_m}) \leftarrow$ Solve($\mathbf{Sub_m}, arrType$) optimally (Section 2.2);
8 $\quad \mathbf{f} \leftarrow$ update($\mathbf{f}, \mathbf{f_m}$);
9 $\quad \mathbf{A} \leftarrow$ update($\mathbf{A}, \mathbf{A_m}$);
10 $\quad \mathbf{C_m} \leftarrow$ rows in equivalence classes of last k rows in $\mathbf{A_m}$;
11 $\quad m = m + 1$;
12 **end**
13 **return** ($\mathbf{A}$, $\mathbf{f}$)

Algorithm 1: Split & Carry Algorithm

Besides the first five input values which are common to the optimal model, we use the parameter S to adjust the start size of the sub-problems; S captures the minimum number of k-sets to be included in each sub-problem. S is a user parameter with value greater than 2 because there is only one trivial k-set when $S = 1$. The value selected for S depends on the desired efficiency and utility of the k-anonymization process. When S is small we are limiting the potential space from which we make k-sets; but since S determines the minimum size of the sub-problem a large value will increase the complexity of each sub-problem. We will demonstrate in our experiments that a small S suffices when k is small and the number of records is large. In general the running times of the sub-problems depend largely on the statistical properties of the dataset, when we have a large dataset we are likely to find many records that are similar which means each sub-problem to be solved has a small solution space; they also depend on the availability of computational resources because MILP solvers are often multi-threaded (e.g. Gurobi). In Fig. 5a and Fig. 5b we demonstrate the effect of S on a reasonably-sized dataset in terms of utility and running time.

[4] k-set: shorthand for a set of equivalent records of size k.

Figure 1: Chain of sub-problems



In Fig. 1 we present an illustration of the Split & Carry algorithm. The sub-problems are indexed as $(Sub_1, Sub_2, ..., Sub_t)$, $t = \lceil \frac{n}{k \cdot S} \rceil$. We have also the concept of carry-over records in this algorithm, represented as $(C_1, C_2, ...C_{t-1})$. We can visualize the algorithm as being a chain of sub-problems, where the *locks* that chain together sub-problems are the boundary carry-over records. Records in the sorted matrix that are near the boundary of two consecutive sub-problems can be similar to records in both sub-problems; therefore, after we have determined the equivalence class for such a boundary record in a sub-problem, we carry the entire equivalence class to the next sub-problem. A careful consideration is needed to ensure that the records indirectly equivalent to the boundary record can satisfy k-anonymity on their own, i.e. the total number of all indirectly equivalent records to this boundary record must be at least k; otherwise we have to also carry these records to the next sub-problem. Because the number of carry-over records from the previous sub-problem can be different, each individual sub-problem may end up with a different run-time size. In the worst case scenario, it is possible that an entire sub-problem gets carried over to the next. For example, suppose we have $k = 3$ with $S = 2$. We index the records in this sub-problem as $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, one of two scenarios can happen: 1) The last k records form an equivalence set within themselves. 2) Each of the first k records is equivalent to at least one of the last k records. In the first scenario the equivalence classes are exactly $\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6\}$. In the second scenario we must have an equivalence class of the form $\{x_1, x_i, x_l\}$, $\{x_2, x_i, x_l\}$ or $\{x_3, x_i, x_l\}$ with $\{x_i, x_l\} \subset \{x_4, x_5, x_6\}$; in this case all of the records will be carried over to the next sub-problem. Intuitively as the size of the next sub-problem increases, it becomes unlikely that all of its records will be carried over further. Thus, we expect the sizes of the sub-problems to be bounded. We provide an upper-bound on the sizes in Lemma 2. Note that a more general implementation would be to carry over equivalent rows to the last l records, where $l \neq k$ is also a customizable parameter. In the current implementation, k affects the sub-problem sizes which becomes a problem as k gets larger.

In Line 1 of Algorithm 1, we compute the variances of each column, where we then sort the matrix across entries with comparison order equal to the order of increasing variance in Line 2. We initialize the first set of carry-over records C_0 to be empty in Line 3 and initialize the objective value (i.e. total information loss) f to be 0 in Line 4. We then loop over the chain of sub-problems in Line 6 - Line 11; for each sub-problem Sub_m we get the set of next $k \times S$ records and add to it the carry-over records C_{m-1} from the

previous sub-problem in Line 6, where we then solve the resulting sub-problem by the general MILP to obtain anonymized outcome A_m and objective value f_m in Line 7; we then update the total information loss f by adding to it the part of the objective value f_m corresponding to non-carry over records (Line 8) and also store the subset of corresponding anonymized records in A_m (Line 9) to A. In Line 10 we get the set of carry-over records C_m, which are records equivalent to the boundary records (i.e. the last k records). We repeat Line 6 to Line 11 until we solve all sub-problems. We include a worked example in Appendix C to demonstrate the steps of this algorithm.

Remarks: The reason for carrying over records equivalent to the boundary records is two-fold: we have described that boundary records can be close to both sub-problems, and as equivalence is transitive this is true for their equivalent records as well. The equivalent rows also serve as initial equivalent rows to the next sub-problem, i.e. the optimal solution for the next sub-problem can in fact determine that the records carried over from a previous sub-problem are not closer to the new subset of records, and output the carry-over records as sets of equivalent rows the way they were fed in. The lower and upper bounds used in the sub-problems must be the same as those computed in the original problem, as the sub-problems should be optimized with respect to the original objective function.

LEMMA 2: The size of each sub-problem is bounded by $k \times (2k - 1 + S)$.

Proof. In each sub-problem, at the step where we carry over records the worst scenario happens when the last k records are in distinct equivalence classes. For each boundary record, we carry over only records that are necessary, i.e., we carry over exactly k records if the remaining indirectly equivalent records satisfy k-anonymity on their own (with size >= k). Thus, the worst case happens when we have $(k - 1)$ indirectly equivalent records to each boundary record, then we must also carry over the entire set of such indirectly equivalent records. That is, we carry over $k \times (k + (k - 1))$ records to the next sub-problem, which contains $k \times S$ records before the carry-over. Thus, we have $k \times (2k - 1 + S)$ records in the resulting sub-problem in the worst possible case. □

In practice we expect that we should rarely encounter a sub-problem with size equal to its possible upper-bound; we provide the actual distributions of the sizes of the sub-problems for the tests of Fig. 5 in Appendix D, Table 8.

THEOREM 1: An upper-bound for the complexity of the Split & Carry algorithm is:

$$O(\lceil \frac{n}{k \times S} \rceil \times 2^{p(k \times (2k-1+S) \times (\frac{k \times (2k-1+S)-1}{2} + 2m))}). \quad (37)$$

Proof. Consider the complexity of the general MILP. Since non-integer linear programs (LP) have polynomial complexity [Dobkin and Reiss, 1980], in the worst case scenario the general MILP has to search all of the nodes and solve an LP at each node. Thus, an upper bound for the complexity of the general MILP is $O(2^{p(N)})$ where $p(\cdot)$ is a polynomial function, N is the number of variables in the problem. In our formulation $N = \binom{n}{2} + 2 \times m \times n = \frac{n(n-1)}{2} + 2 \times m \times n$ for an input matrix of size $n \times m$. The Split & Carry algorithm solves a series of sub-problems each with at most $k \times (2k - 1 + S)$ records as shown in Lemma 2, where the complexity of each sub-problem is bounded by:

$$O(2^{p(k \times (2k-1+S) \times (\frac{k \times (2k-1+S)-1}{2} + 2m))}). \quad (38)$$

Since we have to solve $\lceil \frac{n}{k \times S} \rceil$ such sub-problems, this gives expression (37). □

Early Termination: From expression (38) we see that the running time for solving a sub-problem can increase quickly as we increase k, m or S. A feature of the Gurobi Solver is that it supports Early Termination, i.e., it can halt the search for feasible nodes after a pre-defined condition is reached. Thus, it is possible to set a strict time limit on the Split & Carry algorithm. For example, if one would like the algorithm to run for no more than T seconds (ignoring overhead costs other than the solving of the sub-problems), one can set the "TimeLimit" parameter of the Gurobi solver to be $T \Big/ \lceil \frac{n}{k \times S} \rceil$ seconds. Inevitably, setting a time limit would reduce optimality of the solution.

Categorical attributes encoding: We have talked about the treatment of categorical data in our model in Section 2.5. If we decide to split up the categorical attribute into binary attributes we may encounter a limitation of the Split & Carry algorithm, as this will increase the number of attributes and the sizes of the sub-problems for this algorithm.

3.2 Greedy Search Algorithm

In this subsection we describe another algorithm which improves upon the initial feasible solution. As we observe in expression (37), while the complexity of Split & Carry algorithm scales linearly in the number of records as opposed to exponentially in the case of the general optimal problem, the complexity still scales exponentially in terms of the number of attributes and k. In this section we provide a greedy search algorithm which is not plagued with the *curse of dimensionality* [Aggarwal, 2005]. This algorithm uses similar ideas as the algorithms presented by Doka et al. [Doka et al., 2015], but it produces anonymized outcomes consistent with our formulation in Section 2.2 (i.e. containing sets of at least k equivalent records) instead of outcomes appearing as in the *freeform* formulation of Doka et al. [Doka et al., 2015].

In the Doka et al. algorithms, k outer iterations are performed where at each iteration the original dataset structure is transformed into a complete bipartite graph[5]. In our algorithm, we perform a series of k inner iterations. We again start from the initial feasible solution containing records sorted according to the order of increasing variance. Then for each indexed record x_i in the sorted list, we construct the k-set (Eq_i) by iteratively choosing from the remaining records and adding to it, and subsequently removing the chosen record from the remaining set. In each iteration, we aim to generate the smallest objective value, $f_i := f(Eq_i)$, until

[5]A graph is complete bipartite if the set vertices can be partitioned into two disjoint sets where there is an edge from each vertex in one set to each vertex in the other.

Eq_i contains k records. Two advantages of our algorithm are: 1) it does not work with complete bipartite graphs and as such does not need to store weights for $n \times n$ edges; 2) it does not require a backtracking[6] process and thus does not need to store information from the previous states. Given these advantages the algorithm is more memory efficient and can be applied over datasets containing large numbers of records. However, we remark that both our Greedy Search and the Doka et al. algorithms [Doka et al., 2015] have the limitation that they aim to find equivalence classes of size k. This is a major limitation of the greedy algorithm approach as in order to find a globally optimal solution the set should be allowed to contain many equivalence classes of different sizes greater than k. Our Greedy Search algorithm is described in Algorithm 2. Since this algorithm is not constrained by the number of attributes in terms of performance, we have the option of decomposing categorical attributes into vectors of binary attributes which is desirable to maintain better utility for categorical data.

The inputs to our greedy search algorithm are:

(1) $arrType$ an array describing the data type of each column (from {Integer, Continuous} or {Categorical, Numerical})
(2) $x = [x_{ij}]_{i \in I, j \in J}$, the dataset in a matrix form
(3) $U = [U_j]_{j \in J}$, $L = [L_j]_{j \in J}$, upper and lower bounds of each column
(4) $W = [w_j]_{j \in J}$, weights for each column
(5) k, minimum number of equivalent rows desired

In Line 1 and Line 2 of Algorithm 2, we again compute the variances of each column, and sort the records as in the initial feasible solution and Algorithm 1. In line 3 we have an optional step to transform the columns with categorical attributes into vectors of columns of binary attributes. A discussion on how this is represented can be found in Appendix B. In Line 4 we initialize the total objective value f to be 0, and the collection of equivalent sets A to be empty. In Line 5 - Line 18 we loop over each remaining record of the sorted list; we initialize the equivalence class of record x_i to be the set containing itself in Line 6, and the objective value f_i for the equivalence class Eq_i to be 0 in Line 7. In Line 8 we remove the record x_i from the remaining records. Then we perform $k-1$ iterations in in Line 9 - Line 17. In Line 10 - Line 17 we compute the objective value f_i^j for each remaining record x_j, so that in Line 13 we can find the record that corresponds to incurring min loss if it is added to Eq_i, and it is then added to Eq_i in Line 14. We then update the objective value f_i to be the information loss corresponding to the new Eq_i in Line 15, and remove the new member of Eq_i from the remaining records in Line 16. At the end of $k-1$ iterations we obtain Eq_i of size k, and we update the total information loss f by adding f_i to it in Line 18. We update the collection of equivalence classes A with the new found Eq_i in Line 19.

Convergence: Since we are always removing k indices from $I := \{1, 2, ..., n\}$, when the size n is a multiple of k, convergence is clear; if n is not divisible by k, for each remaining index j after the $\lfloor n/k \rfloor$ iterations, we distribute x_j to the existing k-set that would incur the least utility loss if we were to add x_j to it. Thus, the set of indices I will be exhausted.

[6]Backtracking describes a step that involves revisiting records that have been put into an equivalence class in a backward fashion as in [Doka et al., 2015].

Data: $x, U, L, W, k, arrType$
Result: $\mathbf{A}, \mathbf{f}$
1 $VAR = [VAR]_{j \in J} \leftarrow$ compute variances of all attributes;
2 $\tilde{x} \leftarrow sort(x, VAR)$;
3 (Optional per $arrType$)Transform the columns containing categorical data into vectors of columns containing binary data;
4 $A \leftarrow \emptyset$; $\mathbf{f} \leftarrow 0$;
5 **for** $x_i \in \tilde{x}$ **do**
6 $Eq_i \leftarrow \{x_i\}$, initialize equivalent set Eq_i for x_i;
7 $\mathbf{f_i} \leftarrow 0$;
8 remove x_i from $\tilde{x}$;
9 **for** $l = 1$ *to* $k-1$ **do**
10 **for** $x_j \in \tilde{x}$ **do**
11 compute and store objective value $f_i^j := f(x_j \cup Eq_i)$ using $arrType$;
12 **end**
13 find $x_{j'}$ that would give lowest f_i^j;
14 add $x_{j'}$ to Eq_i;
15 $\mathbf{f_i} \leftarrow$ update$(\mathbf{f_i}, f_i^{j'})$;
16 remove $x_{j'}$ from $\tilde{x}$;
17 **end**
18 $\mathbf{f} \leftarrow$ update$(\mathbf{f}, \mathbf{f_i})$;
19 $\mathbf{A} \leftarrow$ update$(\mathbf{A}, Eq_i)$
20 **end**
21 **return** $(\mathbf{A}, \mathbf{f})$

Algorithm 2: Greedy Search Algorithm

THEOREM 2: The complexity of the Greedy Search algorithm is:

$$O\Big(\sum_{p=0}^{\lfloor n/k \rfloor} \{(k-1) \times (n - (p-1) \times k) + n)\} \tag{39}$$

Proof. Each time we loop over an index we remove k entries (including itself) from the dataset x. Thus, the outermost for-loop is repeated at most n/k times. At the p^{th} iteration of the outermost for-loop, we need to compute the min of $n - (p-1) \times k$ objective values, thus, the complexity of the middle for-loop is $O((k-1) \times (n - (p-1) \times k))$. If n is not divisible by k, for each remaining record we distribute it to the equivalence class that would incur the least utility loss by adding this record; since there are n/k equivalence classes and at most $k-1$ remaining records this last step has complexity $O(k \times \frac{n}{k})$. □

Given the above theorem, although greedy algorithm may provide relatively less favourable solutions, its complexity is much less dependent on the number of attributes and is bounded by $O(n^2)$ regardless of the choice of k.

Attribute	Cardinality	[Min,Max]	DataType
\Gender	2	[1,2]	Categorical/Integer
\Age	75	[16,90]	Numerical/Integer
\Marital Status	6	[1,6]	Categorical/Integer
\Race	7	[1,7]	Categorical/Integer
\Birthplace	83	[1,426]	Categorical/Integer
\Education Level	17	[4,20]	Numerical/Integer
\Work Class	7	[13,29]	Categorical/Integer
\Occupation	47	[4,79]	Categorical/Integer

Table 3: CENSUS Dataset

4 EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of our practical algorithms in terms of optimality, performance and scalability. In Section 4.1 we describe the dataset we have used for running the experiments and the experimental environment setup. In Section 4.2 we present the first set of experiments and provide the results from our algorithms along with results from the optimal solution (i.e. solution of the general MILP). In Section 4.3 we provide benchmark results of our algorithms against the other algorithms in terms of utility and running time. In Section 4.4, we report the results of the last set of experiments focusing on the scalability aspect of the algorithms. We selected three classes of k-anonymity algorithms for this study; namely heuristic-based algorithms by [Doka et al., 2015], a partitioning-based algorithm by [LeFevre et al., 2006] and a clustering-based algorithm by [Byun et al., 2007]. We provide a discussion and interpretation of the results in Section 4.5.

4.1 Experimental Setup

The main database for our tests is from IPUMS USA, consisting of 500,000 records of U.S. census data [University of Minnesota, 2018] (hereafter referred to as the CENSUS dataset). For each experiment we extracted datasets of various sizes from the main dataset as described in each experiment. Table 3 describes the characteristics of the main datasets with 8 attributes in terms of the attribute name, type (numerical or categorical), min/max values and number of distinct values available for each attribute. The characteristics of the extended datasets with up to 35 attributes are described in Table 10 in Appendix E. The tests were performed on machines with 8 CPUs and 7.2 - 8 GB of RAM. The results were validated to be consistent across the machines. Moreover, tests appearing in the same figure were strictly run on the same machine.

To avoid database bias, we have reported the results of our experiments using an alternative database in Appendix F.

4.2 Optimality

To demonstrate optimality, we benchmark our practical algorithms (Split & Carry and Greedy Search) against the optimal formulation (general MILP with Gurobi solver) in terms of information loss, and from different dimensions. In Fig. 2a and Fig. 2c we show the averages of ratios of objective values from our algorithms over those from the MILP when the number of records grows and when the number of attributes grows, respectively. To reduce the impact of variability in the datasets the averages are computed over 20 datasets randomly drawn from the CENSUS database. The number of attributes is set to 4 for the results illustrated in Fig. 2a and Fig. 2b, and the number of records is set to 20 for the results shown in Fig. 2c and Fig. 2d. We observe that consistently our Split & Carry achieves better utility than Greedy Search. An interesting finding from this experiment is that increasing the number of attributes does not increase loss in optimality for both algorithms.

We also show the averages of ratios for the running times on the same datasets when the size of dataset increases and when the number of attributes grows in Fig. 2b and Fig. 2d, respectively. We see that there are significant performance benefits in our algorithms especially as we increase the size of the dataset. In particular we observe that the running time for Greedy Search compared to the optimal solution is negligible. Although we see in Fig. 2 that Greedy Search has significantly better performance than Split & Carry, it is not generally the case and we show this as we further compare the running times of these two algorithms for large datasets in the scalability section. The difficulty in measuring the complexity of the Split & Carry algorithm comes from the difficulty in providing an exact or even tighter upper bound in the complexity of any specific instance of the MILP. The complexity for Split & Carry we've provided in Theorem 1 is merely an upper bound, and in practice as the size of our dataset increases we expect a much better performance than what we presented in Theorem 1. Nonetheless, expression (37) in Theorem 1 provides the relationship between the number of records n and complexity of the Split & Carry; that is, the complexity scales linearly in the number of records, which we also demonstrate in the scalability section.

4.3 Benchmarking

We first benchmarked our algorithms with the three algorithms presented by Doka et al. [Doka et al., 2015], hereafter we refer to these as: Doka Greedy, Doka SortGreedy and Doka Hungarian. We implemented all three algorithms in Python. While we have confirmations from the first author regarding specific details in the implementations, we acknowledge that there could be implementations that can give slightly better running times than what we will display here; for example we had used the Munkres module [7] for the Hungarian algorithm combined with the iteration scheme described in [Doka et al., 2015] for the implementation of Doka Hungarian. Despite potential differences in the implementation details, however, two important facts provide a sound basis for any conclusions we draw from our experiments:

(1) As consistent with the results in [Doka et al., 2015], in general the Doka Hungarian algorithm provides the best utility among all three Doka algorithms.
(2) By theoretical construction, our Greedy Search algorithm has better complexity than all three Doka algorithms.

We also benchmarked our algorithms against the Mondrian Multidimensional k-Anonymity algorithm [LeFevre et al., 2006] (hereafter referred to as Mondrian), and we were able to download the program directly from [UT Dallas Data Security and Privacy Lab, 2012]. The Mondrian algorithm is a partitioning-based algorithm in which the dataset is repeatedly partitioned into disjoint subsets according to some statistic in a tree-like manner. The Mondrian algorithm can produce anonymized outcome as generalized ranges,

[7] munkres 1.0.12: https://pypi.org/project/munkres/

Figure 2: Running time and object values of Split & Carry and Greedy Search against optimal MILP, as number of records or attributes change.

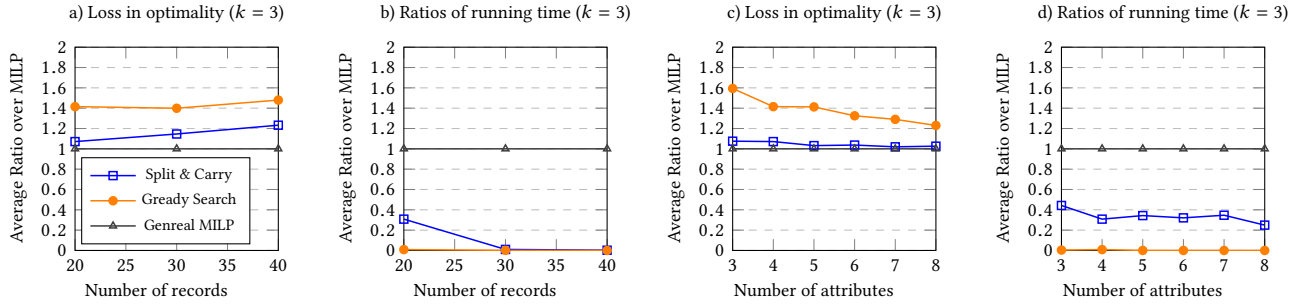


Figure 3: Benchmarking running time and information loss of Greedy Search against other algorithms, as number of records and value of k change.

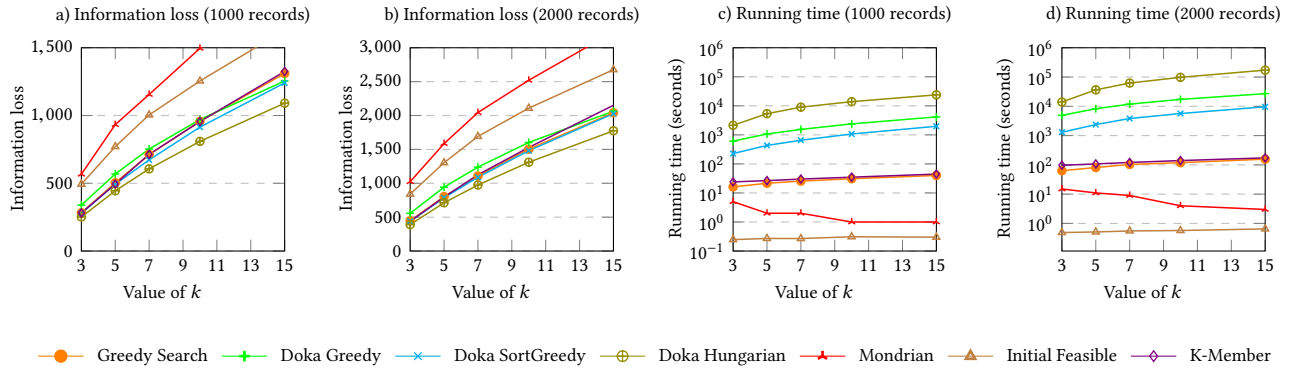


Figure 4: Ratios of objective values of Greedy Search over other algorithms.

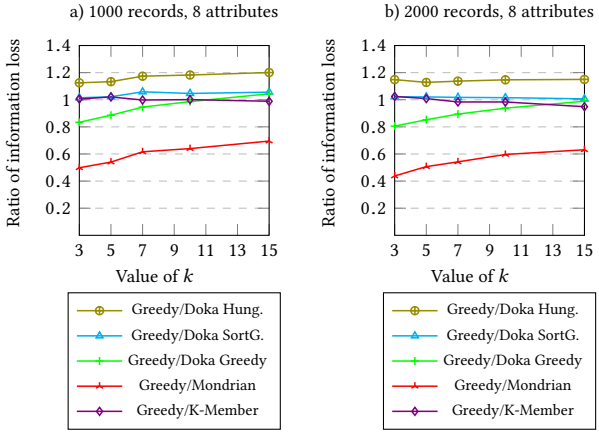
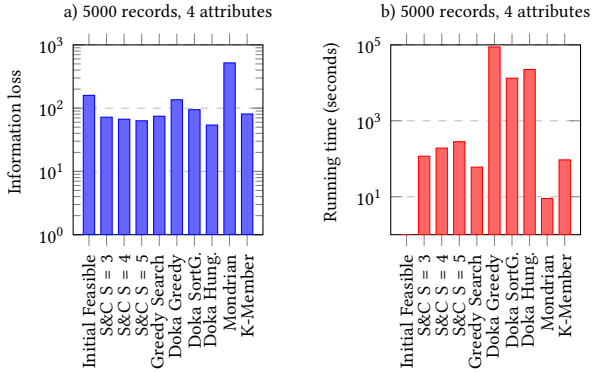


Figure 5: Comparison of objective values and running times of Split & Carry and Greedy Search against other algorithms.



it can also produce outcome providing complete distribution information for categorical data; however the Mondrian does not output a utility measure. We could compute our utility measure directly on the Mondrian, but the algorithm provides open bounds for some records and in fact it does not always provide the tightest possible bounds; e.g., in the anonymized outcome by Mondrian in Fig. 5, for one equivalence class on the Age attribute it produced generalized range (60, 90] while the tightest bounds for this equivalence class were [61, 80]. We make the following notes for the benchmarking results we present on Mondrian:

(1) Since it takes a simple post-processing step to find the tightest bounds for any equivalence class, utility values were computed on the equivalence classes using the tightest bounds.
(2) The running times presented for Mondrian do not include the time it took to compute utility; however the Mondrian has theoretical complexity equal to $O(n \log(n))$ where n is the size of the matrix.

Finally we benchmarked our algorithms against the K-Member Clustering algorithm by [Byun et al., 2007] (hereafter referred to as K-Member). K-Member is a clustering algorithm where the algorithm forms clusters of size k as equivalence classes. There are similarities between K-Member and our Greedy Search

algorithm, in that for the record currently under consideration, both algorithms greedily finds the next record to be in its equivalence class by minimizing the additional information loss incurred. Apart from incorporating weights, the most differentiating step in Greedy Search is the order with which we move to the next record. The basis of Greedy Search is to improve upon the initial feasible solution.

We present our benchmark results against Mondrian, K-Member and Doka algorithms in Fig. 3 and Fig. 4 with various values of k, where we use the GCP to measure utility as the GCP had been used in other literature previously and so does not put our algorithm at an advantage (please see Appendix B for GCP). This also shows that our Greedy Search algorithm can be adapted to other utility measure since its optimality and performance do not depend on the choice of utility measure. In the tests reported in Fig. 3a and Fig. 3c we used 1000 records on all 8 attributes, and in Fig. 3b and Fig. 3d we used 2000 records on the 8 attributes. We can see that Greedy Search can achieve utility that is not too far from those achieved by Doka algorithms but it has a much better running time; while the Mondrian has a very efficient running time and always finishes in a few seconds, the utility it provides is far less favourable. Our Greedy Search performs similarly to K-Member, but as the value of k increases we begin to notice a slight advantage. We can also observe that as we increase the dataset size from 1000 records to 2000 records, the gap in the utility between our Greedy Search and the Doka algorithms narrowed, and in particular the utility of our Greedy Search converged to that of Doka SortGreedy. We also display the ratios of objective values of our Greedy Search over Mondrian, K-Member and the Doka algorithms. We see that for small k, Greedy Search is performing better than Doka Greedy, but as k increases Doka Greedy starts to have advantage over our Greedy Search. The general trend for the tests on 1000 and 2000 records is, when we fix the size of the dataset, as k increases the utility of the Doka algorithms increases. We also see in general that for a fixed k the utility gap between our Greedy search and Doka Hungarian decreases, when we increase the dataset size. That is, as we increase the size of the dataset, the probability of having records that are equivalent or very similar increases. Therefore, even though Doka algorithms produce outcomes that assume a *freeform*, which has a larger solution space than our anonymized outcomes (containing equivalence classes of size at least k), the utility gain from using freeform is small as the anonymized outcomes of these algorithms contain many records that differ in a small number of attributes by small amounts. Fig. 4a and Fig. 4b also show that as we increase from 1000 to 2000 records, the utility of Greedy Search increases relative to all algorithms (as the ratios of objective values over Doka, Mondrian and K-Member decrease).

In Fig. 5 we compare the objective values and running times of our algorithms (Split & Carry with $S = 3, 4, 5$ and Greedy Search) against the Doka, Mondrian and K-Member algorithms, on a dataset of 5000 records with 4 attributes, where the attributes are treated as numerical. We see that for a dataset with a reasonable size and small number of attributes, and for small k ($= 4$), our algorithms have utility close to Doka Hungarian but with better performance, and our algorithms perform better against Doka Greedy and Doka SortGreedy in both utility and performance; whereas the Mondrian has much better performance but also far less utility than our algorithms. However, if the user is only interested in performance, then we recommend the initial feasible solution which provides decent utility and efficient performance and in this test it performs better than Mondrian in both aspects.

4.4 Scalability

In this section we examine the scalability of the algorithms. The tests in this section are based on datasets containing at least 20,000 records. We do not include the Doka algorithms in our scalability evaluation due to two reasons: 1) we observed in Section 4.3 that the Doka algorithms took 3 - 24 hours to anonymize 5000 records (Fig. 5b); 2) we tried to run the Doka algorithms on a dataset with 10,000 records but we received an "OutOfMemoryError". Indeed, the Doka algorithms require storage of previous states for backtracking as well as representation of $n \times n$ edges for complete bipartite graphs, where the latter requires storing 100,000,000 numbers when $n = 10,000$. We consider scalability of the remaining algorithms in terms of the number of records as well as the number of attributes in Fig. 6 - Fig. 8, where the attributes are treated as numerical.

In Fig. 6 we show the running time and information loss values of the algorithms for small values of k ($= 3, 5$) on several CENSUS datasets with 4 attributes. The sizes of the datasets range from 20,000 to 100,000. Note for the case of $k = 5$ in Fig. 6b and 6d we used the Early Termination (ET) feature for Split & Carry, and set "TimeLimit" to be 1000 seconds for each sub-problem, though only very few sub-problems actually triggered ET. We see that for a small number of attributes and small values of k, K-Member has slightly better utility than Greedy Search, but Greedy Search has better running time. We also see that Split & Carry significantly outperforms K-Member and Greedy Search in both running time and utility. The efficiency of Split & Carry is due mainly to two aspects: 1) it solves small sub-problems and efficiency is reaped from state-of-the-art MILP solver; 2) the small sub-problems have small solution spaces since it's likely that there are identical or very similar records in a sub-problem as the dataset contains a large number of records. The results for Mondrian were omitted from these figures for better visual presentation of the other algorithms. Although Mondrian always finishes in a few minutes in all of the tests in this experiment, it has information loss ranging 9 - 56 times that of the initial feasible solution.

In Fig. 7 we look at how the algorithms react to increasing values of k. We acknowledge that Split & Carry is not suited for large values of k and thus it has been excluded from all the experiments with large k. We use a CENSUS dataset with 20,000 records and 8 attributes. We see that while Mondrian has negligible running time compared to other algorithms, its utility is much less. Moreover, the utility gap between Mondrian and the other algorithms increases as k increases. On the other hand, we see that although Greedy Search has slightly less utility than K-Member for small k ($= 5$), but as k increases it starts to gain utility over K-Member and this utility gap further widens for large k ($= 200$).

In Fig. 8 we look at how the algorithms perform when we increase the number of attributes. We use CENSUS datasets containing 20,000 records on 15, 25 and 35 attributes. We observe that Mondrian is still taking minimal time to compute, however the

Figure 6: Running time & information loss for smaller k (3, 5) as the number of records grows up to 100,000.

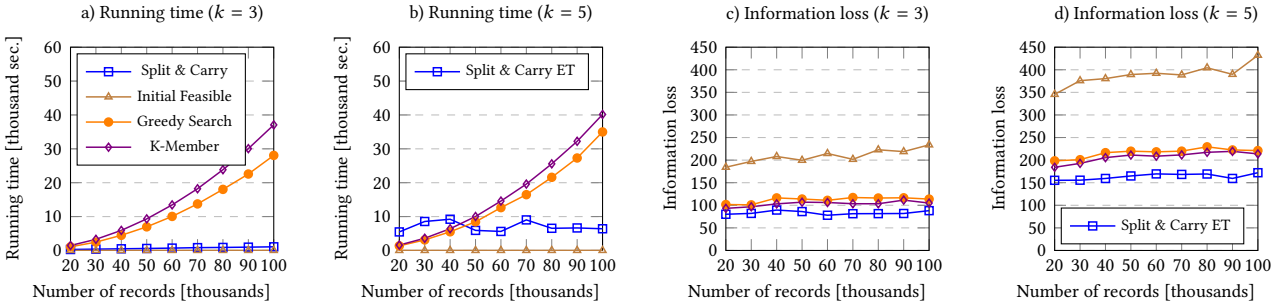


Figure 7: Running time & information loss for 20,000 records and 8 attributes, as the value of k grows up to 200.

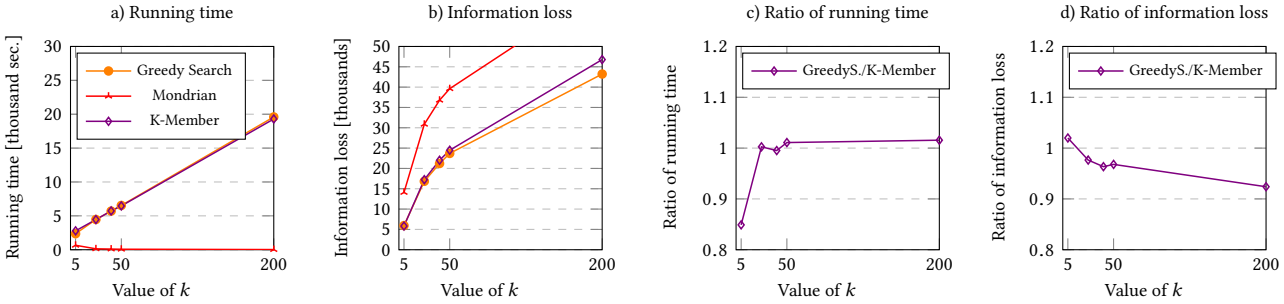
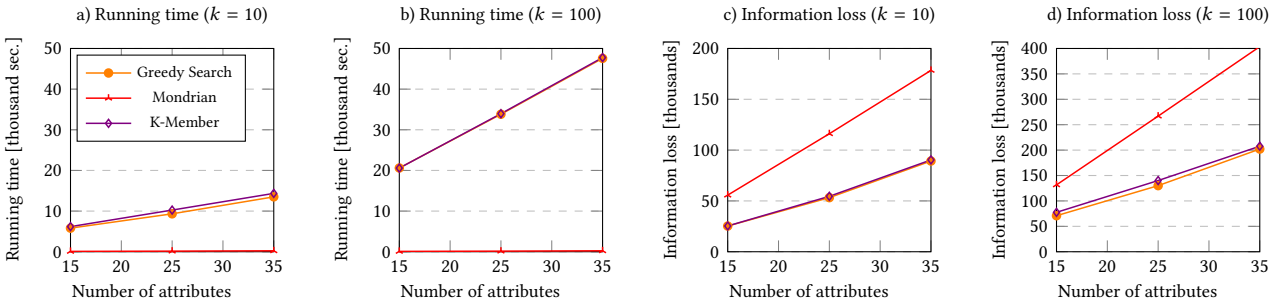


Figure 8: Running time & information loss on 20,000 records, for smaller k (= 10) and large k (= 100), as the number of attributes grows up to 35.



utility gap between Mondrian and other algorithms increases as the number of attributes increases. At small k (= 10) we see that Greedy Search and K-Member have almost the same utility. At k = 100 we observe again that Greedy Search has slightly better utility. Interestingly, we observe consistent results across different numbers of attributes, it appears only the value of k affects how these two algorithms perform compared to each other.

4.5 Discussion on Evaluation Results

In the Optimality experiments in Section 4.2 we see that our practical algorithms provide near-optimal utility for a small k with significant performance improvement over the general MILP. In some cases, Split & Carry actually provides the same optimal objective value as the MILP. We see in the Scalability experiments that for a small number of attributes and small k, Split & Carry delivers excellent performance and utility. Split & Carry is distinguished from the other algorithms in that it always produces optimal solutions for small subsets of the original dataset, and the chaining of the sub-problems enables (in some cases) the possibility of a global optimal solution. In Split & Carry, many equivalence classes might have size greater than k as long as doing so minimizes the information loss. On the other hand, the Greedy Search, K-Member and Doka algorithms are greedy algorithms but their common focus is on finding equivalence classes of size k. Other than indirectly equivalent records, only in cases where the number of records is not divisible by k one would see a small number of equivalence classes with size greater than k in these algorithms. In the case of the Mondrian partitioning algorithm, while it is true that records in the same partition are *close* to each other, the boundary records of two adjacent partitions are also *close*. Since the partitions are determined by median values, utility might not be preserved as such boundary records might be closer to each other than to the records in their own partitions. Thus Split & Carry features two important aspects which are crucial to solving the k-anonymity problem optimally in an efficient way: 1) that the construction of the equivalence classes is not constrained to finding sets of size k,

but also any size greater than k; 2) when we work with smaller subsets for efficiency reasons, there is possibility to move records from one subset to another, i.e. we do not lose the link to the original dataset.

The biggest challenge for our Split & Carry algorithm is still in the size of the sub-problems. For efficiency we are limited to only small sub-problems on small numbers of attributes with small values of $k(\leq 6)$. In practice we often see $k = 3$ or 5 being cited as the minimum requirement [Canada Institute for Health Information, 2014, Cancer Care Ontario, 2018, National Center for Health Statistics, 2019, Statistics Canada, 2014]. Therefore our Split & Carry algorithm can still be a useful tool for k-anonymization.

We see that in the Benchmarking experiments in Section 4.3, the Doka algorithms in general have better utility than our practical algorithms. We discussed in Section 4.3 that this utility gain in Doka algorithms, comes mainly from the fact that their k-anonymized outcomes take a *freeform*, which does not make k equivalent records in the anonymized outcome but rather that for each anonymized record one can identify k records from the original dataset that fall into the ranges of the anonymized record. Moreover, we see in our experiments that for a fixed value of k, as the number of records increases, the utility gap between the Doka algorithms and Greedy Search decreases; in particular Greedy Search performs similarly to Doka Greedy and Doka SortGreedy in Fig. 3b where $n = 2000, m = 8$, and Split & Carry is close to Doka Hungarian in Fig. 5a where $n = 5000, m = 4$. However, our algorithms have much better running time. In general, we expect that for datasets where the value of k and m are large relative to n (e.g. $n < 5000, m = 8, k > 15$) the Doka algorithms will have the best utility due to the *freeform* formulation; but scalability becomes an issue for Doka algorithms for large-sized datasets due to their theoretical complexities and their need to store previous states and $n \times n$ edges. In the cases where k is small relative to n, then the advantage of *freeform* diminishes due to high probability of having many similar records in the dataset.

Throughout the Benchmarking and Scalability experiments, we observe that Mondrian has efficient running time but much larger information loss compared to other algorithms. Thus, it is only scalable in terms of running time; but we note that still the initial feasible solution provides better running time and even better utility than Mondrian. Moreover, the utility gap between Mondrian and Greedy Search increases with the value of k and the number of attributes, as can been seen in Fig. 7b, Fig. 8c, and Fig. 8d.

The initial feasible solution captures an ordering that gives Greedy Search an advantage over K-Member for large values of k. Consider a sorted list of records, such as the one in Table 2b. Notice that the top record appearing in the sorted list is only close to the rest of the list in one direction, i.e., it is only close to records that are immediately below it in this list. Suppose we begin constructing the equivalence class for record i, where i is not the top record (e.g. in this example $i \neq 12$). If we remove records that are close to 12, then 12 becomes farther away from the rest of the list, and more information loss would be incurred to bring 12 to another equivalence class. The larger the value of k, the more records we are likely to remove that are close to 12 before we get to it. Now if we first start with 12, then after removing the equivalent records for 12, we will have a new top record in the remaining list and the process repeats. This is the logic behind the choice of ordering to construct equivalence classes for Greedy Search. We see the effectiveness of this ordering in the Benchmarking and Scalability experiments, where in Fig. 3, Fig. 4, Fig. 7 and Fig. 8, Greedy Search has consistent better utility (though sometimes only slightly) than K-Member for larger values of k (e.g. $k >= 15$).

5 RELATED WORK

The model of k-anonymity for privacy protection from its introduction by Samarati and Sweeny [Samarati and Sweeney, 1998, Sweeney, 2002] has received considerable attention from research community and over years different algorithms have been proposed for applying generalization and suppression techniques in order to achieve k-anonymity. The algorithms differ depending on the domain of application (e.g., data publishing, data mining and statistical disclosure control) [Ayala-Rivera et al., 2014] or the techniques used to provide k-anonymity guarantee while preserving data utility.

Among these algorithms many seek to achieve k-anonymity using search strategies or optimization objectives [Bayardo and Agrawal, 2005, Li et al., 2007, Xiao and Tao, 2006a,b]. Multiple research studies (e.g., [Aggarwal et al., 2005, Meyerson and Williams, 2004, Xu et al., 2006]) have shown that optimally solving a k-anonymization problem is an NP-hard problem. Thus, a number of efficient algorithms have been developed to enable the practicability of anonymization on large datasets. For the purpose of studying our related work, we group the algorithms by the primary techniques or problem representations they use, as partitioning-based, clustering-based and heuristic-based.

The algorithms that employ a partitioning-based technique take a geometric view (exhibiting notions of hyper-cubes) on the problem. They can be seen as taking a *top-down* approach, where the dataset is repeatedly partitioned into smaller subsets according to some criteria, usually defined by thresholds, where eventually partitions of size at least k are formed [LeFevre et al., 2006]. Partitioning-based algorithms usually have the advantage of efficient performance, as equivalence classes are created in a *one-pass* manner without repeated visits to the constructed equivalence classes.

The clustering-based algorithms can be seen as taking a spatial view (but not necessarily geometric due to the non-definitive shapes of the clusters) on the problem, where clusters are formed based on the similarities of the records represented as points in the space. These algorithms usually take a *bottom-up* approach, in that clusters are formed in a successive manner until the entire dataset is exhausted [Aghdam and Sonehara, 2016, Byun et al., 2007, Gionis et al., 2008, Goldberger and Tassa, 2009]. Clustering-based algorithms appear naturally suited for k-anonymization, where information loss can be appropriately encoded as a distance metric, and equivalence classes are clusters of records of size at least k.

We consider as heuristic-based algorithms those that employ a specific problem representation; in particular those algorithms which represent the dataset as a graph, where the records are viewed as vertices, and equivalence is defined via matchings [Doka et al., 2015, Tassa et al., 2012]. The benefit of sound problem representation is evident in the case of the Doka Hungarian algorithm [Doka et al., 2015], where an existing algorithm (Hungarian for finding perfect matchings) can be readily used for solving the problem.

In this work, we aimed to provide a perspective from an optimization stand point. The problem of maximizing utility while satisfying k-anonymity constraint is clearly an optimization problem. A mathematical formulation of k-anonymity as an optimization problem allows us to gauge how much utility is lost in any practical algorithms we devise compared to a theoretical optimum. A similar approach has been taken by Zhang et al. [Zhang et al., 2015] where a formal optimization problem is defined and *k-anonymity by containment* is considered the constraint. They defined containment as a subset S of the feature space that satisfies k-anonymity. The focus of this proposal is on categorical data (specifically binary data) only, with the objective to maximize the size of subset S. In our work, we formulated the general optimal k-anonymization problem as an MILP. While the general model has exponential complexity, we used two properties to devise a practical optimization-based algorithm (Split & Carry): 1) records that are far apart are unlikely to end up in the same equivalence class in an optimal solution; 2) equivalence in k-anonymity is transitive. We remarked that a global optimal solution is sometimes achieved with our algorithm, as demonstrated in Section 4.2 of our experimental evaluation. However, we recognize the cases where there are many attributes or a larger k is desired, Split & Carry will have inefficient performance and thus provide an alternative (Greedy Search) that provides less utility but efficient performance. Our Greedy Search employs a search strategy which shares similar aspects with the K-Member Clustering algorithm proposed by [Byun et al., 2007].

The K-Member algorithm by [Byun et al., 2007] is similar to our Greedy Search in the way we construct an equivalence class for a single record; however the ordering with which we consider records for construction are different, and we've shown our Greedy Search has an advantage for larger values of k. Our objective function is also different from theirs as we have introduced weighting. In the work by Doka et al. [Doka et al., 2015], they offer a MIP formulation for the *freeform* k-anonymity problem. The formulation by Doka et al. is freeform in the sense that a record can be assigned to many different equivalent classes and such equivalence is not transitive. This gives rise to a larger solution space and thus better utility in the optimization problem. In freeform formulation, each anonymized record matches $k-1$ other records of the original dataset, but the number of matches is unknown when we link it to an externally available dataset. Because the data custodian cannot check every external table, Samarati [Samarati, 2001] argued that k-anonymity requirement should be satisfied in the anonymized data themselves.

On the same note as freeform k-anonymity, we also remark that there are related algorithms which seek to provide weaker forms of k-anonymity [Gionis et al., 2008, Tassa et al., 2012] that do not make k identical records, but either that an original record is consistent with k records in the anonymized dataset ((1,k)-anonymity); or that an anonymized record is consistent with k records in the original dataset ((k,1)-anonymity), or both ((k,k)-anonymity). Tassa et al. [Tassa et al., 2012] further introduced *k-concealment* that is a stronger variant of (1,k)-anonymity which they argued to offer the same security guarantee as k-anonymity *computationally*. This gives an interesting research direction for future work, as our Split & Carry can theoretically be adapted to these forms of k-anonymity by relaxing some constraints.

6 CONCLUSIONS

In this paper we introduced a mathematical formulation for k-anonymity as a Mixed Integer Linear Program with a weighted objective function, where the weights can be customized to adjust the likelihoods of generalization on the attributes to suit different research uses. Recognizing that MILPs are hard in general [Papadimitriou, 1981, von zur Gathen and Sieveking, 1978], we also introduced two practical algorithms, both of which are memory efficient and can be applied to datasets containing large numbers of records. We evaluated our algorithms based on their optimality, performance and scalability; we also provided benchmark tests against other classes of algorithms (heuristic-based, partitioning-based and clustering-based). We demonstrated that our Greedy Search algorithm can either achieve similar utility with more efficient running time, or better utility with a less efficient yet still very good running time. We also provided tests to show a remarkable improvement in performance and utility in our Split & Carry over other algorithms when the number of records is large and the number of attributes is small, for small k. From our experiments and theoretical justifications, we conclude that both our algorithms are suitable tools for anonymization on large datasets. We recommend that the Split & Carry algorithm be used when the number of attributes is small and the dataset is very large, for common values of k which are small. When we have many attributes or if we are interested in a larger k value, the recommended choice is Greedy Search whose complexity is always bounded by $O(n^2)$ regardless of the number of attributes and choice of k.

There are several future directions for our research. We have implemented the Split & Carry algorithm with the number of carry-over records equal to k plus the number of distinct equivalent rows to the last k records. As discussed this can lead to performance issues as k becomes larger. As a future research direction, we can instead use a customizable parameter l in place of k to mitigate the problem. It would be interesting to experiment with this parameter to see whether there would be recommendable choices for l.

We can also modify the existing algorithm in the set of records we carry. The current algorithm carries over the k-set of each boundary record if there are either zero or at least k remaining indirectly equivalent records, otherwise we carry over the k-set plus all indirectly equivalent records. We could instead carry over a subset of the k-set, in particular for the m^{th} sub-problem, we can determine the subset as follows: Let x_i be the boundary record under consideration, denote its equivalence class by $Eq_i := \{x_i, x_{i_2}, ..., x_{i_k}\}$. For $l = i_2, .., i_k$, compute the objective value $f_l := f(Eq_i \setminus x_l)$. Then we define the set of carry-over records for x_i to be the $k-(k-r)$ records with the largest f_l's, where r is the size of the set of indirectly equivalent records, $r < k$. In other words, we carry those records that contribute the smallest information loss in Eq_i, which means these records can be potentially grouped into an equivalence class with less information loss than Eq_i in the next sub-problem.

Another way to remedy the situation is, instead of specifying S - the minimum number of the k-sets to include in a sub-problem, we can use a parameter P to specify the exact size of each sub-problem, and a parameter A which describes how many k-sets we accept into our final solution. For example, for $k = 3$ we can have $P = 12$, and $A = 2$, i.e. we accept the two k-sets with minimum utility loss

into our final solution, and carry the remaining records into the next sub-problem of exact size $P = 12$, creating a chain of sub-problems by *bubble carry*. Inevitably in this modified algorithm we will have to solve more sub-problems than our current Split & Carry (leading to a linear increase in complexity), however, in this way we are controlling the size of each sub-problem which we know to have good complexity when the number of attributes is small. This simple modification might have better running time than our current algorithm where in the latter there is some variation in the sizes of the sub-problems.

Both our algorithms use the initial feasible solution as the starting point, and we have seen in the experiments that the initial feasible solution is very efficient and already has utility advantage over the Mondrian algorithm. Therefore, determining an optimal sorting order is also valuable as future work; it can potentially improve utility to an extent that the initial feasible solution by itself, can also be merited as an efficient solution for anonymization.

The k-anonymity model as a privacy preservation technique has a number of limitations that impact our formulation. When an adversary is in possession of external knowledge that can be linked to the anonymized dataset, the adversary might be able to deduce additional information about an individual; some of such adversarial scenarios are described by De Capitani Di Vimercati et al.[De Capitani Di Vimercati et al., 2012]. In these scenarios there are attributes that contribute more to potential re-identification of an individual than others (e.g., a person with a particular birth date who lives in an area of a particular zip code). In our formulation, we can assign smaller weights to these attributes to over generalize them, thus decrease the risk of re-identification. There is, however, one concern with using weights; in a scenario that an adversary has access to multiple datasets originated from the same dataset but anonymized by different weight vectors, the adversary might be able to infer additional information on the original dataset. In particular, in each anonymized set, the attribute with the largest weight has the smallest gap in its generalized bounds. Thus by combining different datasets the adversary can arrive at a much finer dataset. This problem can be mitigated by randomizing the anonymized datasets before their release, however the adversary might still be able to detect additional patterns. A more in-depth study to address the security properties of applying weights presents an interesting future research direction.

ACKNOWLEDGMENT

Supports from Vector Institute for Artificial Intelligence and Natural Sciences & Engineering Research Council of Canada (NSERC) are acknowledged. The authors would also like to thank Dr. K. Doka for kindly answering questions on their algorithm implementations.

REFERENCES

Aggarwal, C. C., 2005. On k-anonymity and the curse of dimensionality. In: Proceedings of the 31st International Conference on Very Large Data Bases. VLDB Endowment, pp. 901–909.

Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A., November 2005. Approximation algorithms for k-anonymity. In: Proceedings of the International Conference on Database Theory (ICDT 2005).
URL http://ilpubs.stanford.edu:8090/645/

Aghdam, M. R. S., Sonehara, N., 2016. Achieving high data utility k-anonymization using similarity-based clustering model. IEICE Transactions on Information and Systems 99 (8), 2069–2078.

Ardagna, C. A., Cremonini, M., di Vimercati, S. D. C., Samarati, P., 2011. An obfuscation-based approach for protecting location privacy. IEEE Transactions on Dependable and Secure Computing 8 (1), 13–27.

Ayala-Rivera, V., McDonagh, P., Cerqueus, T., Murphy, L., et al., 2014. A systematic comparison and evaluation of k-anonymization algorithms for practitioners. Transactions on Data Privacy 7 (3), 337–370.

Bayardo, R. J., Agrawal, R., 2005. Data privacy through optimal k-anonymization. In: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE, pp. 217–228.

Burridge, J., 2003. Information preserving statistical obfuscation. Statistics and Computing 13 (4), 321–327.

Byun, J.-W., Kamra, A., Bertino, E., Li, N., 2007. Efficient k-anonymization using clustering techniques. In: International Conference on Database Systems for Advanced Applications. Springer, pp. 188–200.

Canada Institute for Health Information, 2014. CIHI portal - privacy impact assessment.
URL https://www.cihi.ca/en/portal_addendum_1407_pdf_en.pdf

Cancer Care Ontario, 2018. Data use & disclosure policy.
URL https://www.ccohealth.ca/sites/CCOHealth/files/assets/DataUseAndDisclosurePolicy.pdf

Chawla, S., Dwork, C., McSherry, F., Smith, A., Wee, H., 2005. Toward privacy in public databases. In: Theory of Cryptography Conference. Springer, pp. 363–385.

De Capitani Di Vimercati, S., Foresti, S., Livraga, G., Samarati, P., 2012. Data privacy: definitions and techniques. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 20 (06), 793–817.

Dobkin, D. P., Reiss, S. P., 1980. The complexity of linear programming. Theoretical Computer Science 11 (1), 1–18.

Doka, K., Xue, M., Tsoumakos, D., Karras, P., 2015. k-anonymization by freeform generalization. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. ACM, pp. 519–530.

Dwork, C., 2011. A firm foundation for private data analysis. Communications of the ACM 54 (1), 86–95.

El Emam, K., Dankar, F. K., 2008. Protecting privacy using k-anonymity. Journal of the American Medical Informatics Association 15 (5), 627–637.

Genova, K., Guliashki, V., 2011. Linear integer programming methods and approaches–a survey. Journal of Cybernetics and Information Technologies 11 (1).

Ghinita, G., Karras, P., Kalnis, P., Mamoulis, N., 2009. A framework for efficient data anonymization under privacy and accuracy constraints. ACM Transactions on Database Systems 34 (2), 9.

Gionis, A., Mazza, A., Tassa, T., 2008. k-anonymization revisited. In: 2008 IEEE 24th International Conference on Data Engineering. IEEE, pp. 744–753.

Goldberger, J., Tassa, T., 2009. Efficient anonymizations with enhanced utility. In: 2009 IEEE International Conference on Data Mining Workshops. IEEE, pp. 106–113.

Gurobi Optimization, LLC., 2018. Mixed-integer programming (mip)-a primer on the basics.
URL http://www.gurobi.com/resources/getting-started/mip-basics

Information and Privacy Commissioner of Ontario, Jun. 2016. De-identification guidelines for structured data.
URL https://www.ipc.on.ca/privacy/de-identification-centre/

Iyengar, V. S., 2002. Transforming data to satisfy privacy constraints. In: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 279–288.

Lee, H., Kim, S., Kim, J. W., Chung, Y. D., 2017. Utility-preserving anonymization for health data publishing. BMC medical informatics and decision making 17 (1), 104.

LeFevre, K., DeWitt, D. J., Ramakrishnan, R., 2006. Mondrian multidimensional k-anonymity. In: Data Engineering, 2006. ICDE 2006. Proceedings of the 22nd International Conference on. IEEE, pp. 25–25.

Li, N., Li, T., Venkatasubramanian, S., 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, pp. 106–115.

Liu, K., Kargupta, H., Ryan, J., 2006. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. IEEE Transactions on Knowledge and Data Engineering 18 (1), 92–106.

Meyerson, A., Williams, R., 2004. On the complexity of optimal k-anonymity. In: Proceedings of the 23rd ACM SIGMOD Symposium on Principles of Database Systems. ACM, pp. 223–228.

Mittelmann, H., Jun. 2018. Mixed integer linear programming benchmark.
URL http://plato.asu.edu/ftp/milpc.html

Mosek, 2018. Modelling cookbook 2.3.
URL https://docs.mosek.com/modeling-cookbook/mio.html

National Center for Health Statistics, 2019. Preventing disclosures: Rules for researchers.
URL https://www.cdc.gov/rdc/data/b4/disclosuremanual.pdf

Nemhauser, G., Wolsey, L., 1988. Integer and combinatorial optimization. Wiley-Interscience series in discrete mathematics and optimization. Wiley.
URL https://books.google.ca/books?id=uG4PAQAAMAAJ

Papadimitriou, C. H., 1981. On the complexity of integer programming. Journal of the ACM (JACM) 28 (4), 765–768.

Samarati, P., 2001. Protecting respondents identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13 (6), 1010–1027.

Samarati, P., Sweeney, L., 1998. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Tech. rep., SRI International.

Statistics Canada, 2014. Statistics canada quality guidelines - disclosure control.
URL https://www150.statcan.gc.ca/n1/pub/12-539-x/steps-etapes/4058325-eng.htm

Sutton, A., Samavi, R., Doyle, T. E., Koff, D., 2018. Digitized trust in human-in-the-loop health research. In: Proceedings of the 16th Annual Conference on Privacy, Security and Trust, PST 2018, August 28-30. IEEE, Belfast, Northern Ireland, UK, pp. 1–10.

Sweeney, L., 2002. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (05), 557–570.

Tassa, T., Mazza, A., Gionis, A., 2012. k-concealment: An alternative model of k-type anonymity. Transactions on Data Privacy 5 (1), 189–222.

University of Minnesota, 2018. Ipums-usa.
URL https://www.ipums.org/

U.S. Department of Health & Human Services, 2015. Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act (HIPAA) privacy rule.
URL https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html

US Department of Transportation, 2016. Fatal analysis reporting system.
URL ftp://ftp.nhtsa.dot.gov/FARS/2016

UT Dallas Data Security and Privacy Lab, 2012. UTD anonymization toolbox.
URL http://www.cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php

von zur Gathen, J., Sieveking, M., 1978. A bound on solutions of linear integer equalities and inequalities. Proceedings of the American Mathematical Society 72 (1), 155–158.

Williams, H., 1993. Model Building in Mathematical Programming. Wiley.
URL https://books.google.ca/books?id=hX3CQgAACAAJ

Wong, W. K., Mamoulis, N., Cheung, D. W. L., 2010. Non-homogeneous generalization in privacy preserving data publishing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. ACM, pp. 747–758.

Xiao, X., Tao, Y., 2006a. Anatomy: Simple and effective privacy preservation. In: Proceedings of the 32nd International Conference on Very Large Data Bases. VLDB Endowment, pp. 139–150.

Xiao, X., Tao, Y., 2006b. Personalized privacy preservation. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. ACM, pp. 229–240.

Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A. W.-C., 2006. Utility-based anonymization using local recoding. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785–790.

Zhang, B., Mohammed, N., Dave, V., Hasan, M. A., 2015. Feature selection for classification under anonymity constraint. arXiv preprint arXiv:1512.07158.

APPENDIX

A Weighting Attributes

We present an example of applying weights to the attributes using a dataset from the database of FARS [US Department of Transportation, 2016]. In Table 4, we have the original dataset in (a) on 4 attributes: AGE, SEX, INJ_SEV (injury severity) and DRINKING; we also have the outcome of the dataset anonymized with equal weights in (b), and the outcome anonymized with weight vector (0.8, 0.05, 0.1, 0.05) in (c). We see that the upper-lower bound gaps in the anonymized outcome for the AGE attribute narrowed when we applied a large weight to this attribute, i.e., we achieve better utility for AGE; however this is achieved at the expense of the SEX and DRINKING attributes.

B Categorical Data

When we convert a categorical attribute into a vector of binary attributes, we use 1 to indicate presence of the property. For example, in the dataset used for the results shown in Figs. 3a - 3d, "Marital Status" attribute is represented as $(u_1, u_2, u_3, u_4, u_5, u_6)$ with each binary attribute u_i representing a permissible value in Marital Status. In record 628 the entry for this attribute is 1 (representing status "Married; spouse present"), and thus represented as $(1, 0, 0, 0, 0, 0)$. Its equivalence class by Greedy Search are the records $\{628, 648, 819\}$, and the entries for records 648, 819 under Marital Status are 1 and 6 (representing status "Never married/single"), respectively. The anonymized outcome is represented as "1 | 1 | 6", which means the entries in this class can be translated back into high-level attributes as {"Married; spouse present", "Never married/single"}.

We give a definition of the GCP here that is similar to the definitions in [Doka et al., 2015, Wong et al., 2010].

DEFINITION 8: The Normalized Certainty Penalty (NCP) information loss metric for the anonymized outcome x'_{ij} of an attribute a_j is defined by:

$$NCP(x'_{ij}) = \begin{cases} \frac{count(x'_{ij})-1}{|a_j|} : a_j \texttt{ is categorical} \\ D_{ij} : a_j \texttt{ is numeric} \end{cases}$$

where D_{ij} is as defined in Definition 4.

DEFINITION 9: We also define the Global Certainty Penalty (GCP) information loss metric for anonymized record $x'_i := [x'_{ij}]_{j \in J}$ to be:

$$GCP(x'_i) = \sum_{j \in J} NCP(x'_{ij}) \tag{40}$$

C Worked Examples

In this subsection we work through two examples to demonstrate the steps of the Split & Carry and Greedy Search algorithms. Our input dataset is as in Table 2a. We let $k = 3$ and $S = 3$.

Example 1: Split & Carry

The first step is to sort by variance as described in Section 2.3. The initial feasible solution creates 6 k-sets, namely $\{[12, 1, 11], [14, 7, 10], [16, 19, 13], [6, 9, 5], [8, 2, 5], [17, 0, 8, 18, 3]\}$. The first sub-problem contains the first 3($S = 3$) k-sets, and an initial solution with each k-set as equivalent rows is passed into to the Gurobi solver. The optimal solution for the first sub-problem is give in Table 5b. The k boundary records are $\{16, 19, 13\}$, and they are equivalent to records $\{14, 7, 10\}$, thus the carry-over records from Sub-problem 1 to Sub-problem 2 are $\{14, 7, 10, 16, 19, 13\}$. The equivalence classes of the non-carried records become part of the final solution. The carry-over records are added to the remaining 3 k-sets to form Sub-problem 2 as in Table 6a. We then solve Sub-problem 2 and obtain the optimal solution in Table 6b. Putting the solutions together we obtain the outcome in Table 4b. In this example, the optimal solution to Sub-problem 2 does not change the equivalence classes of the carry-over records, in general this need not be the case. We also note that the final solution provided by Split & Carry in this example is the same as the optimal solution by the general MILP.

Example 2: Greedy Search

The first step is again to sort by variance as described in Section 2.3. Then we move down the sorted list of records. For each record we find $k - 1$ records such that the least information loss is incurred when they are assigned to the k-set of this record. In this example, we first look at the record with index 12, and determine $\{1, 11\}$ to be in its equivalence class. We remove the records $\{12, 1, 11\}$ from the list. The next record in the remaining sorted list is 14, and we determine to k-set for 14 to be $[14, 7, 10]$ and remove these from the list. We continue in this manner until we are left with 2 records $\{4, 18\}$ in the list. For each remaining record, we find in our existing collection of k-sets the set that would incur the least information loss if the said record was added to it, and add the said record to this k-set. The complete set of iterations for this example is given in Table 7.

D On the parameter S

The upper bounds on the sizes of the sub-problems in Fig. 5a - 5b are provided by LEMMA 2, and they are: 24, 27 and 30, for $S = 3$, $S = 4$ and $S = 5$, respectively. We give the actual distributions on the sizes of the sub-problems in Table 8a - 8c and see that the sizes rarely reach their upper bounds.

E Statistics of the Datasets

In this subsection we provide the statistics on the datasets used in our tests. In Table 9 we display the variances of the CENSUS dataset (on 8 attributes) for different numbers of records, the min/max values were given in Table 3. We also display the statistics for the extended datasets of 15-35 attributes in Table 10.

Note that the variances for each attribute in Table 9 are similar across different sizes, thus we expect the information loss incurred to be similar across these datasets when the same k is used. Indeed this is what we observed in Section 4.4 Fig. 6c and Fig. 6d.

F Alternate Database

In this subsection we present some test results using alternative datasets from the FARS [US Department of Transportation, 2016] database containing traffic accidents data. We have datasets on

index	AGE	SEX	INJ_SEV	DRINKING
0	64	2	4	0
1	29	1	0	0
2	42	2	0	0
3	41	2	4	1
4	53	1	2	1
5	59	1	4	0
6	49	1	4	0
7	59	1	2	0
8	80	1	4	0
9	50	1	4	0
10	64	1	3	0
11	55	1	0	0
12	25	1	0	0
13	42	1	4	0
14	33	1	2	0
15	31	2	2	0
16	68	1	3	0
17	20	2	4	0
18	40	1	4	1
19	18	1	4	0

(a) Original dataset

index	AGE	SEX	INJ_SEV	DRINKING
0	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
1	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
2	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
3	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]
4	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]
5	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
6	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
7	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
8	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
9	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
10	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
11	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
12	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
13	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
14	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
15	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
16	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
17	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
18	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]
19	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]

(b) Anonymized - equal weights

index	AGE	SEX	INJ_SEV	DRINKING
0	[64 - 80]	[1 - 2]	[3 - 4]	[0 - 0]
1	[29 - 33]	[1 - 2]	[0 - 2]	[0 - 0]
2	[40 - 42]	[1 - 2]	[0 - 4]	[0 - 1]
3	[40 - 42]	[1 - 2]	[0 - 4]	[0 - 1]
4	[49 - 53]	[1 - 1]	[2 - 4]	[0 - 1]
5	[55 - 59]	[1 - 1]	[0 - 4]	[0 - 0]
6	[49 - 53]	[1 - 1]	[2 - 4]	[0 - 1]
7	[55 - 59]	[1 - 1]	[0 - 4]	[0 - 0]
8	[64 - 80]	[1 - 2]	[3 - 4]	[0 - 0]
9	[49 - 53]	[1 - 1]	[2 - 4]	[0 - 1]
10	[64 - 80]	[1 - 2]	[3 - 4]	[0 - 0]
11	[55 - 59]	[1 - 1]	[0 - 4]	[0 - 0]
12	[18 - 25]	[1 - 2]	[0 - 4]	[0 - 0]
13	[40 - 42]	[1 - 2]	[0 - 4]	[0 - 1]
14	[29 - 33]	[1 - 2]	[0 - 2]	[0 - 0]
15	[29 - 33]	[1 - 2]	[0 - 2]	[0 - 0]
16	[64 - 80]	[1 - 2]	[3 - 4]	[0 - 0]
17	[18 - 25]	[1 - 2]	[0 - 4]	[0 - 0]
18	[40 - 42]	[1 - 2]	[0 - 4]	[0 - 1]
19	[18 - 25]	[1 - 2]	[0 - 4]	[0 - 0]

(c) Anonymized - unequal weights (.8, .05, .1, .05)

Table 4: Sample dataset from FARS[US Department of Transportation, 2016] anonymized by equal and unequal weights

index	AGE	SEX	INJ_SEV	DRINKING
12	25	1	0	0
1	29	1	0	0
11	55	1	0	0
14	33	1	2	0
7	59	1	2	0
10	64	1	3	0
16	68	1	3	0
19	18	1	4	0
13	42	1	4	0

(a) Sub-problem 1 Dataset

index	AGE	SEX	INJ_SEV	DRINKING
12	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
1	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
11	[25 - 55]	[1 - 1]	[0 - 0]	[0 - 0]
14	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
7	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
10	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
16	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
19	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
13	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]

(b) Solution to Table 5a

Table 5: Sub-problem 1 of Worked Example 1

index	AGE	SEX	INJ_SEV	DRINKING
14	33	1	2	0
7	59	1	2	0
10	64	1	3	0
16	68	1	3	0
19	18	1	4	0
13	42	1	4	0
6	49	1	4	0
9	50	1	4	0
5	59	1	4	0
8	80	1	4	0
2	42	2	0	0
15	31	2	2	0
17	20	2	4	0
0	64	2	4	0
4	53	1	2	1
18	40	1	4	1
3	41	2	4	1

(a) Sub-problem 2 Dataset

index	AGE	SEX	INJ_SEV	DRINKING
14	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
7	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
10	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
16	[59 - 68]	[1 - 1]	[2 - 3]	[0 - 0]
19	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
13	[18 - 42]	[1 - 1]	[2 - 4]	[0 - 0]
6	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
9	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
5	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
8	[49 - 80]	[1 - 1]	[4 - 4]	[0 - 0]
2	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
15	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
17	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
0	[20 - 64]	[2 - 2]	[0 - 4]	[0 - 0]
4	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]
18	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]
3	[40 - 53]	[1 - 2]	[2 - 4]	[1 - 1]

(b) Solution to Table 6a

Table 6: Sub-problem 2 of Worked Example 1

Index	Remaining Records	k-Sets
12	{14,7,10,16,19,13,6,9,5,8,2,15,17,0,4,18,3}	[12,1,11]
14	{16,19,13,6,9,5,8,2,15,17,0,4,18,3}	[12,1,11],[14,7,10]
16	{19,13,6,8,2,15,17,0,4,18,3}	[12,1,11],[14,7,10],[16,5,9]
19	{8,2,15,17,0,4,18,3}	[12,1,11],[14,7,10],[16,5,9],[19,13,6]
8	{2,15,4,18,3}	[12,1,11],[14,7,10],[16,5,9],[19,13,6],[8,0,17]
2	{4,18}	[12,1,11],[14,7,10],[16,5,9],[19,13,6],[8,0,17],[2,15,3]
4	{18}	[12,1,11],[14,7,10,4],[16,5,9],[19,13,6],[8,0,17],[2,15,3]
18	{}	[12,1,11],[14,7,10,4],[16,5,9],[19,13,6,18],[8,0,17],[2,15,3]

Table 7: Steps of Greedy Search for Example 2.

Size	Count
9	1
11	1
12	432
13	33
14	30
15	21
16	22
17	16

(a) $S = 3, k = 3$

Size	Count
12	1
14	1
15	292
16	27
17	29
18	16
19	16
20	11
21	8
22	5
23	5
24	3
26	1
27	2

(b) $S = 4, k = 3$

Size	Count
15	1
18	200
25	15
22	16
23	12
26	19
20	17
28	9
19	23
21	12
27	3
24	4
29	1
30	1
13	1

(c) $S = 5, k = 3$

Table 8: Distributions of sub-problem sizes for Fig. 5a - 5b.

4 attributes: MONTH (of occurrence), AGE, SEX, INJ_SEV. All 4 attributes can be viewed as numeric (with SEX being binary).

In Fig. 9a - 9b, we see that our algorithms still achieve similar utility to the Doka Hungarian algorithm while having better performance. Our Greedy Search appears to have better performance than our Split & Carry in Fig. 9b when we have only 1000 records;

Number of Records	Gender	Age	Marital Status	Race	Birthplace	Education Level	Work Class	Occupation
20,000	0.25	224.47	4.83	0.57	3340.06	7.33	11.27	78.00
30,000	0.25	225.03	4.82	0.56	3402.68	7.30	11.22	78.53
40,000	0.25	225.31	4.82	0.57	3414.58	7.33	11.20	78.72
50,000	0.25	225.64	4.81	0.58	3417.06	7.38	11.31	78.70
60,000	0.25	226.29	4.81	0.57	3372.39	7.36	11.34	78.66
70,000	0.25	226.37	4.83	0.56	3410.34	7.36	11.28	78.51
80,000	0.25	226.32	4.83	0.56	3412.74	7.38	11.28	78.53
90,000	0.25	226.96	4.83	0.56	3424.04	7.38	11.28	78.69
100,000	0.25	227.14	4.83	0.57	3412.27	7.39	11.31	78.50

Table 9: Variances (rounded to 2 decimals) by attribute for the CENSUS datasets with 8 attributes.

Attribute	Variance	[min, max]
Family Unit	0.004533004	[1,3]
Family Size	1.442669131	[2,16]
Age of Oldest Child	58.72416437	[0,90]
Age of Youngest Child	56.41184578	[0,90]
Relationship to Household Head	0.824784829	[1,12]
Gender	0.241520454	[1,2]
Age	100.447806	[16,90]
Birth Quarter	1.252327406	[1,4]
Marital Status	1.167666581	[1,6]
Birth Year	100.3407262	[1889,1963]
Times Married	0.148862341	[0,2]
Age at First Marriage	20.55224999	[0,90]
Children Ever Born	4.679614458	[0,13]
Race	0.393475514	[1,7]
Birthplace	4471.679274	[1,465]
Years in the United States	0.63840967	[0,5]
Language Spoken	30.45895319	[1,96]
Highest Grade of Schooling	8.217952835	[1,23]
Class of Worker	0.068740375	[1,2]
Class of Worker - Detailed	11.5164085	[13,29]
Weeks Worked Last Year	168.0059844	[0,52]
Hours Worked Last Week	140.8818287	[1,99]
Usual Hours of Work	156.7482934	[0,99]
Weeks Unemployed Last Year	26.31284308	[0,52]
Income From Wages	98538554.92	[0,75000]
Other Income	2669021.007	[0,75000]
Occupation	81.01775489	[4,79]
Migration Status, 5 Years	0.459946975	[1,4]
Place of Work, State	94.98776039	[1,80]
Means of Transportation to Work	39.7001238	[11,70]
Vehicle Occupancy	0.708769336	[0,7]
Time in Transit	239.9953666	[0,99]
Citizenship	0.13065803	[0,3]
Year of Immigration	0.68959079	[0,6]
Work Disability Status	0.033916173	[1,2]

Table 10: Variances, min/max values of the CENSUS datasets with 15-35 attributes.

however, we see in Fig. 10b that when we increase the number of records to 40000, Split & Carry has both better performance and utility. These results are consistent with our earlier observations.

G Hardware Information

In this subsection we list the typical hardware specifications for running the experiments. We used Google Cloud Computing Services (https://cloud.google.com/) with machine type n1-highcpu-8. In Table 11 we list the specifications for one of the processors, the remaining seven are identical except for the processor id, core id, apicid and initial apicid.

Figure 9: Comparison of objective values and running times for FARS database with 1000 records, 4 attributes and $k = 3$

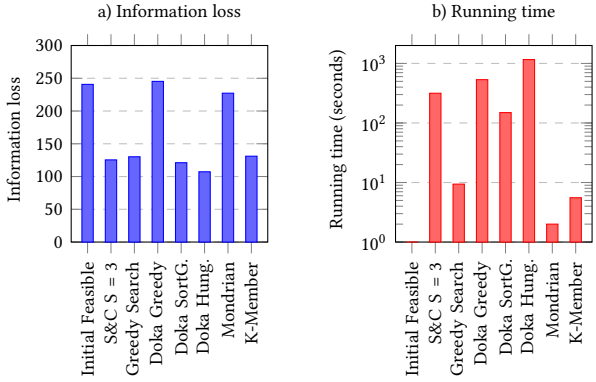
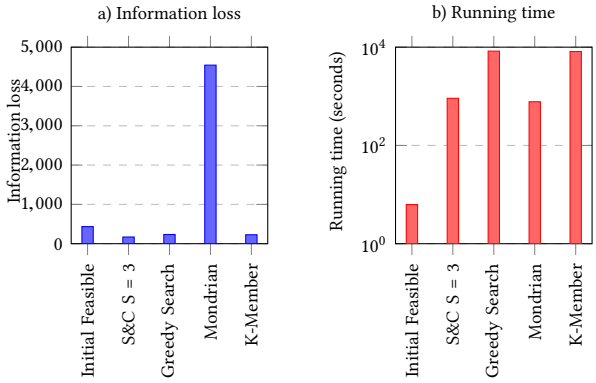


Figure 10: Comparison of objective values and running times for FARS database with 40,000 records, 4 attributes and $k = 3$



MemTotal	7352436 kB
MemFree	6587692 kB
MemAvailable	6966836 kB
Buffers	50336 kB
Cached	542508 kB
SwapCached	0 kB
Active	409856 kB
Inactive	233532 kB
Active(anon)	50700 kB
Inactive(anon)	10120 kB
Active(file)	359156 kB
Inactive(file)	223412 kB
Unevictable	0 kB
Mlocked	0 kB
SwapTotal	0 kB
SwapFree	0 kB
Dirty	0 kB
Writeback	0 kB
AnonPages	50540 kB
Mapped	23064 kB
Shmem	10280 kB
Slab	43912 kB
SReclaimable	29668 kB
SUnreclaim	14244 kB
KernelStack	2160 kB
PageTables	2780 kB
NFS_Unstable	0 kB
Bounce	0 kB
WritebackTmp	0 kB
CommitLimit	3676216 kB
Committed_AS	95452 kB
VmallocTotal	34359738367 kB
VmallocUsed	0 kB
VmallocChunk	0 kB
HardwareCorrupted	0 kB
AnonHugePages	0 kB
ShmemHugePages	0 kB
ShmemPmdMapped	0 kB
HugePages_Total	0
HugePages_Free	0
HugePages_Rsvd	0
HugePages_Surp	0
Hugepagesize	2048 kB
DirectMap4k	99316 kB
DirectMap2M	3256320 kB
DirectMap1G	5242880 kB

Table 11: Memory Specifications

processor	0
vendor_id	GenuineIntel
cpu family	6
model	79
model name	Intel(R) Xeon(R) CPU @ 2.20GHz
stepping	0
microcode	0x1
cpu MHz	2200
cache size	56320 KB
physical id	0
siblings	8
core id	0
cpu cores	4
apicid	0
initial apicid	0
fpu	yes
fpu_exception	yes
cpuid level	13
wp	yes
bugs	cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs
bogomips	4400
clflush size	64
cache_alignment	64
address sizes	46 bits physical, 48 bits virtual
power management:	

Table 12: CPU Specifications