Genetic Algorithms in Matrix Representation and Its Application in Synthetic Data

Yingrui Chen[*], Mark Elliot[**] and Joe Sakshaug[***]

[*] University of Manchester, yingrui.chen@manchester.ac.uk
[**] University of Manchester, mark.elliot@manchester.ac.uk
[***] University of Manchester, joe.sakshaug@manchester.ac.uk

Abstract: This paper is the implementation of an earlier position paper (Chen, Elliot & Sakshaug, 2016) and explains how to use a new form of genetic algorithms (matrix GAs) to generate synthetic data and provides a proof of concept using a small individual-level microdata set. The new method is able to iteratively optimise synthetic data based on a set of utility parameters until its difference from the original data achieves a desired level. The paper describes the advantages of this method and its potential in synthetic data production. It covers theoretical and computerised model design and specifies further development of this study.

1 Introduction

Algorithms are the core of machine learning (ML), which is designed for solving problems with non-trivial solution spaces and where prior knowledge cannot be completely specified. ML instantiates a series of instructions that transform input into output; modern machine learning algorithms are increasingly designed for unstructured global optimisation problems (Ethem, 2014). Notwithstanding this, algorithmic methodology presupposes that most problems can be reduced to optimisation problems with objectifiable constraints and objectives.

The properties of machine learning algorithms show significant potential for synthetic data production. Firstly, ML does not require an accurate pre-assumed model to cover all statistical properties that the synthetic data is supposed to have. Machine learning algorithms are interruptible so users can set any statistic as the objective any time during model processing. Secondly, it uses less human effort to find optimal solutions, which is more efficient with big datasets.

The idea of using machine learning algorithms in data protection has been mentioned by several researchers: Drechsler (2010) proposes modern optimisation tools as a generator of synthetic data. Navarro-Arribas and Torra (2015) describe data protection as an optimising problem with two opposite constraints: information loss and disclosure risks and proposed using Genetic Algorithms (GAs) to improve the protection of databases or to improve parameters in disclosure control models. More recently the current authors established a framework for a GA approach to producing synthetic data (Chen et al 2016).

1.1 Genetic Algorithms

Genetic algorithms (GAs) are one of the well-known machine learning algorithms. Conceptually, they mimic the process of natural selection. GAs use a parallel search to randomly select *individuals* from a *population* of candidates, apply crossover (exchange information between candidates) and mutate the candidates (perturb information) until the whole system reaches convergence (where all candidates are identical) or the system meets some user defined criterion (Goldberg, 1989). Every run of selection, crossover and mutation processes is called a generation. The number of generations can be used to measure the speed of convergence of a GA model. In GA, the set of candidates in the current generation is generally called "the

population", but in order to distinguish to the 'population' we generally use in statistics, the population of candidate datasets will be called "the GA population".

The most common representation of individuals in GA populations is in the form of binary strings (effectively one-dimensional arrays). There is a relative lack of work on what are called matrix-GAs (two-dimensional arrays). For the synthetic data production use case, there are strong reasons to use a matrix representation. Firstly, the matrix is the standard way to present microdata, which contains information directly collected from individual population units (Ciriani et al, 2007). Secondly, as Marchette and Solka (1996) observe, matrix GAs can most appropriately display data structures of their candidates so they can not only optimise records but also their relation to surrounding locations in a dataset.

2 The properties of GAs

2.1 Fitness Functions

As an optimisation problem, the objective of synthetic data production is to minimise its difference in statistical inference of the synthetic datasets compared to the original one (Abowd & Lane, 2004). In GAs, the measurement of how close an individual is to the objective is called fitness. Hence, the fitness of a candidate dataset within the GA population is measured by its analytical similarity to the original dataset – a property that is often called utility in the statistical disclosure control literature (see for example Duncan et al 2011). Practically, the similarity is evaluated by the divergence of a certain statistical properties between the two datasets: the smaller the divergence is the more similar they are and therefore the higher fitness of the synthetic dataset.

2.2 Selection Schemes

Candidates from the GA population in the current generation are selected into a pool before crossover. The purpose of selection is to eliminate worse candidates and give higher chance to better candidates to survive and pass their good properties to the next generation. Candidates are selected from the current GA population with replacement so that one each candidate can crossover with others more than once (Melanie, 1998). Standard selection schemes can be classified into two categories based on their mechanisms: Fitness-proportional selection schemes and Ranking-based selection schemes. Fitness-proportional schemes select candidates according to their fitness values. The fitter a candidate is the more likely it is to survive. Ranking based schemes select candidates based on the ranks of their fitness values; the candidate with highest rank is the most likely to survive. The probability from fitness-proportional selection depends on the value of fitness itself. In ranking-based selection, the candidate of the given rank always has fixed probability to be selected no matter how much it is better than others. One that is much better than the other candidates may only have slightly higher chance to be selected compared to others in the same generation. Fitness-proportional selection schemes avoid this issue and cause less elimination of fitter candidates.

Fitness-proportional selection	Ranking-based selection schemes
Roulette Wheel Selection Stochastic Universal Selection	Tournament selection Linear ranking selection Exponential ranking selection Truncation selection*

Table 1.1 Selection schemes

This paper only compares stochastic universal selection (SUS) and linear ranking selection that can be considered as representatives of the two categories. For more details about other selection schemes please check Blickle and Thiele's (1995) work. SUS selects candidates according to the probability that is proportional to one's fitness value. In linear ranking selection, Candidates in the GA population are ranked in ascending order of their fitness values from 1 to N, where N refers to the best and 1 refers to the worst. All candidates should be assigned into different ranks even though they may have same fitness values. The probability of a candidate to be selected is linearly proportional to its rank but not fitness (Chudasama et al, 2012). For each selection scheme we ran 10 simulations over population sizes 10, 50, and 100 (using a toy datasets described in section 3.0) and took the average number of generations when the process reaches convergence. It is expected to take longer when the GA population size increases. The process terminates after all candidates converge or there appears any candidate whose divergence value is less than 0.02 (calculation of divergence is described in 3.1). The average divergence values after convergence were also compared and the less the value is the higher the optimality the result has.

	Stochastic Universal Selection		Linear Ranking selection	
Population size	Average Number of generations before convergence	Average divergence value after convergence	Average Number of generations before convergence	Average divergence value after convergence
10	31	0.042	10	0.025
50	119	0.017	50	0.012
100	219	0.015	80	0.008

Table 1.2 *Comparison between SUS and linear ranking selection*

The table shows that SUS took longer to reach convergence compared to linear ranking selection and the optimality of its results is neither as satisfactory as the latter. It does not indicate that SUS is a weak selection scheme since it explored more possible solutions during the process.

2.3 Crossover

Crossover is a special operator of GAs that differs them from other algorithms. The crossover operator in this paper is to exchange a selected random block between two attributes in two candidates. Suppose there are N non-identical synthetic datasets in the population of GA that are denoted as $X^n, n \in \{1, \dots, N\}$. Each synthetic dataset consists of K attributes and M cases, $x_{ij}, i \in \{1, \dots, K\}, j \in [1, \dots, M]$. Crossover that occurred between a random pair of synthetic datasets, for example, X^1 and X^2 looks like:

$$\begin{pmatrix} \boxed{x_{11}^1} & x_{12}^1 & \dots & x_{1k}^1 \\ \boxed{x_{21}^1} & \boxed{x_{22}^1} & \dots & x_{2k}^1 \\ x_{31}^1 & x_{32}^1 & \dots & \boxed{x_{3k}^1} \\ x_{41}^1 & \boxed{x_{42}^1} & \dots & \boxed{x_{4k}^1} \\ x_{51}^1 & x_{52}^1 & \dots & \boxed{x_{5k}^1} \\ x_{61}^1 & x_{62}^1 & \dots & x_{6k}^1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1}^1 & x_{m2}^1 & \dots & x_{mk}^1 \end{pmatrix} \quad \begin{pmatrix} \boxed{x_{11}^2} & x_{12}^2 & \dots & x_{1k}^2 \\ \boxed{x_{21}^2} & \boxed{x_{22}^2} & \dots & x_{2k}^2 \\ x_{31}^2 & x_{32}^2 & \dots & \boxed{x_{3k}^2} \\ x_{41}^2 & \boxed{x_{42}^2} & \dots & \boxed{x_{4k}^2} \\ x_{51}^2 & x_{52}^2 & \dots & \boxed{x_{5k}^2} \\ x_{61}^2 & x_{62}^2 & \dots & x_{6k}^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1}^2 & x_{m2}^2 & \dots & x_{mk}^2 \end{pmatrix}$$

Fig 1.1 Selected random block before crossover

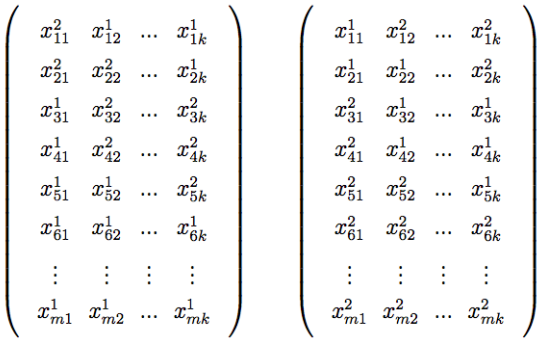

Fig 1.2 Selected random block after crossover

2.4 Mutation

Crossover was once considered as the main mechanism of variation to solution spaces in GAs, but Melanie (1998) argued that researchers should pay the same attention to mutation. Like crossover, there are at least two ways to perform mutation in matrix and this paper only studies mutation on single attributes. The process of mutation is: firstly, the model decides whether mutation happens on a single synthetic attribute based on a given probability. If the decision is "Yes", then the model replaces randomly selected positions by new values that are chosen from a user-determined distribution, which could be the univariate-distribution of the original attribute, uniform distribution or something else. The probability of mutation is usually fixed during the process.

3 Model and Simulations

3.0 Data

The toy dataset used here as a proof of concept a small dataset with 32 records and 4 attributes. There are three reasons to start small: 1) matrix GAs massively increase the workload compared to string GAs, so using a small dataset accelerates the process of programme development and testing; 2) the complexity of matrix GAs results in a higher likelihood of errors and using a small dataset allows to check results line by line; 3) matrix GAs are new in GAs so there are no previous cases to be referred to.

The process of optimisation starts from a set of datasets generated from the univariate distributions of original attributes. The initial GA population size can be any amount greater than 4. The larger the population is the slower the process is but the more variation that exists and the higher chance of discovering the global optima.

3.1 GA model Specification

The approach is designed to generate synthetic datasets for non-hierarchical categorical microdata. Users input a dataset and select the statistics that they wish to preserve. The system will firstly calculate those statistics for the dataset. Then it will generate the required number of synthetic versions using the univariate distributions. The number of synthetic versions is user-determined and will be set up as the size of the initial GA population. These synthetic datasets will then enter the GA process and become iteratively optimised until one or more of them reaches the desired level of closeness to the original data according to the selected statistics.

For the test case used for in this proof of concept, Cramer's V was employed as the single objective to measure the closeness between the synthetic and original data. Shlomo (2009) indicates that Chi-square and Cramer's V (c) are two key statistics to

measure relevance between pairs of categorical attributes. Cramer's V is derived from the Chi-square statistic and can evaluate the level of association between two categorical attributes. So the objective is to minimise the mean of differences of Cramer's V (ϕ_c) between each of the attributes. Suppose the 4 original attributes are X_1, X_2, X_3 and X_4 and its corresponding synthetic versions are X'_1, X'_2, X'_3 and X'_4, the objective can be expressed as a minimisation of the function F:

$$F = \binom{4}{2}^{-1} \sum_{i=1}^{3} \sum_{j=i+1}^{4} |\phi_c(x'_i, x'_j) - \phi_c(x_i, x_j)| \quad (1)$$

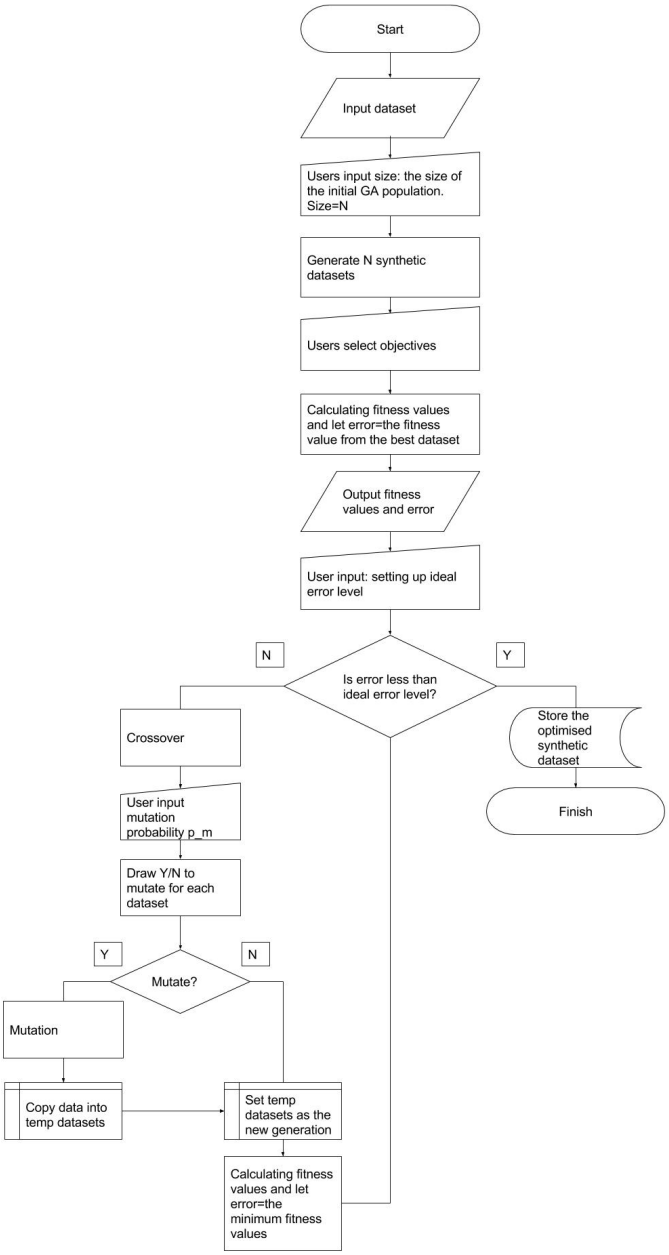

Fig 2.1 Flowchart of the GA model

3.2 Simulations from a general GA approach

The success of an algorithm depends on the details of its operators (Melanie, 1998). Although it might be better in some cases to use a GA model that is specifically designed based on prior knowledge of the particular problem, but general GA models can also work efficiently. The following graphs show how the divergence values (calculated using equation (1)) change over generations using a GA model with linear ranking selection. They show three situations: a GA model with crossover only, a GA model with mutation only and a GA model with both.

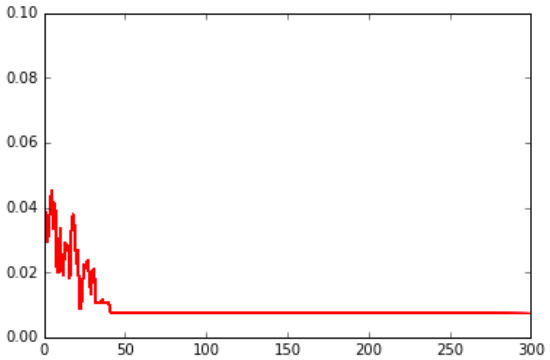

Fig 2.2 Mean fitness values by generation for Simulation with crossover only and toy dataset

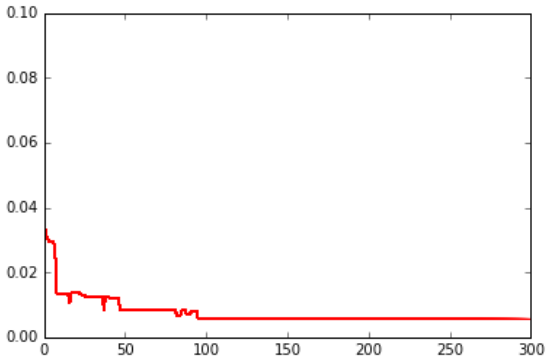

Fig 2.3 Mean divergence values by generation for simulation with mutation (mutation rate=0.1) only

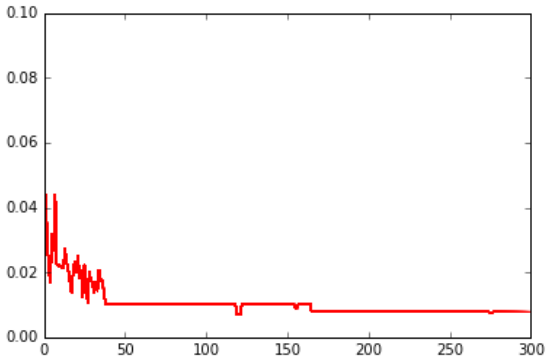

Fig 2.4 Mean divergence values by generation for simulation with crossover and mutation rate=0.1

Comparison between these graphs clearly justifies how crossover and mutation contributes variation to the searching space of optimal solutions and why a parallel search engine has higher chance to find a near-optimum solution. The simulation with crossover only (Fig 2.2) explored more solutions compared to the one with mutation only (Fig 2.3) but it converged quickly and stopped searching new solutions once the

convergence reached. The combination (Fig 2.4) apparently found more solutions before convergence and still searched for more after the convergence reached. Graphs 2.5 to 2.7 compare the three situations above over a dataset with the same four attributes as the toy dataset but with 1000 cases. They may not differ as much as the smaller dataset but still show that the combination of crossover and mutation explore more possible solutions and the convergence speed (in terms of number of generations) is still fast. However, the convergent solution is further away from the global optima.

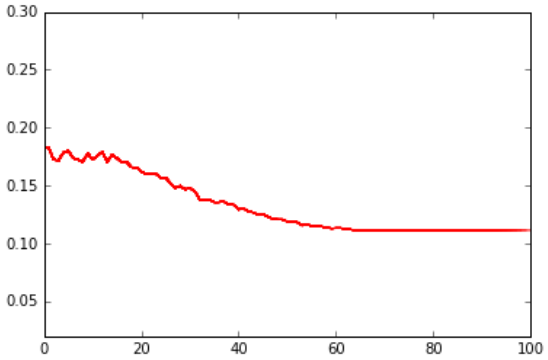

Fig 2.5 Mean divergence values by generation for simulation with crossover only for a larger dataset

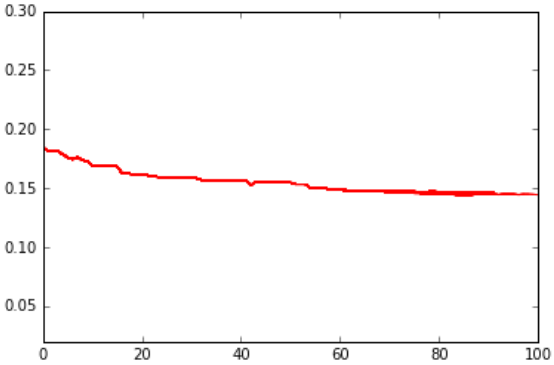

Fig 2.6 Mean divergence values by generation for simulation with mutation (mutation rate=0.1) only for a larger dataset

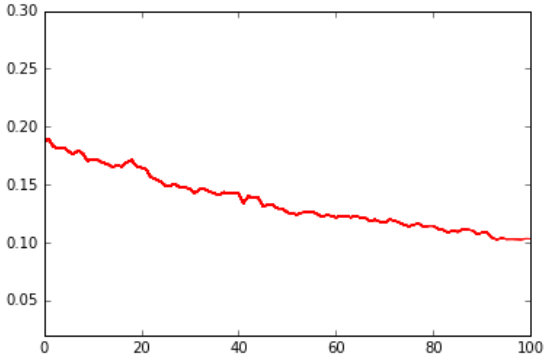

Fig 2.7 Mean divergence values by generation for simulation with crossover and mutation rate=0.1 for a larger dataset

4. Discussion

4.1 Selection

It is difficult to decide how "elitist" the selection schemes should be. A gentle selection operator like SUS assigns probabilities to all candidates to be selected but prolongs the time required for the whole GA population to mature. A restricted operator kills less fit candidates in one shot but results in faster convergence to the local optima. Identifying the appropriate selection operator for synthetic data generation is a topic for future work.

4.2 Adaptive Crossover

At the beginning the trial was implemented on a small GA population; thus, all selected candidates participated crossover to explore the solution space. For a larger GA population like 1000 candidates), an adaptive crossover with adaptive probability p_c is preferred to avoid uncontrollable growth from unordered crossover between candidates. Srinivas and Patnaik (1994) proposed an adaptive crossover rate that changes over generations. Suppose f_c is the divergence value of the best candidate in the generation. The existence of k_c constrains p_c in the range of $[0.0, 1.0]$.

$$p_c = \frac{k_c(f_c - f_{min})}{\bar{f} - f_{min}}, k_c < 1.0$$

There exist some drawbacks of using adaptive crossover. Firstly, there is no clear rule to tell when to change k_c, or whether the population size is large enough to perform adaptive crossover. Furthermore, the formula prevents the best candidate from crossing over with others, which limits evolution if the process is not sufficiently developed or the GA population size is small. In diagram 3.1 we compare the impact from different k_c to p_c over a group of 20 candidates:

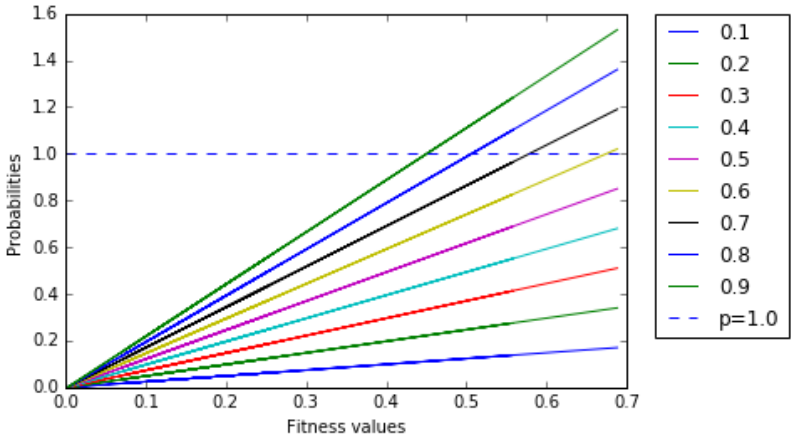

Fig 3.1 Comparison of the effect of difference k_c to p_c

It should be clarified that p_c is not a probability that sums to unity over all candidates but rather is specific to each candidate. The value of p_c returned by the formula can be greater than 1.0, and then needs to be adjusted to 1.0 heuristically. Future work will determine whether to use adaptive crossover by comparing fixed and adaptive crossover over the same GA population.

4.3 Adaptive Mutation

There are two common approaches to design mutation with adaptive probability p_m. One assumes that p_m is adaptive in favour of the difference between individual fitness f_i and the average fitness $\overline{f}$ of the current generation. If $f_i > \overline{f}$ (better than the average), then p_m should be low to keep properties of the good individual. If $f_i < \overline{f}$

(worse than the average), then p_m should be reasonably high to explore a more diverse range of new solutions (Libellin & Alba, 2000). An alternative is that p_m empirically decreases over the number of generations but is independent to individual fitness values. However, this reduces users' control on values of p_m (Thierens, 2002).

Previous research has shown that adaptive mutation is based on the fact that varying probabilities perform better than fixed probabilities in GAs. (see for example Thierens, 2002 for discussion) However, there are some downsides. First of all, it increases computational workload unnecessarily if fixed mutation probability in GAs can also give satisfactory outputs. Secondly, the involvement of adaptive mutation, especially when p_m depends on the individual fitness level, makes the whole process less predictable and impacts the consistency of the whole model (Thierens, 2002). It will require further research to decide whether - and which kind of - adaptive mutation should be used in the GA model for producing synthetic data.

5 Conclusion

We present here GAs as an alternative method for the production of synthetic data. GAs have the feature of being able to automatically optimise synthetic datasets based on given statistical properties and users can update these properties when the model is processing. The process of GAs is to select better candidates (synthetic datasets) and to produce new candidates through crossover and mutation. Since these candidates have higher chances to be selected and bear offspring, the process will eventually converge to an optimal solution. In further study we will make adjustments to different operators in the model and critically we will introduce disclosure control metrics to guarantee the generated dataset provides the statistical properties of the original dataset without duplicating it

References

Blickle, T. & Thiele, L. (1995). *A Comparison of Selection Schemes used in Genetic Algorithms.* Computer Engineering and Communication Networks Lab. Swiss Federal Institute of Technology.

Chen, Y., Elliot, M., Sakshaug, J. (2016). *A Genetic Algorithm Approach to Synthetic Data Production*. Proceedings of the 1st International Workshop on AI for Privacy and Security. Article No. 13.

Chudasama, C.; Shah, S.M. and Panchal, M. (2011) *Comparison of Parents Selection Methods of Genetic Algorithm for TSP*. International Conference on Computer Communication and Networks CSI- COMNET-2011.

Ciriani, V., di Vimercati, S.D.C., Foresti, S., Samarati, P., (2007). *Microdata Protection*. In: Yu T., Jajodia S. (eds.) Secure Data Management in Decentralized Systems, pp. 291–321, Springer, New York.

Cortez, P. (2014). *Modern Optimization with R*; Springer. p.v

Drechsler, J. (2010). *Using support vector machines for generating synthetic datasets; in* Privacy in Statistical Databases; Springer Berlin Heidelberg; pp. 148-161.

Ethem, A. (2014). *Introduction to Machine Learning.* Third Edition, MIT Press.

Liberllin, S., and Alba, P., (2000). *Adaptive mutation in genetic algorithms*, Soft computing. vol.4 pp.76-80.

Melanie, M. (1998). *An Introduction to Genetic Algorithms.* MIT press

Navarro-Arribas, G. and Torra, V. (2015). *Advanced Research in Data Privacy;* Studies in Computational Intelligence. Vol. 567. Springer Switzerland. pp.3-37.

Shlomo, N. (2009), *Releasing Microdata: Disclosure Risk Estimation.* D*ata Masking and Assessing Utility.* Working Paper M09/02. University of Southampton.

Srinivas, M. and Patnaik, L. M. (1994). *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*. IEEE Transactions on systems, Man and Cybernetics; 24(4). pp.656-668.

Thierens, D. (2002) *Adaptive mutation rate control schemes in genetic algorithms.* Evolutionary Computation*, 2002. CEC '02.* Proceedings of the 2002 Congress on,vol.1, pp. 980 – 98