The Application of Genetic Algorithms to Data Synthesis: A Comparison of Three Crossover Methods

DOI:
10.1007/978-3-319-99771-1_11

Document Version
Accepted author manuscript

The Application of Genetic Algorithms to Data Synthesis: A Comparison of Three Crossover Methods

Yingrui Chen[1], Mark Elliot[2], and Duncan Smith[3]

[1] University of Manchester, Manchester, M13 9PL, UK
Yingrui.chen@postgrad.manchester.ac.uk
[2] University of Manchester, Manchester, M13 9PL, UK
Mark.Elliot@manchester.ac.uk
[3] University of Manchester, Manchester, M13 9PL, UK
Duncan.G.Smith@manchester.ac.uk

Abstract. Data synthesis is a data confidentiality method which is applied to microdata to prevent leakage of sensitive information about respondents. Instead of publishing real data, data synthesis produces an artificial dataset that does not contain the real records of respondents. This, in particular, offers significant protection against reidentification attacks. However, effective data synthesis requires retention of the key statistical properties of (and respecting the multiple utilities of) the original data. In previous work, we demonstrated the value of matrix genetic algorithms in data synthesis [4]. The current paper compares three crossover methods within a matrix GA: parallelised (two-point) crossover, matrix crossover, and parametric uniform crossover. The crossover methods are applied to three different datasets and are compared on the basis of how well they reproduce the relationships between variables in the original datasets.

Keywords: Genetic algorithms, Data synthesis, Data privacy

1 Introduction

Published data are provided in many formats, although the underlying data are often microdata collected from some population [5]. Confidentiality protection techniques for microdata attempt to camouflage sensitive information in the original data while retaining its statistical properties for analysts. Data synthesis is a protection technique that produces a synthetic dataset that is designed to

preserve the same statistical properties as the original data and provide sufficient variables to allow proper multivariate analyses [1].

The quality of synthetic data is strongly dependent on the design of the synthetic data generator [7]. Properties that are not explicitly included in the generator will not be present in the synthetic dataset (unless they are structurally or statistically related to properties that are, and therefore emerge from the synthesis process). Unforeseen analysis on fully synthetic data may therefore lead to different results from the same analysis on the original data [8].

In this paper we use Genetic Algorithms(GAs) to generate synthetic data. GAs are iterative optimising algorithms that simulate the process of natural evolution. They comprise of three main operators: selection, crossover and mutation. A group of candidate solutions are specified (the initial population). The fitnesses of these candidates are calculated and a selection operator selects a subset of the fitter candidates which are used to generate a new population. In crossover some pairs of these selected candidates are combined (using a variety of methods) to produce new candidate solutions. Some candidates are then subjected to mutation – random changes that will produce changes in fitness. After crossover and mutation we have the new population / generation. The process is repeated a number of times in order to (hopefully) generate fitter solutions than those in the initial population. Crossover and mutation rates can be varied from one iteration to the next, and tuning of these parameters can greatly influence performance.

GAs have been proposed as a potential method to protect respondents' from disclosures from published data. For example, Navarro-Arribas and Torra [9] mentioned that data protection could be treated as an optimisation problem with conflicting objectives and cite GAs as one approach to delivering this. Reasons for using GAs to produce synthetic data are: (i) they are designed to solve problems that have no observable solution space. The *a priori* knowledge required for setting up the initial population is minimal. (ii) GAs are interruptable so do not require complete *a priori* knowledge to set up objectives and, most crucially, (iii) GAs work well at optimising across competing constraints and therefore could, if well designed, have advantages over orthodox statistical model based synthesizers in: ameliorating overfitting, generating emergent properties and accommodating unforeseen analyses.

Matrix GAs are believed to capable of representing and solving more complex problem structures than the more orthodox bitstring GAs [12] [14] [15]. Although GAs have been used in various optimisation problems, the exploration of applications for matrix GAs has been limited. However, given that micro-

data are essentially matrices the production of synthetic microdata seems an obvious application. In previous work we have evaluated the potential for matrix GAs with promising initial results [3] [4]. The current paper explores the performance of three different crossover methods for matrix GAs in producing synthetic data. We consider three datasets, with different data structures, and sampled from different survey populations.

Note that in this initial phase of this research, we are concerned only with optimising the utility of the synthesised data and not with the residual disclosure risk. The rationale for this is twofold: (i) optimising the utility of a synthetic dataset represents a difficult problem by itself and adding in the contrary constraint of disclosure control will introduce further complexity, and (ii) of the two elements the utility problem is the more significant for synthetic data; if this cannot be solved the efficiency of the risk optimisation will be irrelevant. Understanding the properties of the utility optimisation problem before introducing the complexity disclosure control as an objective is therefore the appropriate research strategy.

1.1 Microdata and Contingency Tables

A microdata set for n cases and m variables is usually represented as an n by m matrix indexed $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Here we use Y to denote and original dataset and its synthetic version is denoted as X. X shares the same structure as Y as illustrated in Fig. 1.

$$\begin{pmatrix} y_{11} & y_{12} & \ldots & y_{1m} \\ y_{21} & y_{22} & \ldots & y_{2m} \\ y_{31} & y_{32} & \ldots & y_{3m} \\ y_{41} & y_{42} & \ldots & y_{4m} \\ \vdots & \vdots & \vdots & \vdots \\ y_{n1} & y_{n2} & \ldots & y_{nm} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ x_{31} & x_{32} & \ldots & x_{3m} \\ x_{41} & x_{42} & \ldots & x_{4m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix}$$

Fig. 1. Microdata Y and its synthetic version X

For categorical variables the same information can be encoded in a contingency table, which captures the between-variate structure of the candidate. Assume our variables take values in finite sets I_j so that $I = \underset{j \in [1..m]}{\times} I_j$ denotes the possible configurations of the variables. Then a contingency table is an m-dimensional table containing a count for each member of I. For example, if we

denote the jth column of a microdata set Y as $Y_{:,j}$, then the 2-dimensional contingency table constructed from distinct columns $Y_{:,j}$ and $Y_{:,k}$ is $CT(Y_{:,j}, Y_{:,k})$ with entries $n_{r,c}$ is,

$$n_{r,c} = \sum_{i=1}^{n} [Y_{i,j} = (I_j)_r \wedge Y_{i,k} = (I_k)_c] \tag{1}$$

where the square brackets are Iverson brackets and the levels of I_j and I_k are indexed $r \in [1..|I_j|]$ and $c \in [1..|I_k|]$ respectively.

1.2 Objectives

Respecting variable associations in the original data is an important aspect of producing high quality synthetic data. Thus, objective functions are designed based on the differences between synthetic (contingency) tables and original tables in low dimensions. A measure of the difference between a pair of contingency tables is the Jensen-Shannon distance D_{JS} between their normalised (to sum to 1) counterparts[4]. Suppose P and Q are two discrete probability distributions, then $D_{JS}(P||Q)$ is given by:

$$D_{JS}(P||Q) = (\frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M))^{\frac{1}{2}} \tag{2}$$

where $M = \frac{1}{2}(P + Q)$ and D_{KL} is the well-known Kullback-Leibler divergence.

So our distance measure for a pair of 2-dimensional contingency tables is defined as:

$$\Delta(X, Y, \{j, k\}) = D_{JS}(\frac{1}{n} CT(X_{:,j}, X_{:,k}) || \frac{1}{n} CT(Y_{:,j}, Y_{:,k})) \tag{3}$$

Our first objective function is the mean of these distances over all pairs of variables:

$$F_1(X, Y) = \binom{m}{2}^{-1} \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} \Delta(X, Y, \{j, k\}) \tag{4}$$

[4] Regarding the choice of divergence measure. The Kullback-Leibler divergence cannot be used directly because of the requirement for absolute continuity. Aside from that constraint there was no prior compelling reason for picking any specific measure, and there is no specific empirical work to guide us. The Jensen-Shannon distance was chosen mainly on the basis that it is a true metric, unlike e.g. the Jensen-Shannon divergence. The impact of using alternative measures is another area which future research could explore

Analogous measures are also considered for all 3-dimensional and all 4-dimensional contingency tables. So our other two objectives are defined as:[5]

$$F_2(X,Y) = \binom{m}{3}^{-1} \sum_{S \in P_3([1..m])} \Delta(X,Y,S) \tag{5}$$

$$F_3(X,Y) = \binom{m}{4}^{-1} \sum_{S \in P_4([1..m])} \Delta(X,Y,S) \tag{6}$$

where $P_k(Z)$ denotes the members of the powerset of Z of size K.

The fitness of each candidate is calculated by the Euclidean distance from the synthetic to the original data in the space delineated by the three objective functions. The fitness value is normalized to the range $[0,1]$ by dividing by $\sqrt{3}$.[6] So our overall objective function is:

$$F = \sqrt{3}^{-1} \sqrt{(F_1(X,Y)^2 + F_2(X,Y)^2 + F_3(X,Y)^2} \tag{7}$$

2 Crossover Methods

A crossover operator produces variation in a GA population. The operators considered here will change a pair of individuals by swapping randomly selected sub-matrices. In the case of uniform crossover these sub-matrices will necessarily have dimension 1×1 and we will essentially be swapping individual elements of the matrices.

The three crossover methods presented here have been used previously in various application areas, but not usually compared and certainly not in the context of synthetic data generation. Two of them use the mechanism of two-point crossover where not all sub-matrices (or elements) have an equal probability of being swapped (positional bias). The third operator, uniform crossover, does not suffer from positional bias and is included in order to examine the impact of positional bias on the effectiveness of matrix GA data synthesizers.

Parallelised Crossover The design of parallelised crossover is based on a two-point crossover method from linear GAs, which swaps the elements between two randomly selected crossover points a and b between a pair of bitstrings. Since

[5] Clearly this is not a complete set of possible objectives but these are probably necessary to produce reasonable synthetic categorical data and provide sufficient complexity for our crossover experiments

[6] So on this scale 0 is the best fitness possible and 1 is the worst

solutions that GAs generate are operationally independent (in that a change in one individual has no direct effect on another), crossover and mutation can be parallelised [2]. Parallelised crossover occurs on a single variable and therefore it is possible to have m sub-processors working separately on different variables in the generator. The generator works by randomly choosing a sub-matrix from within a single data column and swapping with the corresponding sub-matrix in the paired candidate. Fig. 2 illustrates parallelised crossover between a pair of candidates X^1 and X^2:

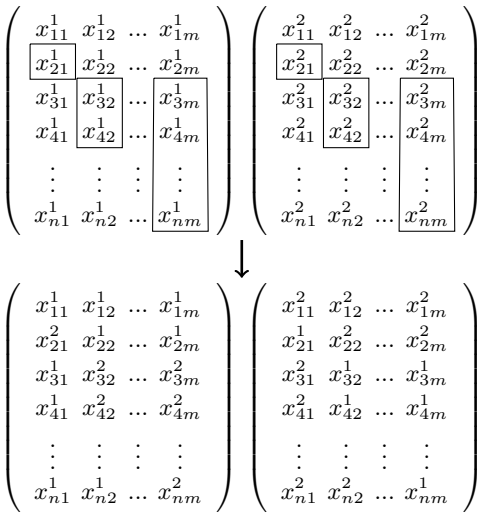

Fig. 2. X^1 and X^2 in parallelized crossover

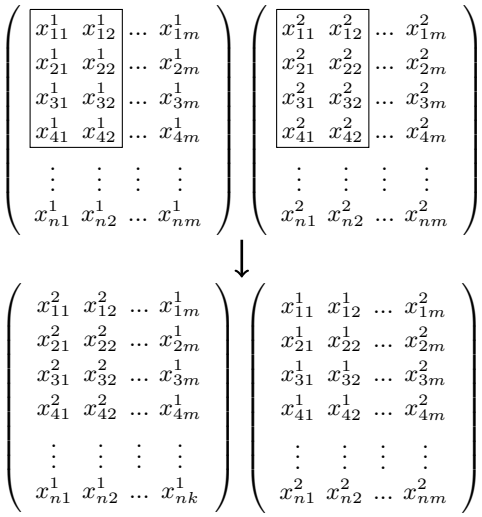

Fig. 3. X^1 and X^2 in Matrix crossover

Matrix Crossover Matrix crossover was first proposed by Wallet et al. [15]. Unlike parallelised crossover, matrix crossover generates crossover points for the rows as well as columns. Thus it swaps elements from a randomly generated sub-matrix (as opposed to the column vectors swapped in parallelised crossover). Fig. 3 illustrates matrix crossover.

Parametric Uniform Crossover (PUC) In PUC, the probability of crossover being applied to each element (1×1 sub-matrix) of the given candidate is determined by a user-specified parameter P_0. Fig. 4 illustrates PUC.

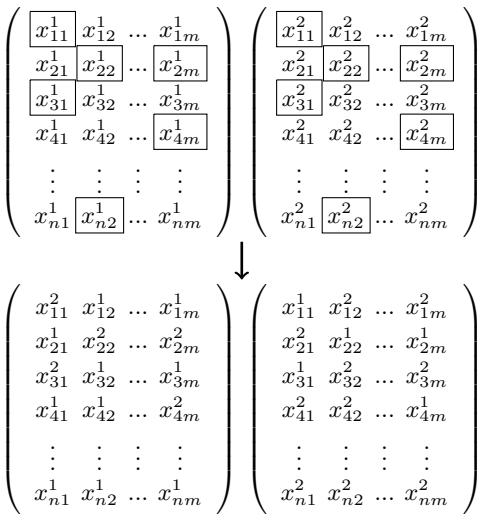

Fig. 4. X^1 and X^2 in PUC

2.1 Positional Bias

Both parallelised crossover and matrix crossover are based on the idea of two-point crossover. In parallelised crossover, the element with row index i will be swapped if, and only if, one of the selection points has index not greater than i while the other has index greater than i. Thus the swap probability is the hypergeometric probability:

$$P(\min(a,b) \leq i < \max(a,b)) = i(n-i+1)\binom{n+1}{2}^{-1} \tag{8}$$

where a and b are the indices of a pair of (distinct) randomly chosen crossover points.

It is trivial to show that this is a montone increasing function of i where $i < \frac{n}{2}$ and a monotone decreasing function of i where $i > \frac{n}{2}$.

For matrix crossover we also select a pair of crossover points for the columns and the probability of an element with index (i, j) being swapped is a product of hypergeometric probabilities. PUC, on the other hand contains no positional bias and it is therefore useful to provide us with an implicit evaluation of the effect of positional bias on the optimising ability of the matrix GA data synthesizer.

3 Empirical Study

3.1 Design

The three crossover methods were compared using three datasets that were each sampled from a different social survey. All three datasets contain 10 variables and 1000 cases. Dataset 1 was sampled from the Crime Survey for England and Wales, 2015-2016 [10] and has 10 binary variables. Dataset 2 was sampled from European Union Statistics on Income and Living Conditions, 2009 [11]. It has 6 binary variables, 1 three-category variable and 3 four-category variables. Dataset 3 was sampled from the Citizenship Survey, 2010-2011 [6]. It contains 1000 cases and 10 variables including 4 binary variables, 2 four-category variables, 2 six-category variables, 1 nine-category variable and 1 eleven-category variable.

For each dataset there was a fixed initial population of 100 candidates that was generated by independently sampling (with replacement) from the univariate distributions of the original data. Deterministic tournament selection[7] was used to select candidates with tournament size $t = 2$.

Synthetic data were generated using GAs with two distinct crossover rates. Matrix crossover used rates of 1.0 and 0.7. The corresponding crossover rates for parallelised crossover and PUC were chosen so that the probability of swapping individual elements was similar.

The synthetic data generator used a low mutation rate ($p_m = 0.01$) to reduce the noise in the final results.[8] Candidates chosen for mutation had a randomly selected sub-matrix swapped with data independently sampled from the original univariate distributions.

For each set of parameters we generated 10 synthetic populations. Each such trial was run for 100 generations.

[7] In generalised tournament selection, candidates are randomly selected into tournaments of size t (with or without replacement). The probability that a candidate wins the tournament and enters crossover is given by $p(1-p)^r$ where p is a parameter (such that $1/t < p \leq 1$) and r is the rank of the candidate's fitness within the tournament. In deterministic tournament selection p is set to 1.

[8] Mutation is another important operator in GA that helps find more promising candidates from the solution space and reduces the risk of becoming caught in local optima. However it can also reduce the fitness of a candidate. Here our focus is on comparing crossover operators so we selected a low mutation rate to reduce the noise in the final results. In future work will examine the relationship between the two operators.

Table 1. Fitness values of the initial population of each of the three test dataset

Fitness of Fixed Population (size=100) for each Data			
	Best fitness	Mean	s.d.
Data 1	0.0734	0.0800	0.0034
Data 2	0.2176	0.2259	0.0031
Data 3	0.2544	0.2610	0.0027

3.2 Experimental Results

Table 2 shows the means and standard deviations of the fittest solutions in the final (100th) populations. The rightmost column shows the fitness of the best individual that was generated over the 10 trials.

The generator used the objective function in Equation 7. The number of individual contingency tables compared depends on the number of variables and would increase substantially if we extended the measure to, say, 5-dimensional tables. Table 2 shows that the fitness of candidates for Dataset 1 is always closer to the original data compared with the other two no matter which crossover operator is used, followed by Dataset 2 and Dataset 3. This is monotonic with the complexity of the data structures of the three datasets. This issue will need further exploration to establish how the degree of complexity affects the viability of GA generated synthesis.

The experimental results also indicate that positional bias does impact the effectiveness of matrix GA generator. All the best means and individuals after 100 generations for the three datasets are generated by the synthesizer with the PUC operator that has $p_0 = 0.3818$. The second best mean and individuals are generated by the same synthesizer with $p_0 = 0.1911$.

Moreover, there was a significant improvement on the initial population no matter which crossover method was used. The increase of fitness (decrease in distance from the original data) indicates that matrix GA is efficient in generating synthetic data with real-coded data or even more complex data structures. Table 3 shows the mean improvement of the fitness of the population from the beginning to the 100th generation over all trials.

4 Conclusions

Our experimental results indicate that PUC performs better than matrix and parallelised crossover in producing synthetic data for all three datasets. This is likely to be due to the lack of positional bias. Results also indicate that the

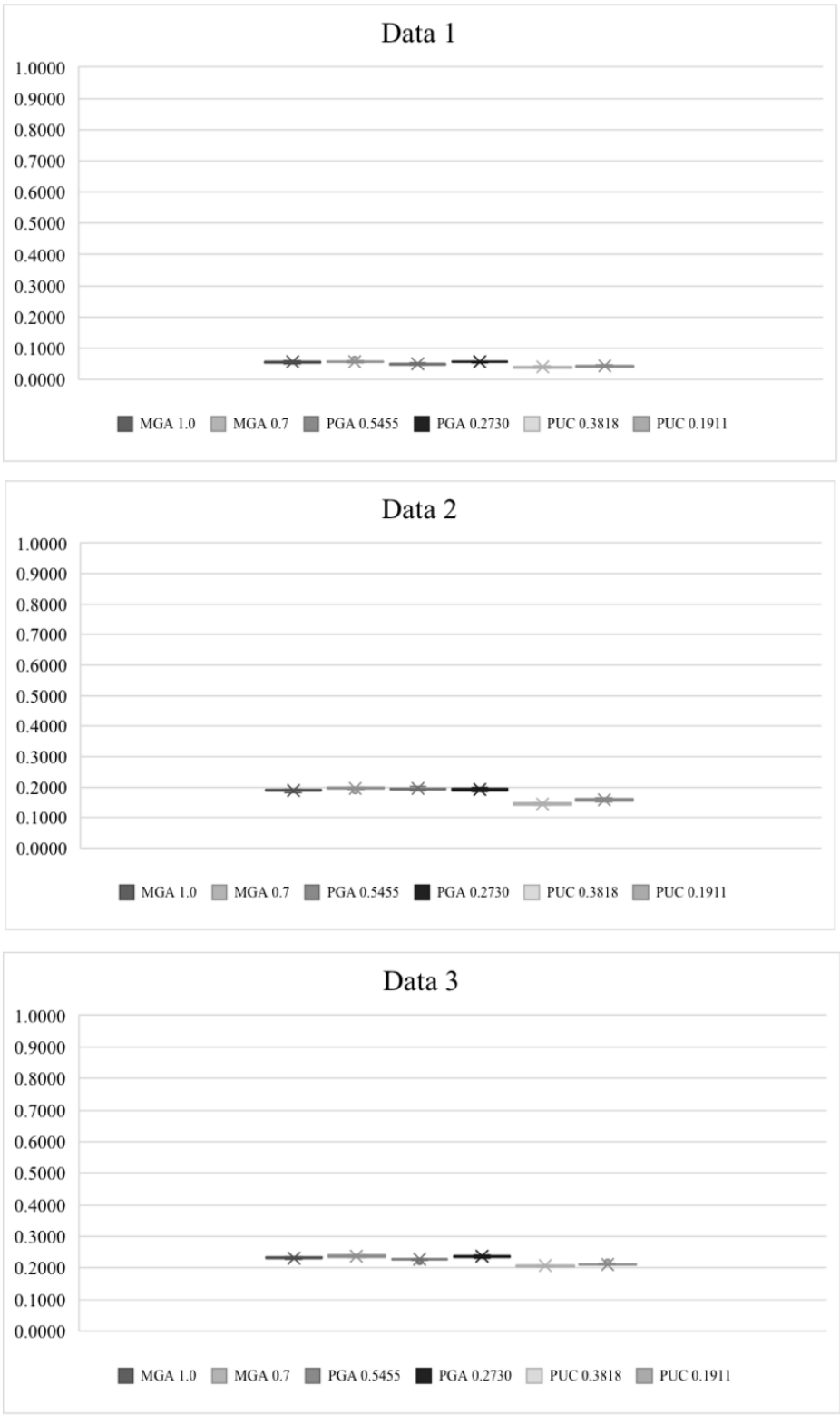

Fig. 5. Box plots of final fitness values of the best individual for Dataset 1, 2 and 3 from ten trials of matrix GA synthetic data generator using different crossover operators

Table 2. Summary statistics from ten trials of matrix GA data synthesizers equipped with three different operators: matrix crossover (MGA), parallelized crossover (PGA) and PUC.

Crossover Type	Crossover Rate	Data	Best Fitness Value		Best Individual
			Mean	s.d.	
MGA	1	Data 1	0.0564	0.0026	0.0517
		Data 2	0.1887	0.0030	0.1818
		Data 3	0.2319	0.0020	0.2281
	0.7	Data 1	0.0579	0.0022	0.0541
		Data 2	0.1958	0.0034	0.1889
		Data 3	0.2375	0.0032	0.2340
PGA	0.5455	Data 1	0.0497	0.0022	0.0472
		Data 2	0.1936	0.0038	0.1885
		Data 3	0.2259	0.0019	0.2213
	0.273	Data 1	0.0561	0.0015	0.0533
		Data 2	0.1929	0.0041	0.1867
		Data 3	0.2363	0.0025	0.2315
PUC	0.3818	Data 1	0.0393	0.0021	0.0362
		Data 2	0.1450	0.0025	0.1408
		Data 3	0.2060	0.0009	0.2048
	0.1911	Data 1	0.0429	0.0022	0.0397
		Data 2	0.1576	0.0038	0.1521
		Data 3	0.2112	0.0020	0.2092

performance of matrix GA on synthetic data generation strongly depends on the structure of data and the number of cases. For example, the optimisation of Dataset 1 is the most effective because it has the simplest data structure (containing only binary variables) compared to Dataset 2 and Dataset 3.

Beyond the issue of positional bias, the overall performance for all three crossover operators in producing synthetic data is reasonable. All approaches significantly improved the fitness of the 100 candidates from the initial population over 100 generations.

Our future research will focus on testing the effectiveness and practicality of the matrix GA generator by introducing adaptive crossover rates, more objectives and larger datasets. A key element missing from these initial experiments has been the assessment of disclosure risk. As outlined in the introduction, this was a rational approach to isolate the difficult problem of optinmising utility. However, a full GA data synthesiser should incorporate risk. Therefore, in future work we will bring measures of disclosure risk into the GA framework. In

Table 3. Mean fitness improvement over ten trials on the best fitness value of initial population

On mean of the best fitness values over all trials	
Data 1	0.0296
Data 2	0.0470
Data 3	0.0362

many ways this is when the GA approach will come into its own. The risk utility trade-off is usually dealt with as a two step-process and optimisng both within a single framework is likely to be more efficient.

Overall, these initial experiments using matrix GA generators to generate synthetic data show that matrix GA is of interest for the problem of data synthesis and for solving problems with higher-dimensional and complex structures in general.

References

1. J. M. Abowd, and J. Lane, New approaches to confidentiality protection: Synthetic da-ta, remote access and research data centers; In Privacy in statistical databases, Springer Berlin Heidelberg, 282-289. (2004)
2. E. Cantu-Paz and D. Goldberg, Efficient Parallel Genetic Algorithms: Theory and Prac-tice, Computer Methods in Applied Mechanics and Engineering, vol.186, 221-238. (2000)
3. Y. Chen, M. Elliot and J. Sakshaug, A Genetic Algorithm Approach to Synthetic Data Production, in Proceedings of the 1st International Workshop on AI for Privacy and Se-curity. Article No. 13. (2016)
4. Y. Chen, M. Elliot and J. Sakshaug, 2017. Genetic Algorithms in Matrix Representation and Its Application in Synthetic Data, UNECE Work Session on Statistical Data Con-fidentiality. `https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2017/2_Genetic_algorithms.pdf`. Last access 20/12/2017. (2017)
5. V. Ciriani, S. D. C. di Vimercati, S. Foresti and P. Samarati, Microdata protection, in Se-cure Data Management in Decentralized Systems, T Yu, and S. Jajodia (edt.) Springer, New York, 291321. (2007)
6. Department for Communities and Local Government, Ipsos MORI. Citizenship Survey, 2010-2011. [data collection]. UK Data Service. SN: 7111, `http://doi.org/10.5255/UKDA-SN-7111-1`, Last access: 20/12/2017. (2012)
7. J. Drechsler, Synthetic data, where do we come from? Where do we want to go?, in Synthetic Data Workshop; Office of National. (2014)

8. O. Maimon and L. Rokach, Data Mining and Knowledge Discovery Handbook, Springer Science and Business Media; p. 704. (2010)
9. G. Navarro-Arribas and V. Torra, Advanced Research on Data Privacy in the ARES Pro-ject; Advanced Research in Data Privacy; Studies in Computational Intelligence. Vol. 567. Springer Switzerland; 3-14. (2015)
10. Office for National Statistics. Crime Survey for England and Wales, 2015-2016. [data collection]. UK Data Service. SN: 8140, http://doi.org/10.5255/UKDA-SN-8140-1. Last access 11/01/2018. (2017)
11. Office for National Statistics. Social Survey Division, Northern Ireland Statistics and Research Agency, Eurostat. (2011). European Union Statistics on Income and Living Conditions, 2009. [data collection]. UK Data Service. SN: 6767, http://doi.org/10.5255/UKDA-SN-6767-1. Last access 11/01/2018. (2009)
12. 3. P. Pongcharoen, A. Khadwilard, and A. Klakankhai, Multi-matrix Real-coded Genetic Algorithm for Minimising Total Costs, in Logistics Chain Network, World Academy of Science, Engineering and Technology International Journal of Economics and Man-agement Engineering, Vol:1, No.11, 574-597. (2007)
13. M. Srinivas and L. M. Patnaik, Genetic algorithms: A Survey, Computer, vol. 27, no. 6, 17-26. (1994)
14. L. Sun, Y. Zhang, and C. Jiang, A matrix real-coded genetic algorithm to the unit commitment problem, in Electric Power Systems Research; 76; pp.716-728. (2006)
15. B. C. Wallet, D. J. Marchette and J. L. Solka, A matrix Representation for Genetic Algorithms, in Proceedings of Automatic Object Recognition IV of SPIE Aerosense, Na-val Surface Warfare Center Dahlgren; Virginia. (1996)