A simulated annealing algorithm for joint stratification and sample allocation

Mervyn O'Luing,[1] Steven Prestwich,[1] S. Armagan Tarim[2]

Abstract

This study combines simulated annealing with delta evaluation to solve the joint stratification and sample allocation problem. In this problem, atomic strata are partitioned into mutually exclusive and collectively exhaustive strata. Each partition of atomic strata is a possible solution to the stratification problem, the quality of which is measured by its cost. The Bell number of possible solutions is enormous, for even a moderate number of atomic strata, and an additional layer of complexity is added with the evaluation time of each solution. Many larger scale combinatorial optimisation problems cannot be solved to optimality, because the search for an optimum solution requires a prohibitive amount of computation time. A number of local search heuristic algorithms have been designed for this problem but these can become trapped in local minima preventing any further improvements. We add, to the existing suite of local search algorithms, a simulated annealing algorithm that allows for an escape from local minima and uses delta evaluation to exploit the similarity between consecutive solutions, and thereby reduces the evaluation time. We compared the simulated annealing algorithm with two recent algorithms. In both cases, the simulated annealing algorithm attained a solution of comparable quality in considerably less computation time.

Keywords: Simulated annealing algorithm; Optimal stratification; Sample allocation; R software.

[1] *Insight Centre for Data Analytics, Department of Computer Science, University College Cork, Ireland.* Email: mervyn.oluing@insight-centre.org, steven.prestwich@insight-centre.org

[2] *Cork University Business School, University College Cork, Ireland.* Email: armagan.tarim@ucc.ie

1 Introduction

In stratified simple random sampling, a population is partitioned into mutually exclusive and collectively exhaustive strata, and then sampling units from each of those strata are randomly selected. The purposes for stratification are discussed in Cochran et al. (1977). If the intra-strata variances were minimized then precision would be improved. It follows that the resulting small samples from each stratum can be combined to give a small sample size.

To this end, we intend to construct strata which are internally homogeneous but which also accommodate outlying measurements. To do so, we adopt an approach which entails searching for the optimum partitioning of *atomic strata* (however, the methodology can also be applied to *continuous* strata) created from the Cartesian product of categorical stratification variables, see Benedetti et al. (2008); Ballin and Barcaroli (2013, 2020).

The Bell number, representing the number of possible partitions (stratifications) of a set of atomic strata, grows very rapidly with the number of atomic strata (Ballin and Barcaroli, 2013). In fact, there comes a point where, even for a moderate number of atomic strata and the most powerful computers, the problem is intractable, i.e. there are no known efficient algorithms to solve the problem.

Many large scale combinatorial optimisation problems of this type cannot be solved to optimality, because the search for an optimum solution requires a prohibitive amount of computation time. This compels one to use *approximisation algorithms* or *heuristics* which do not guarantee optimal solutions, but can provide approximate solutions in an acceptable time interval. In this way, one trades off the quality of the final solution against computation time (Van Laarhoven and Aarts, 1987). In other words, heuristic algorithms are developed to find a solution that is "good enough" in a computing time that is "small enough" (Sörensen and Glover, 2013).

A number of heuristic algorithms have been developed to search for optimal or near optimal solutions, for both univariate and multivariate scenarios of this problem. This includes the hierarchichal algorithm proposed by Benedetti et al. (2008), the genetic algorithm proposed by Ballin and Barcaroli (2013) and the grouping genetic algorithm proposed by O'Luing et al. (2019). Although effective, the evaluation function in these algorithms can be costly in terms of running time.

We add to this work with a simulated annealing algorithm (SAA) (Kirkpatrick et al., 1983; Černỳ, 1985). SAAs have been found to work well in problems such as this, where there are many local minima and finding an approximate global solution in a fixed amount of computation time is more desirable than finding a precise local minimum (Takeang and Aurasopon, 2019). We present a SAA to which we have added delta evaluation (see section 5) to take advantage of the similarity between consecutive solutions and help speed up computation times.

We compared the performance of the SAA on atomic strata with that of the grouping genetic algorithm (GGA) in the *SamplingStrata* package (Ballin and Barcaroli, 2020). This algorithm implements the grouping operators described by O'Luing et al. (2019). To do this, we used sampling frames of varying sizes containing what we assume to be completely representative details for target and auxiliary variable columns.

Further to the suggestion of a *Survey Methodology* reviewer, we subsequently compared the SAA with a traditional genetic algorithm (TGA) used by Ballin and Barcaroli (2020) on continuous strata. In both sets of experiments, we used an initial solution created by the k-means algorithm (Hartigan and Wong, 1979) in a two-stage process (see section 2.3 for more details).

Section 2 provides background information on atomic strata, introduces the SAA and motivates the addition of delta evaluation as a means to improve computation time. Two-stage simulated annealing is also discussed. Section 3 of the paper describes the cost function and evaluation algorithm. Section 4 provides an outline of the SAA. Section 5 presents the improved SAA with delta evaluation. Section 6 provides a comparison of the performance of the SAA with the GGA using an initial solution and fine-tuned hyperparameters. Section 7 then provides details of the comparison of the SAA with the genetic algorithm in Ballin and Barcaroli (2020) on continuous strata. Section 8 presents the conclusions and section 9 suggests some further work. The appendix - section 10 contains background details on precision constraints, the hyperparameters, and the process of fine-tuning the hyperparameters for both comparisons as well as the computer specifications.

2 Background information

2.1 Stratification of atomic strata

Atomic strata are created using categorical auxiliary variable columns such as *age group*, *gender* or *ethnicity* for a survey of people or *industry*, *type of business* and *employee size* for business surveys. The cross-classification of the class-intervals of the auxiliary variable columns form the atomic strata.

Auxiliary variable columns which are correlated to the target variable columns may provide a gain in sample precision or *similarity.* Each target variable column, y_g, contains the value of the survey characteristic of interest, e.g. *total income*, for each population element in the sample.

Once these are created, we obtain summary statistics, such as the number, mean and standard deviation of the relevant observed values, from the one or more target variable columns that fall within each atomic stratum. The summary information is then aggregated in order to calculate the means and variances for each stratum which in turn are used to calculate the sample allocation for a given stratification.

The partitioning of atomic strata that provides the *global minimum* sample allocation, i.e. the minimum of all possible sample allocations for the set of possible stratifications, is known as an *optimal stratification.* There could be a multiple of such partitionings. Although an optimum stratification is *the* solution to the problem, each stratification represents a solution of varying quality (the lower the cost (*minimum* or *optimal* sample allocation) the higher the quality). For each stratification, the cost is estimated by the Bethel-Chromy algorithm (Bethel, 1985, 1989; Chromy, 1987). A more detailed description, and discussion of the methodology for this approach for joint determination of stratification and sample allocation, can be found in Ballin and Barcaroli (2013).

2.2 Simulated annealing algorithms

The basic principle of the SAA (Kirkpatrick et al., 1983; Černỳ, 1985) is that it can accept solutions that are inferior to the current best solution in order to find the global minima (or maxima). It is one of several stochastic local search algorithms, which focus their attention within a local neighbourhood of a given initial solution (Cortez, 2014), and use different stochastic techniques to escape from attractive local minima (Hoos and Stützle, 2004).

Based on physical annealing in metallurgy, the SAA is designed to simulate the controlled cooling process from liquid metal to a solid state (Luke, 2013). This controlled cooling uses the temperature parameter to compute the probability of accepting inferior solutions (Cortez, 2014). This acceptance probability is not only a function of the temperature, but also the difference in cost between the new solution and the current best solution. For the same difference in cost, a higher temperature means a higher probability of accepting inferior solutions.

For a given temperature, solutions are iteratively generated by applying a small, randomly generated, perturbation to the current best solution. Generally, in SAAs, a perturbation is the small displacement of a randomly chosen particle (Van Laarhoven and Aarts, 1987). In the context of our problem, we take perturbation to mean the displacement (or re-positioning) of q (generally $q = 1$) randomly chosen atomic strata from one randomly chosen stratum to another.

With a perturbation, the current best solution transitions to a new solution. If a perturbation results in a lower cost for the new solution, or if there is no change in cost, then that solution is always selected as the current best solution. If the new solution results in a higher cost, then it is accepted at the above mentioned acceptance probability.

This acceptance condition is called the Metropolis criterion (Metropolis et al., 1953). This process continues until the end of the sequence, at which point the temperature is decremented and a new sequence begins.

If the perturbations are minor, then the current solution and the new solution will be very similar. Indeed, in our SAA we are assuming only a slight difference between consecutive solutions owing to such perturbations (see section 4.1 for more details). For this reason we have added delta evaluation, which will be discussed further in section 5, to take advantage of this similarity and help improve computation times.

Accordingly, and as mentioned in the introduction, we present a SAA with delta evaluation and compare it with the GGA when both are combined with an initial solution. We also compare it with a genetic algorithm used by Ballin and Barcaroli (2020) on continuous strata. We provide more background details on initial solutions in section 2.3 below.

2.3 Two-stage simulated annealing

A two stage simulated annealing process, where an initial solution is generated by a heuristic algorithm in the first stage, has been proposed for problems such as the *cell placement problem* (Grover, 1987; Rose et al., 1988) or the *graph partitioning problem* (Johnson et al., 1989). Lisic et al. (2018) combined an initial solution, generated by the k-means algorithm, with a simulated annealing algorithm, for a problem similar in nature to this problem, but where the sample allocation as well as strata number are fixed, and the algorithm searches for the optimal arrangement of sampling units between strata.

The simulated annealing algorithm used by Lisic et al. (2018) starts with an initial solution (stratification and sample allocation to each stratum) and, for each iteration, generates a new candidate solution by moving one atomic stratum from one stratum to another and adjusting the sample allocation for that stratification. Each candidate solution is then evaluated to measure the coefficient of variation (CV) of the target variables and is accepted, as the new current best solution, if its objective function is less than the preceding solution. Inferior quality solutions are also accepted at a probability, ρ, which is a function of a tunable temperature parameter and the change in solution quality between iterations. The temperature cools, at a rate which is also tunable, as the number of iterations increases.

Following this work, Ballin and Barcaroli (2020) recommended combining an initial solution, generated by k-means, with the grouping and traditional genetic algorithms. They demonstrate that the k-means algorithm provides better starting solutions when compared with the starting solution generated by a stochastic approach. We also combine a k-means initial solution with the SAA in the experiments described in sections 6 and 7.

3 The joint stratification and sample allocation problem

Our aim is to partition L atomic strata into H non-empty sub-populations or strata. A partitioning represents a stratification of the population. We aim to minimise the sample allocation to this stratification while keeping the measure of similarity less than or equal to the upper limit of precision, ε_g. This similarity is measured by the CV of the estimated population total for each one of G target variable columns, $\hat{T}_g$. We indicate by n_h the sample allocated to stratum h and the survey cost for a given stratification is calculated as follows:

$$C\left(n_1, \ldots, n_H\right) = \sum_{h=1}^{H} C_h n_h$$

where C_h is the average cost of surveying one unit in stratum h and n_h is the sample allocation to stratum h. In our analysis C_h is set to 1.

The variance of the estimator is given by:

$$\mathrm{VAR}\left(\hat{T}_g\right) = \textstyle\sum_{h=1}^{H} N_h^2 \left(1 - \frac{n_h}{N_h}\right) \frac{S_{h,g}^2}{n_h} \quad (g = 1, \ldots, G)$$

where N_h is the number of units in stratum h and $S^2_{h,g}$ is the variance of stratum h for each target variable column g.

As mentioned above ε_g is the upper precision limit for the CV for each $\hat{T}_g$:

$$CV(\hat{T}_g) = \frac{\sqrt{VAR(\hat{T}_g)}}{E(\hat{T}_g)} \leq \varepsilon_g \; .$$

The problem can be summarised in this way:

$$\begin{array}{ll} \min & n = \sum_{h=1}^{H} n_h \\ \text{subject to} & CV\left(\hat{T}_g\right) \leq \varepsilon_g \quad (g = 1, \ldots, G) \; . \end{array}$$

To solve the allocation problem for a particular stratification with the Bethel-Chromy algorithm the upper precision constraint for variable g can be expressed as follows:

$$CV\left(\hat{T}_g\right)^2 \leq \varepsilon_g^2 \equiv \textstyle\sum_{h=1}^{H} \frac{N_h^2 S^2_{h,g}}{n_h} - N_h S^2_{h,g} \leq E(\hat{T}_g^2)\varepsilon_g^2$$

$$\equiv \textstyle\sum_{h=1}^{H} \frac{N_h^2 S^2_{h,g}}{\left(E(\hat{T}_g^2)\varepsilon_g^2 + \sum_{h=1}^{H} N_h S^2_{h,g}\right)\, n_h} \leq 1.$$

Then we substitute $\frac{N_h^2 S^2_{h,g}}{\left(E(\hat{T}_g^2)\varepsilon_g^2 + \sum_{h=1}^{H} N_h S^2_{h,g}\right)}$ with ξ_h, g and replace the problem summary with the following:

$$\begin{array}{l} \min n = \sum_{h=1}^{H} n_h \\ \sum_{h=1}^{H} \frac{\xi_{h,g}}{n_h} \leq 1 \quad (g = 1, \ldots, G) \; . \end{array}$$

where $\frac{1}{n_h} > 0$. The Bethel-Chromy algorithm uses Lagrangian multipliers to derive a solution for each n_h.

$$\frac{1}{n_h} = \begin{cases} \frac{\sqrt{1}}{(\sqrt{\sum_{g=1}^{G} \alpha_g \xi_h, g} \sum_{h=1}^{H} \sqrt{\sum_{g=1}^{G} \alpha_g \xi_h, g})} & \text{if } \sum_{g=1}^{G} \alpha_g \xi_h, g > 0 \\ +\infty \text{ otherwise} \; . \end{cases}$$

where $\alpha_g = \frac{\lambda_g}{\sum_{g=1}^{G} \lambda_g}$, and λ_g is the Lagrangian multiplier (Benedetti et al., 2008). The algorithm starts with a default setting for each α_g and uses gradient descent to converge to a final value for them.

4 Outline of the simulated annealing algorithm

The SAA with delta evaluation is described in algorithm 1 below. We then describe the heuristics we have used in the SAA. Delta evaluation is explained in more detail in section 5.

Algorithm 1 Simulated annealing algorithm

```
function SIMULATEDANNEALING($S$ is the starting solution, $f$ is the evaluation function (Bethel-Chromy algorithm), $best$ is the current best solution, $BSFSF$ is the best solution found so far, $maxit$ is the maximum number of sequences, $J$ is the length of sequence, $T_{max}$ is the starting temperature, $T_{min}$ is the minimum temperature, $DC$ is the Decrement Constant, $L_{max\%}$ is a % of $L$ (number of atomic strata), $P(H+1)$ is the probability of a new stratum, $H+1$, being added)
    $T \leftarrow T_{max}$
    $best \leftarrow$ S
    $Cost(best) \leftarrow f(best)$                                ▷ using Bethel-Chromy algorithm
    while $i < maxit$ && $T > T_{min}$ do
        if RANDOM$(0,1) \leq 1/J$ then
            for $l = 1$ to $L$ do
                if RANDOM$(0,1) \leq P(H+1)$ then
                    move atomic stratum $l$ to new stratum $H+1$        ▷ see section 4.3
                end if
            end for
        end if
        for $j = 1$ to $J$ do
            if $i = 1$ & $j = 1$ then $q = L \times L_{max\%}$
            else if $i = 1$ & $j > 1$ then $q = ceiling(q \times 0.99)$    ▷ 0.99 is not tunable
            else if $i > 1$ then $q = 1$
            end if
            Randomly select $h$ and $h'$
            $next \leftarrow$ PERTURBATION($best$)
                                                   ▷ Assign $q$ atomic strata from $h$ to $h'$
            $Cost(next) \leftarrow f(next)$                        ▷ using delta evaluation
            $\Delta E \leftarrow$COST($next$) $-$ COST($best$)
            if $\Delta E \leq 0$ then
                $best \leftarrow next$
            else if RANDOM$(0,1) < e^{(-\frac{\Delta E}{T})}$ then        ▷ Metropolis Criterion
                $best \leftarrow next$
            end if
            if best <= $BSFSF$ then
                $BSFSF \leftarrow best$
            end if
        end for
        $T \leftarrow T * DC$

    end while
    return $BSFSF$
end function
```

4.1 Perturbation

Consider the following solution represented by the stratification:

$$\{1, 3\},\ \{2\},\ \{4, 5, 6\}$$

The integers within each stratum represent atomic strata. In perturbation, the new solution below is created by arbitrarily moving atomic strata, in this example $q = 1$, from one randomly chosen stratum to another.

$$\{1, 3, 2\}\ ,\ \{\emptyset\},\ \{4, 5, 6\}$$

The first stratum gains an additional atomic stratum $\{2\}$ to become $\{1, 3, 2\}$, whereas the middle or second stratum has been "emptied" (and is deleted), and there remains only two strata. Strata are only emptied when the last remaining atomic stratum has been moved to another stratum.

To clarify how this works in the algorithm: each solution is represented by a vector of integers - atomic strata which have the same integer are in the same stratum. A separate vector of the unique integers in the solution represents the strata. For example, the first solution $\{1, 3\}$, $\{2\}$, $\{4, 5, 6\}$ would be represented by the vector $[1 \quad 2 \quad 1 \quad 3 \quad 3 \quad 3]$ and the strata would be represented by the vector $[1 \quad 2 \quad 3]$. When the new solution is created, the second stratum has been removed and is no longer part of the solution. That is to say, the vector for the new solution is: $[1 \quad 1 \quad 1 \quad 3 \quad 3 \quad 3]$ and the strata vector is $[1 \quad 3]$. With stratum 2 removed, and for clarity, we rename stratum 3 to 2 so that this solution becomes: $[1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2]$, and the strata are now represented by the vector $[1 \quad 2]$. Strata $[1 \quad 2]$ will remain in any further solutions unless another stratum is "emptied" or a new stratum is added.

4.2 Evaluation and acceptance

Each new solution is evaluated using the Bethel-Chromy algorithm and the Metropolis acceptance criterion is applied. If accepted, the new solution differs from the previous solution only by the above mentioned perturbation. If it is not accepted, we continue with the previous solution, and again try moving q randomly chosen atomic strata between two randomly selected strata.

4.3 Sequences and new strata

This continues for the tunable length of the sequence, J. This should be long enough to allow the sequence to reach equilibrium. However, there is no rule to determine J. At the commencement of each new sequence, we have H strata in the current best solution. With a fixed probability of $1/J$, an additional stratum is added. If a new stratum is to be added, the SAA loops through each atomic stratum and moves it to a new stratum, which is called $H + 1$, because each stratum is labelled sequentially from 1 to H (see section 4.1), at a tunable probability, $P(H + 1)$. The algorithm runs for a tunable number of sequences, $maxit$.

4.4 Temperature

The temperature is decremented from a starting temperature, T_{max}, to a minimum temperature, T_{min}, or until $maxit$ has been reached. As we are starting with a near optimal solution, we select T_{max} as no greater than 0.01 and we set T_{min} to be 1.0×10^{-11}.

This is to allow for the advanced nature of the search, and allows the algorithm to focus more on the search for superior solutions, with an ever-reducing probability of accepting inferior solutions. However, a low temperature, T, does not always equate to a low probability of acceptance.

Small positive differences in solution quality (where the new solution has a marginally inferior quality to the current best solution), ΔE, occur often because we are starting with a good quality initial solution. Figure 4.1 demonstrates the probability of such solutions being accepted, $e^{(-\frac{\Delta E}{T})}$, increases the smaller this difference becomes for the same T. Nonetheless, figure 4.1 also demonstrates that for the same changes in solution quality as the T decreases, the probability also decreases (and it behaves increasingly like a hill climbing algorithm).

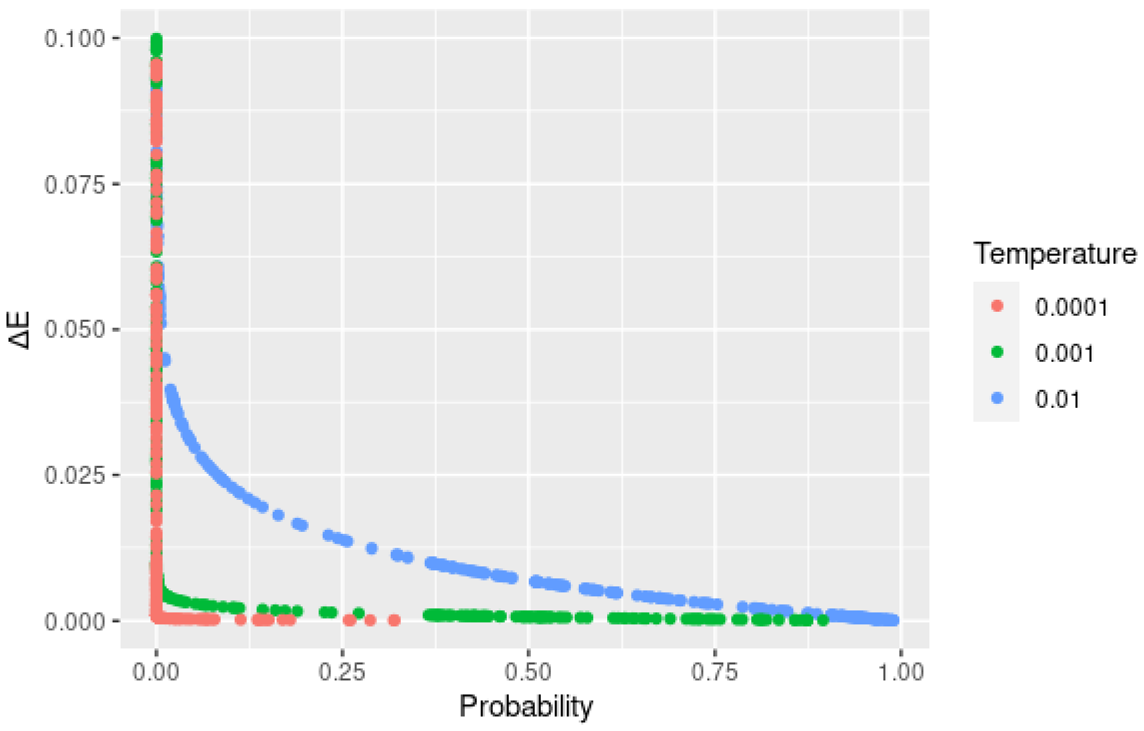

Figure 4.1: Probability of accepting an inferior solution as a function of ΔE and T

5 Improving the performance of the simulated annealing algorithm using delta evaluation

As outlined earlier, the only difference between consecutive solutions is that q atomic strata have been moved from one group into another. As with the other heuristics, q is also tunable, and for the first sequence we have added the option of setting $q > 1$ and reducing q for each new solution in the first sequence until $q = 1$. The reason for this is that, where $q > 1$, the increased size of the perturbation can help reduce the number of strata. In this case, we set q as a tunable percentage of the solution size, or of the number of atomic strata, L, to be partitioned. After the first sequence $q = 1$.

Furthermore, as the strata are mutually exclusive, this movement of q atomic strata from one stratum to another does not affect the remaining strata in any way. Ross et al. (1994) introduce a technique called delta evaluation, where the evaluation of a new solution makes use of previously evaluated similar solutions, to significantly speed up evolutionary algorithms/timetabling experiments. We use the similar properties of two consecutive solutions to apply delta evaluation to the SAA. It follows, therefore, that in the first sequence q should be kept low and the reduction to $q = 1$ should be swift.

The Bethel-Chromy algorithm requires the means and variances for each stratum in order to calculate the sample allocation. However, we use the information already calculated for the remaining $H - 2$ strata, and simply calculate for the two strata affected by the perturbation. Thus, the computation for the means and variances of the H strata is reduced to a mere subset of that otherwise required.

Now recall that the Bethel-Chromy algorithm starts with a default value for each α_g, and uses gradient descent to find a final value for each α_g. This search continues up to when the algorithm reaches a minimum step-size threshold, or alternatively exceeds a maximum number of iterations. This minimum threshold is characterised by ϵ, which is set as 1.0×10^{-11} in Ballin and Barcaroli (2020), and the maximum number of iterations is 200. We make the assumption that this search will be substantially reduced if we use the α_g values from the evaluation of the current solution as a starting point for the next solution.

The above two implementations of delta evaluation result in a noticeable reduction in computation times as demonstrated in the experiments described below.

6 Comparing the performance of the two algorithms

6.1 Evaluation plan

In this section, we outline the comparison of the performance of the grouping genetic algorithm with the simulated annealing algorithm. We used a number of data sets of varying sizes in these experiments. There are a number of regions in each data set (labelled here as domains). An optimal stratification and minimum sample allocation was selected for each domain.

The sum of the samples for all domains provides the total sample size. The sample size, or cost of the solution, defines the solution quality. For more details on domains refer to Ballin and Barcaroli (2013). The aim of these experiments was to consider whether the SAA can attain comparable solution quality with the GGA in less computation time per solution thus resulting in savings in execution times.

However, we also compared the total execution times as this is a consequence of the need to train the hyperparameters for both algorithms. More details are available in the appendix - see section 10.4.

We tabulate the results of these experiments in section 6.4 where for comparison purposes we express the SAA results as a ratio of those for the GGA.

6.2 Comparing the number of solutions generated

After the first iteration the GAA retains the elite solutions, E, from the previous iteration. These are calculated by the product of the elitism rate (the proportion of the chromosome population which are elite solutions), E_R, and the chromosome population size (the number of candidate solutions in each iteration), N_P. As E have already been evaluated they are not evaluated again.

For this reason, we compared the evaluation times for the evaluated solutions in the GGA with all those of the SAA. For the GGA, the total number of evaluated solutions, N_{GGAsol}, is a function of the number of domains, D, the chromosome population size, the non-elite solutions (calculated by the product of $1 - E_R$ and N_P), and the number of iterations, I. For more details on the implementation of GGAs (e.g. elite solutions, elitism rate, chromosome population) we refer the reader to (Falkenauer, 1998).

$$N_{GGAsol} = (D \times (N_P + (N_P \times (1 - E_R) \times (I - 1))))$$

For the simulated annealing algorithm, the maximum number of solutions, N_{SAAsol}, is the number of domains, D, by the number of sequences, $maxit$, by the length of sequence, J. Recall that the SAA also stops if the minimum temperature has been reached - hence we refer to the *maximum* number of solutions rather than the *total*. For comparability purposes however, because the temperature is decremented only at the end of each sequence and we have a small number of sequences in the experiments below we assume the full number of solutions has been generated.

$$N_{SAAsol} = D \cdot maxit \cdot J$$

6.3 Data sets, target and auxiliary variables

Table 6.1 provides a summary by data set of the target and auxiliary variables.

Table 6.1: Summary by data set of the target and auxiliary variables

Dataset	**Target variables**	**Description**	**Auxiliary variables**	**Description**
SwissMunicipalities	Surfacebois	wood area	POPTOT	total population
	Airbat	area with buildings	Hapoly	municipality area
American Community Survey, 2015	HINCP	Household income past 12 months	BLD	Units in structure
	VALP	Property value	TEN	Tenure
	SMOCP	Selected monthly owner costs	WKEXREL	Work experience of householder and spouse
	INSP	Fire/hazard/flood insurance yearly amount	WORKSTAT	Work status of householder or spouse in family households
			HFL	House heating fuel
			YBL	When structure first built
US Census, 2000	HHINCOME	total household income	PROPINSR	annual property insurance cost
			COSTFUEL	annual home heating fuel cost
			COSTELEC	annual electricity cost
			VALUEH	House value
Kiva Loans	term_in_months	duration for which the loan was disbursed	sector	high level categories, e.g. food
	lender_count	the total number of lenders	currency	currency of the loan
	loan	the amount in USD	activity	more granular category, e.g. fruits & vegetables
			region	region name within the country
			partner_id	ID of the partner organization
UN Commodity Trade Statistics data	trade_usd	value of the trade in USD	commodity	type of commodity e.g. "Horses, live except pure-bred breeding"
			flow	whether the commodity was an import, export, re-import or re-export
			category	category of commodity, e.g. silk or fertilisers

The target and auxiliary variables for the Swiss Municipalities data set were selected based on the experiment described in Ballin and Barcaroli (2020). Accordingly, *POPTOT* and *HApoly* were converted into categorical variables using the k-means clustering algorithm. However, we used more domains and iterations in our experiment. More information on this data set is provided by Barcaroli (2014).

For the remaining experiments we selected target and auxiliary variables which we deemed likely to be of interest to survey designers. Further details on the American Community Survey, 2015 (US Census Bureau, 2016), the US Census, 2000 (Ruggles et al., 2017), Kiva Loans (Kiva, 2018), and the UN commodity trade statistics data (United Nations, 2017) metadata are available in O'Luing et al. (2019).

A further summary by data set of the number of records and atomic strata, along with a description of the domain variable, is provided in table 6.2 below.

Table 6.2: Summary by data set of the number of records and atomic strata and a description of the domain variable

Data set	**Number of records**	**Number of atomic strata, L**	**Domain variable**
Swiss Municipalities	2,896	579	REG
American Community Survey, 2015	619,747	123,007	ST (the 51 states)
US Census, 2000	627,611	517,632	REGION
Kiva Loans	614,361	84,897	country code
UN Commodity Trade Statistics data	352,078	351,916	country or area

6.4 Results

As mentioned previously, we used an initial solution in each experiment that is created by the *KmeansSolution* algorithm (Ballin and Barcaroli, 2020). We then compared the performance of the algorithms in terms of average computation time (in seconds) per solution and solution quality. Table 6.3 provides the sample size, execution times and total execution times for the SAA and GGA.

Table 6.3: Summary by data set of the sample size and evaluation time for the grouping genetic algorithm and simulated annealing algorithm

	GGA			**SAA**		
Data set	**Sample size**	**Execution time (seconds)**	**Total Execution time (seconds)**	**Sample size**	**Execution time (seconds)**	**Total Execution time (seconds)**
Swiss Municipalities	128.69	753.82	10,434.30	125.17	248.91	8,808.63
American Community Survey, 2015	10,136.50	13,146.25	182,152.46	10,279.44	517.76	6,822.42
US Census, 2000	228.81	2,367.36	36,298.35	224.75	741.75	8,996.85
Kiva Loans	6,756.19	15,669.11	288,946.79	6,646.67	664.30	7,549.87
UN Commodity Trade Statistics data	3,216.68	6,535.97	88,459.22	3,120.07	1,169.26	12,161.80

The total execution time is the sum of the execution times for 20 evaluations of the GGA and SAA algorithms (by the *MBO* (model-based optimisation) function in the R package *mlrMBO* (Bischla et al., 2017)) using 20 sets of selected hyperparameters (i.e. one set for each evaluation). Details on the precision constraints and hyperparameters for each experiment can be found in the appendix - section 10. Table 6.4 expresses the SAA results as a ratio of those for the GGA.

Table 6.4: Ratio comparison of the sample sizes, execution times, and total execution times for the grouping genetic algorithm and simulated annealing algorithm

Data set	Sample size	Execution time (seconds)	Total execution time (seconds)
Swiss Municipalities	.97	.33	.84
American Community Survey, 2015	1.01	.04	.04
US Census, 2000	.98	.31	.25
Kiva Loans	.98	.04	.03
UN Commodity Trade Statistics data	.97	.18	.14

As can be seen, the sample sizes are similar, however, the SAA shows significantly lower execution and total execution times. When these experiments are run in parallel, for cases where there is a large number of domains, there may not be enough cores to cover all domains in one run. Indeed, it may take several parallel runs to complete the task, and this will affect mean evaluation time. The computer specifications are provided in table 10.2. Table 6.5 shows the number of solutions evaluated by each algorithm to obtain the results shown in table 6.3. It also provides a ratio comparison of the average execution time (in seconds) per solution.

Table 6.5: Number of solutions and ratio comparison of execution time (per second) between the grouping genetic algorithm and simulated annealing algorithm

	Number of solutions evaluated	
Data set	**GGA**	**SAA**
Swiss Municipalities	840,140	210,000
American Community Survey, 2015	2,550,510	459,000
US Census, 2000	10,872	36,000
Kiva Loans	2,190,730	730,000
UN Commodity Trade Statistics data	2,395,026	1,539,000

	Average execution time per solution (seconds)		
Data set	**GGA**	**SAA**	**Proportion**
Swiss Municipalities	0.0009	0.0012	1.3210
American Community Survey, 2015	0.0052	0.0011	0.2188
US Census, 2000	0.2177	0.0206	0.0946
Kiva Loans	0.0072	0.0009	0.1272
UN Commodity Trade Statistics data	0.0027	0.0008	0.2784

The above results indicate that the GGA has evaluated more solutions to find a solution of similar quality to the SAA in all cases, except for the *US Census, 2000* experiment. However, we also can see that the SAA takes less time to evaluate each solution in all cases except for the *Swiss Municipalities* experiment. The average execution time for each experiment can be considered in the context of the size of the data set, parallelisation, and the particular sets of hyperparameters used for the GGA and SAA. In addition to this, there is also memoisation in the evaluation algorithm for the GGA, and the gains obtained by delta evaluation by the SAA.

Gains are more noticeable for larger data sets, because of the size of the solution and number of atomic strata in each stratum. As the strata get larger in size, the movement of q atomic strata from one stratum to another (where q is small) will have a smaller impact on solution quality and, therefore, the delta evaluation will be quicker.

7 Comparison with the continuous method in *SamplingStrata*

We also compared the SAA with the traditional genetic algorithm which Ballin and Barcaroli (2020) have applied to partition continuous strata. We used the target variables outlined in table 6.1 above as both the continuous target and auxiliary variables (for clarity we outline them again in table 7.1 below) along with the precision constraints outlined in table 10.1 (the appendix). In practice, the target variable would not be exactly equal to the auxiliary variable though it is common for the auxiliary variable to be an imperfect version (for example an out-of-date or a related variable) available on the sampling frame. We invite the reader to consider this when reviewing the results of the comparisons below. It is also worth noting that initial solutions were created for both algorithms using the k-means method. Details on the training of hyperparameters for these experiments also can be found in the appendix - section 10.

Table 7.1: Summary by data set of the target and auxiliary variable descriptions for the continuous method

Dataset	**Target variables**	**Auxiliary variables**	**Description**
SwissMunicipalities	Surfacebois	Surfacebois	wood area
	Airbat	Airbat	area with buildings
American Community Survey, 2015	HINCP	HINCP	Household income (past 12 months)
	VALP	VALP	Property value
	SMOCP	SMOCP	Selected monthly owner costs
	INSP	INSP	Fire/hazard/flood insurance (yearly amount)
US Census, 2000	HHINCOME	HHINCOME	total household income
Kiva Loans	term_in_months	term_in_months	duration for which the loan was disbursed
	lender_count	lender_count	the total number of lenders
	loan	loan	the amount in USD
UN Commodity Trade Statistics data	trade_usd	trade_usd	value of the trade in USD

The attained sample sizes are compared in table 7.2 below where the sample size for the SAA is expressed as a ratio of the TGA. After the hyperparameters were fine-tuned (see section 10.6) the resulting sample sizes are comparable.

Table 7.2: Ratio comparison of the sample sizes for the traditional genetic algorithm and simulated annealing algorithm on the continuous method

Data set	TGA	SAA	Ratio
Swiss Municipalities	128.69	120.00	0.93
American Community Survey, 2015	4,197.68	3,915.48	0.93
US Census, 2000	192.71	179.89	0.93
Kiva Loans	3,062.33	3,017.79	0.99
UN Commodity Trade Statistics data	3,619.42	3,258.52	0.90

Table 7.3 compares the execution times for the set of hyperparameters that found the sample sizes for each algorithm in table 7.2 above, as well as the total execution times taken to train that set of hyperparameters.

Table 7.3: Ratio comparison of the execution times and total execution times for the traditional genetic algorithm and simulated annealing algorithm on the continuous method

	TGA		SAA		Ratio comparison	
Data set	Execution time (seconds)	Total execution time (seconds)	Execution time (seconds)	Total execution time (seconds)	Execution time (seconds)	Total execution time (seconds)
Swiss Municipalities	753.82	10,434.30	213.44	1,905.82	.28	.18
American Community Survey, 2015	22,016.95	227,635.51	13,351.19	169,115.92	.61	.74
US Census, 2000	3,361.90	46,801.78	51.94	1,147.36	.02	.02
Kiva Loans	3,232.78	48,746.61	300.16	4,149.06	.09	.09
UN Commodity Trade Statistics data	29,045.23	326,931.63	73.18	1,287.38	.003	.004

These results indicate a significantly lower execution time for the SAA for the attained solution quality. The computational efficiency gained by delta evaluation in the training of the recommended hyperparameters is also evident in the total execution times. For the *American Community Survey, 2015* experiment significantly more solutions were generated by the SAA than the TGA as a result of the given hyperparameters and this impacts the execution and total execution times (see also table 7.4). Table 7.4 compares the number of solutions generated by the traditional genetic algorithm with the simulated annealing algorithm.

Table 7.4: Comparison of the number of solutions generated by the traditional genetic algorithm and simulated annealing algorithm on the continuous method

	Number of solutions evaluated	
Data set	**TGA**	**SAA**
Swiss Municipalities	840,140	175,000
American Community Survey, 2015	918,102	5,100,000
US Census, 2000	43,272	18,000
Kiva Loans	146,730	292,000
UN Commodity Trade Statistics data	20,521,026	85,500

In all cases except for *Kiva Loans* and the *American Community Survey, 2015* the SAA has generated fewer solutions. The low number of solutions generated by both algorithms for the *US Census, 2000* experiment may indicate that the initial k-means solution was near the global minimum. The *American Community Survey, 2015* results indicate that the SAA generated significantly more solutions to get to a comparable sample size with the TGA. As we are moving, predominantly, $q = 1$ atomic strata between strata such changes in this case had limited impact on solution quality from one solution to the next. However, the gains achieved by delta evaluation meant that more solutions were evaluated per second leading to a more complete search and a lower sample size being attained.

For these experiments, the TGA took longer to find a comparable sample size in all cases. As pointed out in O'Luing et al. (2019), traditional genetic algorithms are not as efficient for grouping problems as the grouping genetic algorithm because solutions tend to have a great deal of redundancy. We would, therefore, propose that the GGA be applied also to continuous strata. On the basis of the above analysis, and the performance of SAAs in local search generally speaking along with the added gains in efficiency from delta evaluation, we would also propose that the SAA be considered as an alternative to the traditional genetic algorithm.

8 Conclusions

We compared the SAA with the GGA in the case of atomic strata and the TGA in the case of continuous strata (Ballin and Barcaroli, 2020). The k-means algorithm provided good starting points in all cases. When the hyperparameters have been fine-tuned all algorithms attain results of similar quality.

However, the execution times for the recommended hyperparameters are lower for the SAA than for the GGA with respect to atomic strata and traditional genetic algorithm with respect to continuous strata. Delta evaluation also has advantages in reducing the training times needed to find the suitable hyperparameters for the SAA.

The GGA might benefit from being extended into a memetic algorithm by using local search to quickly improve a chromosome before adding it to the GGA chromosome population.

The SAA, by using local search (along with a probabilistic acceptance of inferior solutions), is well suited to navigation out of local minima and the implementation of delta evaluation enables a more complete search of the local neighbourhood than would otherwise be possible in the same computation time.

9 Further work

The perturbation used by the SAA randomly moves q atomic strata, where mainly $q = 1$, from one stratum to another. This stochastic process is standard in default simulated annealing algorithms. However, as we are using a starting solution where there is already similarity within the strata, this random process could easily move an atomic stratum ($q = 1$) to a stratum where it is less suited than the stratum it was in. This suggests the presence of a certain amount of redundancy in the search for the global minimum.

Lisic et al. (2018) conjecture that the introduction of nonuniform weighting in atomic strata selection could greatly improve performance of (their proposed) simulated annealing method by exchanging atomic strata near stratum boundaries more frequently than more important atomic strata. We agree that, for this algorithm, it would be more beneficial if there was a higher probability that an atomic stratum which was dissimilar to the other atomic strata was selected. We could then search for a more suitable stratum to move this atomic stratum to.

To achieve this we could first randomly select a stratum, and then measure the Euclidean distance of each atomic stratum from that stratum medoid, weighting the chance of selection of the atomic strata in accordance with their distance from the medoid. At this point, an atomic stratum is selected using these weighted probabilities.

The next step would be to use a K-nearest-neighbour algorithm to find the stratum medoid closest to that atomic stratum and move it to that stratum. This simple machine learning algorithm uses distance measures to classify objects based on their K nearest neighbours. In this case, $k = 1$, so the algorithm in practice is a closest nearest neighbour classifier.

This additional degree of complexity to the algorithm may offset the gains achieved by using delta evaluation, particularly as the problem grows in size, thus reducing the number of solutions evaluated in the same running time. It might be more effective to use the column medians as an equivalent to the medoids. This could assist the algorithm find better quality solutions.

However, the above suggestions may only be effective at an advanced stage of the search, where the atomic strata in each stratum are already quite similar.

Acknowledgements

We wish to acknowledge the editorial staff and reviewers of Survey Methodology for their constructive suggestions in the review process for this journal submission, in particular the suggestion to compare the SAA with the traditional genetic algorithm in Ballin and Barcaroli (2020) in the case of continuous strata. This material is based upon work supported by the Insight Centre for Data Analytics and Science Foundation Ireland under

Grant No. 12/RC/2289-P2 which is co-funded under the European Regional Development Fund. Also, this publication has emanated from research supported in part by a grant from Science Foundation Ireland under Grant number 16/RC/3918 which is co-funded under the European Regional Development Fund.

Bibliography

Ballin, M. and G. Barcaroli (2013). Joint determination of optimal stratification and sample allocation using genetic algorithm. *Survey Methodology 39*(2), 369–393.

Ballin, M. and G. Barcaroli (2020). Optimization of sampling strata with the SamplingStrata package. Accessed 3 May 2021. https://barcaroli.github.io/SamplingStrata/articles/SamplingStrata.html

Barcaroli, G. (2014). SamplingStrata: An R package for the optimization of stratified sampling. *Journal of Statistical Software 61*(4), 1–24.

Benedetti, R., G. Espa, and G. Lafratta (2008). A tree-based approach to forming strata in multipurpose business surveys. *Survey Methodology 34*(2), 195–203.

Bethel, J. W. (1985). *An optimum allocation algorithm for multivariate surveys.* American Statistical Proceedings of the Survey Research Methods, Section, 209-212.

Bethel, J. (1989). Sample allocation in multivariate surveys. *Survey methodology 15*(1), 47-57.

Bischla, B., J. Richterb, J. Bossekc, D. Hornb, J. Thomasa, and M. Langb (2017). mlrmbo: A modular framework for model-based optimization of expensive black-box functions. *Gradient Boosting in Automatic Machine Learning: Feature Selection and Hyperparameter Optimization*, 36.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications 45*(1), 41–51.

Chromy, J. R. (1987). Design optimization with multiple objectives. *Proceedings of the Survey Research Methods Section.*

Cochran, W. G. et al. (1977). Sampling techniques. *Canada: John Willey & Sons Inc.*

Cortez, P. (2014). *Modern optimization with R.* Springer.

Falkenauer, E. (1998). *Genetic algorithms and grouping problems.* John Wiley & Sons, Inc.

Grover, L. K. (1987). Standard cell placement using simulated sintering. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pp. 56–59.

Hartigan, J. A. and M. A. Wong (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*(1), 100–108.

Hoos, H. H. and T. Stützle (2004). *Stochastic local search: Foundations and applications.* Elsevier.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2010). Sequential model-based optimization for general algorithm configuration (extended version). *Technical Report TR-2010–10, University of British Columbia, Computer Science, Tech. Rep.*.

Johnson, D. S., C. R. Aragon, L. A. McGeoch, and C. Schevon (1989). Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research 37*(6), 865–892.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *science 220*(4598), 671–680.

Kiva (2018, Mar). Data science for good: Kiva crowdfunding. https://www.kaggle.com/kiva/data-science-for-good-kiva-crowdfunding

Korf, Richard E (1999). *Artificial intelligence search algorithms.* Citeseer.

Lisic, J., H. Sang, Z. Zhu, and S. Zimmer (2018). Optimal stratification and allocation for the june agricultural survey. *Journal of Official Statistics 34*(1), 121–148.

Luke, S. (2013). *Essentials of Metaheuristics* (second ed.). Lulu. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

McKay, M. D., R. J. Beckman, and W. J. Conover (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 42*(1), 55–61.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics 21*(6), 1087–1092.

Microsoft Corporation and S. Weston (2020a). *foreach: Provides Foreach Looping Construct.* R package version 1.5.1.

Microsoft Corporation and S. Weston (2020b). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package.* R package version 1.0.16.

O'Luing, M., S. Prestwich, and S. Armagan Tarim (2019). A grouping genetic algorithm for joint stratification and sample allocation designs. *Survey Methodology 45*(3), 513–531.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/

Rose, J. S., W. M. Snelgrove, and Z. G. Vranesic (1988). Parallel standard cell placement algorithms with quality equivalent to simulated annealing. *IEEE transactions on computer-aided design of integrated circuits and systems 7*(3), 387–396.

Ross, P., D. Corne, and H.-L. Fang (1994). Improving evolutionary timetabling with delta evaluation and directed mutation. In *International Conference on Parallel Problem Solving from Nature*, pp. 556–565. Springer.

Ruggles, S., K. Genadek, R. Goeken, J. Grover, and M. Sobek (2017). Integrated public use microdata series: Version 7.0 [dataset]. minneapolis: University of minnesota, 2017.

Sörensen, K. and F. Glover (2013). Metaheuristics. *Encyclopedia of operations research and management science 62*, 960–970.

Takeang, C. and A. Aurasopon (2019). Multiple of hybrid lambda iteration and simulated annealing algorithm to solve economic dispatch problem with ramp rate limit and prohibited operating zones. *Journal of Electrical Engineering & Technology 14*(1), 111–120.

United Nations (2017, Nov). Global commodity trade statistics. https://www.kaggle.com/unitednations/global-commodity-trade-statistics

US Census Bureau (2016). *2015 ACS Public Use Microdata Sample (PUMS).* Washington, D.C. https://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t#.

Van Laarhoven, P. J. and E. H. Aarts (1987). Simulated annealing. In *Simulated annealing: Theory and applications*, Springer.

10 Appendix - Background details on the comparisons in sections 6 and 7

10.1 Precision constraints

The target upper precision levels for these experiments, i.e. coefficients of variation, for each of the five experiments are provided in table 10.1 below.

Table 10.1: Summary by data set of the upper limits for the coefficients of variation

Data set	**CV**
Swiss Municipalities	0.1
American Community Survey, 2015	0.05
US Census, 2000	0.05
Kiva Loans	0.05
UN Commodity Trade Statistics data	0.05

We selected an upper precision level of 0.1 for the *Swiss Municipalities* data set in keeping with the level set for the experiment in Ballin and Barcaroli (2020). We used an upper precision level of 0.05 for the remaining experiments, given that the upper CV levels generally set by national statistics institutes (NSIs) tend to be between 0.01 and 0.1, and, for this reason, results for CVs in the mid-point of this range are of interest.

10.2 Processing platform

Table 10.2 below provides details of the processing platform used for these experiments.

Table 10.2: Specifications of the processing platform

Specification	**Details**	**Notes**
Processor	AMD Ryzen 9 3950X 16-Core Processor, 3493 Mhz	
Cores	16 Core(s)	
Logical processors	32 Logical Processor(s)	32 cores in R
System model	X570 GAMING X	
System type	x64-based PC	
Installed physical memory (RAM)	16.0 GB	
Total virtual memory	35.7 GB	
OS name	Microsoft Windows 10 Pro	

In all cases, R version 4.0 or greater was used. We used the *foreach* (Microsoft and Weston, 2020a) and *doParallel* (Microsoft and Weston, 2020b) packages to run the experiments in parallel. The number of cores used in the experiments was 31 (32 less 1) and this means that in the three experiments with more than 31 domains (*American Community Survey 2015, Kiva Loans, UN Commodity Trade Statistics data*) the *foreach* algorithm continued to loop through the available cores until a solution had been found for all domains.

10.3 Hyperparameters for the grouping genetic algorithm and simulated annealing algorithm

Tables 10.3 and 10.4 below outline the number of domains in each experiment, along with number of iterations and chromosome population size for the grouping genetic algorithm and along with the number of sequences, length of sequence, and starting temperature for the simulated annealing algorithm. Section 10.4 provides details on fine-tuning the hyperparameters. For more details on the hyperparameters of the GGA we refer the reader to Ballin and Barcaroli (2013) and O'Luing et al. (2019) and of the SAA to sections 2.2 and 4.

Table 10.3: Summary by data set of the hyperparameters for the grouping genetic algorithm for each domain

Data set	**Domains**	**Number of iterations, I**	**Chromosome population size, N_p**	**Mutation chance**	**Elitism rate, E_R**	**Add strata factor**
Swiss Municipalities	7	4,000	50	0.0053360	0.4	0.0037620
American Community Survey, 2015	51	5,000	20	0.0008134	0.5	0.0610529
US Census, 2000	9	100	20	0.0000007	0.4	0.0000472
Kiva Loans	73	3,000	20	0.0007221	0.5	0.0685005
UN Commodity Trade Statistics data	171	1,000	20	0.0004493	0.3	0.0866266

Table 10.4: Summary by data set of the hyperparameters for the simulated annealing algorithm for each domain

Data set	Domains	Number of sequences, $maxit$	Length of sequence, J	Temperature, T	Decrement constant, DC	% of L for maximum q value, $L_{max\%}$	Probability of new stratum, $P(H+1)$
Swiss Municipalities	7	10	3,000	0.0000720	0.5083686	0.0183356	0.0997907
American Community Survey, 2015	51	3	3,000	0.0002347	0.6873029	0.0076477	0.0291729
US Census, 2000	9	2	2,000	0.0006706	0.5457192	0.0189395	0.0806919
Kiva Loans	73	5	2,000	0.0009935	0.7806557	0.0143925	0.0317491
UN Commodity Trade Statistics data	171	3	3,000	0.0007902	0.5072737	0.0234728	0.0013775

10.4 Fine-tuning the hyperparameters for the grouping genetic algorithm and simulated annealing algorithm

In order to fine-tune the initial parameters or *hyperparameters* we used sequential model-based optimization (Hutter et al., 2010). We first generated an initial design of hyperparameters from the value ranges described for the GGA in table 10.5 and in table 10.6 for the SAA below using the latin hypercube design method (McKay et al., 2000).

Table 10.5: Ranges for fine-tuning the hyperparameters for the grouping genetic algorithm

	Iterations			Population size		
Value type	Discrete			Discrete		
Value range	Lower value	Upper value	Increments	Lower value	Upper value	Increments
Swiss Municipalities	500	5,000	500	10	50	10
American Community Survey, 2015	1,000	5,000	1,000	10	20	10
Kiva Loans	1,000	3,000	1,000	10	20	10
UN Commodity Trade Statistics data	500	1,000	500	10	20	10
US Census, 2000	50	100	50	10	20	10

	Mutation chance		Elitism rate, E_R			Add strata factor	
Value type	Numeric		Discrete			Numeric	
Value range	Lower value	Upper value	Lower value	Upper value	Increments	Lower value	Upper value
Swiss Municipalities	0	0.10	0.1	0.5	0.1	0	0.1
American Community Survey, 2015	0	0.001	0.1	0.5	0.1	0	0.1
Kiva Loans	0	0.001	0.1	0.5	0.1	0	0.1
UN Commodity Trade Statistics data	0	0.001	0.1	0.5	0.1	0	0.1
US Census, 2000	0	0.000001	0.1	0.5	0.1	0	0.0001

Table 10.6: Ranges for fine-tuning the hyperparameters for the simulated annealing algorithm

	Number of sequences, maxit			Length of sequence, J		
Value type	Discrete			Discrete		
Value range	Lower value	Upper value	Increments	Lower value	Upper value	Increments
Swiss Municipalities	10	50	10	1,000	3,000	1,000
American Community Survey, 2015	1	3	1	1,000	3,000	1,000
Kiva Loans	1	5	1	1,000	2,000	1,000
UN Commodity Trade Statistics data	1	3	1	1,000	3,000	1,000
US Census, 2000	1	2	1	1,000	2,000	1,000

	Temperature, T		Decrement constant, DC		% L for maximum q value, $L_{max\%}$		Probability of new stratum, P(H+1)	
Value type	Numeric		Numeric		Numeric		Numeric	
Value Range	Lower value	Upper value	Lower value	Upper value	Lower value	Upper value	Lower value	Upper value
Swiss Municipalities	0	0.001	0.5	1	0.0001	0.025	0	0.1
American Community Survey, 2015	0	0.001	0.5	1	0.0001	0.025	0	0.1
Kiva Loans	0	0.001	0.5	1	0.0001	0.025	0	0.1
UN Commodity Trade Statistics data	0	0.001	0.5	1	0.0001	0.025	0	0.1
US Census, 2000	0	0.001	0.5	1	0.0001	0.025	0	0.1

As some of the hyperparameter value ranges were discrete, we used a random forest with regression trees to develop a surrogate learner model. After this, a confidence bound using a lambda value, λ, to control the trade-off between exploitation and exploration was used as the acquisition function. The focus search approach (Bischla et al., 2017) was used to optimise the acquisition function which, in turn, was used to propose the hyperparameters which were evaluated using the surrogate function (which is a cheaper alternative to using the GGA or SAA algorithms). From these, the most promising hyperparameters were then evaluated by the GGA or SAA and the hyperparameters and solution costs added to the initial design. The process was then repeated for a set number of iterations and the best performing hyperparameters and solution outcomes were selected. We implemented this using the *MBO* function with the parameters outlined in table 10.7. These are distinct from the parameters being fine-tuned, which are outlined in tables 10.5 and 10.6

above.

Table 10.7: Parameters used in the* MBO *Function

MBO parameters	**Value**
Initial Design size (Latin Hypercube Design method)	10
Iterations, number of	10
Number of Trees	500
Lambda, λ	5
Focus Search Points	$1,000$

As can be seen from the limited scope of the *MBO* function parameters this was not an exhaustive fine-tuning of the hyperparameters for the GGA and SAA. The aim of these experiments was to consider whether the SAA can attain comparable solution quality with the GGA in less computation time per solution thus resulting in savings in execution times. However, we also compared the total execution times as this is a consequence of the need to train the hyperparameters for both algorithms.

Tables outlining the hyperparameters, in each of the 20 fine-tuning iterations, for each experiment are available from the authors on request. The first 10 sets of hyperparameters were randomly generated from the ranges laid out in tables 10.5 and 10.6. The ranges selected were identified using practical knowledge of the algorithms and data. The second 10 sets reflects the *MBO* function's attempts to learn the hyperparameters that best lead each algorithm towards the optimal solution using the previous solutions as a guide.

10.5 Hyperparameters for the traditional genetic algorithm and simulated annealing algorithm

Tables 10.8 and 10.9 outline the hyperparameters for the tradtional genetic algorithm and the simulated annealing algorithm. The add strata factor option is not available for the traditional genetic algorithm and, therefore, is not included in table 10.8. More details on fine-tuning the hyperparameters are provided in section 10.6.

Table 10.8: Hyperparameters for the traditional genetic algorithm

Data set	Iterations	Population size	Mutation chance	Elitism rate, E_R
Swiss Municipalities	4,000	50	0.0053360	0.4
American Community Survey, 2015	1,000	20	0.0009952	0.1
US Census, 2000	400	20	0.0002317	0.4
Kiva Loans	200	20	0.0817285	0.5
UN Commodity Trade Statistics data	5,000	30	0.0005599	0.2

Table 10.9: Hyperparameters for the simulated annealing algorithm

Data set	Number of sequences, maxit	Length of sequence, J	Temperature, T	Decrement constant , DC	% for maximum q value, $L_{max\%}$	Probability of new stratum, P(H+1)
Swiss Municipalities	5	5,000	0.02311057	0.9427609	0.3736443	0.0229361
American Community Survey, 2015	50	2,000	0.00000005	0.9528952	0.0001021	0.0000008
US Census, 2000	1	2,000	0.00002000	0.9665631	0.0221147	0.0160408
Kiva Loans	2	2,000	0.00053839	0.8660943	0.0014281	0.0216320
UN Commodity Trade Statistics data	2	250	0.00067481	0.9309940	0.0203113	0.0149499

10.6 Fine-tuning the hyperparameters for the traditional genetic algorithm and simulated annealing algorithm

We fine-tuned the hyperparameters for the TGA and SAA using the same methodology described in section 10.4. Tables outlining the hyperparameters, in each of the 20 fine-tuning iterations, for each experiment are available from the authors on request. The first 10 sets were randomly generated using practical knowledge of the algorithms and data to define upper and lower bounds for each hyperparameter. In the second 10 sets the *MBO* function attempts to optimise the hyperparameters using the previous solutions as a guide.