Simulated Annealing for Structure Learning
Statistical Genomics - Final Project

Andrei-Rafael Secuiu

December 15, 2025

1 Introduction

At a first glance, the problem of learning a Bayesian Network (BN) from statistical data is simple: given a data set D, we wish to search for the BN which "best describes" the relationships among variables. However, practical difficulties make this problem challenging. A BN may be viewed as a multivariate probability distribution $(X_1.X_2, \ldots, X_n)$ in which (conditional) independencies among the random variables are graphically represented via a Directed Acyclic Graph (DAG). In particular, it can be shown that for a BN with DAG G, the joint probability distribution can be factorized as below:

$$\mathbb{P}(X_1, \ldots, X_n|G) = \prod_{i=1}^{n} \mathbb{P}(X_i|\text{pa}(X_i))$$

where $\text{pa}(X_i)$ denotes the set of parents for the node corresponding to X_i, according to the DAG G.

Learning a BN is a two-fold task: (1) finding the optimal DAG G and (2) describing the conditional probability distributions $X_i|\text{pa}(X_i)$ given G. When G is provided, the second task may be approached via traditional statistical methods such as Maximum Likelihood (ML) estimation. Henceforth, the main issue lies in the learning of G, also known as *structure learning*. A great difficulty associated with structure learning is the large number of DAGs to be investigated, which is super-exponentially increasing with the number of nodes n. As such, structure learning is an NP-hard problem (Lee and Kim, 2020). This lead to the development of a variety of heuristic search algorithms; a brief overview can be found in the introduction of Adabor et al., 2015.

The iconic distinction between Bayesian and frequentist methods is also reflected in structure learning. Frequentist algorithms aim to find the best DAG G according to a scoring metric, which reduce the problem to one of combinatorial optimization. In contrast, one could consider the scores of all (explored) DAGs and rank the importance of each edge present via Bayesian model averaging. Throughout our lectures, we have encountered two algorithms which reflect these philosophies well: greedy search via Gradient Ascent (GA) (also known as hill-climbing) and structure Markov Chain Monte Carlo (strMCMC). GA is a deterministic algorithm which greedily chooses the neighbour graph that improves the scoring function the most. It is known that GA is prone to finding local maxima, hence an improvement is to repeat it with randomized starting points (Adabor et al., 2015). strMCMC is a stochastic algorithm which generates a sample of DAGs from their posterior distribution via the Metropolis-Hastings scheme; DAGs with good scores are more likely to be sampled. After a sufficiently large MCMC sample is generated, the importance of each possible edge is then determined via the marginal edge posterior. Simulated Annealing (SA) is a technique that lies in between these two extremes. Its aim is frequentist, i.e. finding the DAG that optimizes the score function. However, the method by which the space of DAGs is explored is very reminiscent of strMCMC. "Changes that increase and decrease the score are allowed. Changes that decrease the score become less probable as the search continues" (Hammond and Smith, 2025). Figure 1 shows a graphical illustration of how these three optimization strategies explore the space of DAGs.

Project objectives

The aim of this project is to provide an introductory description of SA, in which its background, mathematical formulation and algorithmic implementation are provided. Furthermore, SA is compared to GA

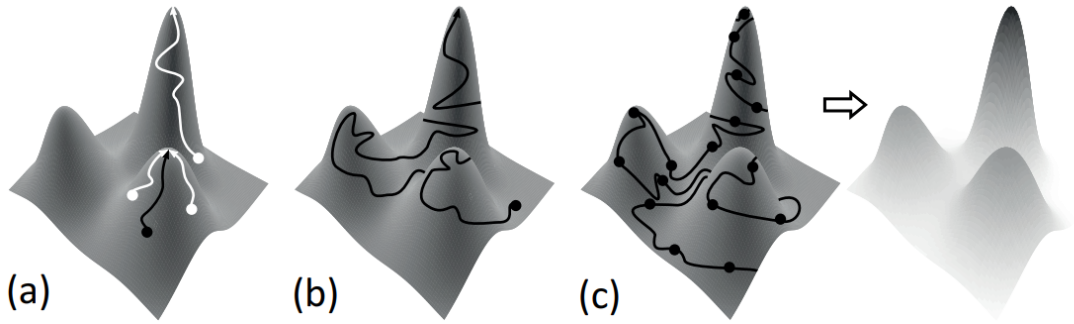


Figure 1: Illustration of the optimization strategies behind (a) GA, (b) SA and (c) strMCMC. Taken from Hammond and Smith, 2025

and strMCMC through a simulation study whose methodology resembles Assignment 3. Through the simulation study, we aim to:

- assess SA's performance relative to GA/strMCMC and determine when its use is justified
- check if different choices for the temperature schedule have a significant impact on SA's performance

2 Background, mathematical formulation and intuition

Background. SA has been first proposed by Kirkpatrick et al., 1983, with the concept behind SA being greatly inspired by the field of statistical mechanics. The word "annealing" comes from an analogy with the real-world annealing procedure done when studying the low-temperature properties of materials. "Experiments that determine the low-temperature state of a material - for example, by growing a single crystal from a melt - are done by careful annealing, first melting the substance, then lowering the temperature slowly, and spending a long time at temperatures in the vicinity of the freezing point. [If that is not done, the substance may settle in locally optimal, meta-stable structures]" (Kirkpatrick et al., 1983).

Theoretical guarantees for the convergence of SA have been given, one of which is by Granville et al., 1994. Their proof, however, is rather technical; in the words of the authors: "the fact that the algorithm will converge is much less intuitive and the proof requires some extra computations based on a lemma of Perron and Frobenius as well as the analysis of the eigenvalues of a stochastic matrix". Therefore, we shall settle for a brief outline of the convergence result, as well as for an intuitive explanation of SA.

What does SA address? The frequentist approach to the learning of BN often aims to find the "best fitting" DAG (regularization of parameters included) according to some metric, which makes structure learning a combinatorial optimization problem. Greedy algorithms, such as GA, are prone to getting stuck into local optima.

To address this issue, one of the simplest approaches is to repeat the procedure multiple times, for randomized starting points, with the hope that one of these starting points will be in a better basin of attraction and leading to a better optimum. It is mentioned in Koller and Friedman, 2009 that this approach is reasonable, but with one possible limitation: moving from one wide hill to another is unlikely. The choice of how many restarts should be used thus becomes tricky. If the number is too small, then escaping the local maxima is unlikely; if the number is too large, the search may stray too far from the region of high-scoring networks (leading to wasted computational resources). This issue is studied in more detail in the Appendix B.

SA provides an alternative framework, conceptually similar to the Metropolis-Hastings algorithm. "In broad outline, the simulated annealing procedure attempts to mix hill-climbing steps with moves that can decrease the score. This mixture is controlled by a so-called *temperature parameter*. When the temperature is 'hot', the search tries many moves that decrease the score. As the search is annealed (the temperature is slowly reduced) it starts to focus only on moves that improve the score. The intuition is

that during the 'hot' phase the search explores the space and eventually gets trapped in a region of high scores. As the temperature reduces it is able to distinguish between finer details of the score 'landscape' and eventually converge to a good maximum" (Koller and Friedman, 2009).

Mathematical formulation. In what follows, the information has been adapted from Janžura and Nielsen, 2006; personal insight has been added to provide a clearer picture that is closer to the course material. Let us denote by S the space of all DAGs with n nodes, and by $f : S \to \mathbb{R}$ the function which assigns the score of the DAG given a data set. The frequentist problem of structure learning can be formulated in the following manner:

$$\text{Find} \quad S^* = \underset{g \in S}{\operatorname{argmax}}\ f(g)$$

For a fixed temperature parameter $T > 0$, let us define

$$P_T(g) = \frac{\exp\left(\frac{f(g)}{T}\right)}{\sum_{g' \in S} \exp\left(\frac{f(g')}{T}\right)}$$

where the denominator is a normalization constant[1]. In the limit $T \to 0$, the terms that belong to the maximizers of f dominate the others; with a bit of calculus, one can show that the function P_T converges to

$$P_0(g) = \begin{cases} |S^*|^{-1} & g \in S^* \\ 0 & \text{otherwise} \end{cases}$$

In other words, if we can sample from P_0, then the maximization problem is solved. However, the manner by which this is achieved in not immediate. It is not feasible to evaluate P_T directly for any T [2]. Hence, a method that is able to sample from an arbitrarily complex distribution which does not require the knowledge of the normalization constant, is needed. An immediate candidate is the Metropolis-Hastings algorithm.

The sampling from P_0 can be viewed as a compound approximation process. Firstly, we approximate P_0 by P_T for $T \approx 0$. Secondly, through the Metropolis-Hastings algorithm, P_T is approximated as the (almost) stationary distribution of an ergotic Markov Chain. Let us describe how the two processes are combined.

Consider an ergotic Markov Chain on S whose transition matrix (kernel) is denoted by M_T. If one views P_T as a row vector and $P_T M_T = P_T$ holds, then P_T is its stationary distribution; the Metropolis-Hastings algorithm constructs M_T via the equation of detailed balance such that $P_T M_T = P_T$. If Q is a suitable proposal scheme and

$$A(g, g') = \min\left(1, \frac{P_T(g')}{P_T(g)} \cdot \frac{Q(g', g)}{Q(g, g')}\right)$$

is the acceptance probability, then $M_T(g, g') = Q(g, g')A(g, g')$ for $g \neq g'$ defines a suitable transition kernel, and the Markov Chain has P_T as its stationary distribution. It is important to mention that when starting with some arbitrary DAG g, the first samples do not necessarily belong to P_T; the Markov Chain needs time to converge to the stationary distribution[3], even when M_T is constructed as above.

SA employs the Metropolis-Hastings algorithm, while also *simultaneously decreasing* the temperature T at each step. For a SA run with N iterations, let us define the *temperature schedule* $(T)_N = (T_1, T_2, T_3, \ldots, T_N)$. If π_{start} is some arbitrary starting distribution on the DAGs, then the distribution of the Markov Chain after N steps is:

$$\pi_N = \pi_{\text{start}} M_{T_1} M_{T_2} \ldots M_{T_N}$$

[1] If one views the problem through the lens of statistical mechanics, the denominator is the famed *partition function*
[2] After all, if we were able to compute it, then we would simply evaluate $f(g')$ for all $g' \in S$ and find the maximizers
[3] i.e. the Markov Chain needs time to get past the burn-in period

(Granville et al., 1994) *Theorem* (convergence). There exists some constant C such that if the temperature schedule $(T)_N$ obeys

$$T_i = \frac{C}{\lfloor \log(i) \rfloor} \tag{1}$$

then π_N converges almost surely to P_0 as $N \to \infty$, regardless of π_{start}.

Intuition. Relying on the same mathematics, one can assign an alternative Bayesian interpretation to SA. If the scoring function f is the log-BGe and we assign a uniform prior on all DAGs in S, then P_1 is the *posterior distribution* of the DAGs. For $T > 1$, the posterior likelihood surface is "flattened" and the Markov Chain is more likely to go from one peak to the other; in the extreme case of $T = \infty$, the posterior is completely flat and SA reduces to a random walk through S. Conversely, when $T < 1$, the peaks are amplified; in the extreme case of $T = 0$, SA accepts only moves that improve the BGe score $\Rightarrow$ SA behaves like GA. SA is a unified framework that allows to continuously change behaviour, from a random walk to Bayesian posterior sampling to hill-climbing.

About the convergence speed. It is necessary that the temperature schedule (T) does not decrease too quickly. There is a delicate balance to be struck, between decreasing the temperature such that P_0 is approximated, while simultaneously ensuring that the Markov Chain gets past the burn-in period. It is an incredibly nice result that for the temperature schedule given by Eq. 1, there is a theoretical guarantee that convergence will hold. However, the schedule in Eq. 1 is too slow to be practically useful and de-facto unknown because of the constant C. The theorem should thus be seen as a proof that SA is conceptually sound (provided that (T) decays sufficiently slowly), and not as a recipe on how to implement SA in practice.

Applications of SA rely on other choices of temperature schedules which decay to 0 more quickly. The theoretical guarantee of convergence is lost, meaning that the output of a finite SA run is not a global optimum. According to Koller and Friedman, 2009, the success of SA depends heavily on the design of the proposal distribution and annealing schedule; "tuning of this temperature parameter is required and can be a complex process" (Hammond and Smith, 2025). In some sense, the idea of having to restart SA multiple times and compare the optimality of the outputs, just like for GA, has returned.

Another quirk of SA is that after some point, more iterations (i.e. a higher N) do not lead to improved results. When the temperature is sufficiently small and SA behaves like GA, the run eventually settles into a (local) optimum; continuing the SA run is equivalent to continuously sampling the same (local) optimum. Henceforth, it is possible to speed up SA runs by investigating the temperatures at which changes in the score are extremely infrequent, and stopping the run when the schedule (T) reaches those temperatures.

Algorithm. The pseudo-code that describes the SA procedure is given in Algorithm 1. Due to its similarity to strMCMC, the R implementations are essentially identical; the modifications are the introduction of the temperature parameter, and the repeat of SA *restarts* times. At the i-th iteration, the log-acceptance probability is

$$\begin{aligned}\log A(g, g') &= \min\left(0, \log\left(\frac{P_T(g')}{P_T(g)}\right) + \log\left(\frac{Q(g', g)}{Q(g, g')}\right)\right) = \min\left(0, \log\left(\frac{\exp\left(f(g')/T_i\right)}{\exp\left(f(g)/T_i\right)}\right) + \log\left(\text{HR}(g, g')\right)\right) \\ &= \min\left(0, \frac{f(g') - f(g)}{T_i} + \log\left(\text{HR}(g, g')\right)\right)\end{aligned}$$

where f is the log-BGe score and $\text{HR}(g, g')$ is the Hastings ratio. For this project, we rely on the same proposal scheme as strMCMC, discussed in the lectures. While it is possible to start SA from any DAG, we choose to always begin with the empty graph.

Improvements of SA. Since SA was first applied to structure learning, there have been at least a couple of improvements. One was proposed by Adabor et al., 2015, in which the authors first perform a SA run, after which the output is used as a starting point for a GA run $\Rightarrow$ develop SAGA. The logic behind SAGA is that after a certain point, SA rarely accepts new changes that decrease the score; when the

Algorithm 1 Simulated Annealing for a temperature schedule $(T)_N$, with randomized restarts

Require: *data*, N, $(T)_N$, *restarts*
  for $i \leftarrow 1$; $i \leq$ restarts; $i \leftarrow i+1$ **do** ▷ Repeat SA learning *restarts* times
    Initialize starting DAG: $g \leftarrow$ *empty_DAG*
    Compute the score $f(g)$
    for $j \leftarrow 2$; $j \leq N$; $j \leftarrow j+1$ **do** ▷ An individual SA run
      Sample a new g' according to $Q(g, .)$
      Compute $f(g')$ and $\mathrm{HR}(g, g')$
      Evaluate $A(g, g') = \min\left(0, \frac{f(g')-f(g)}{T_j} + \log\left(\mathrm{HR}(g, g')\right)\right)$
      Sample $u \sim \mathrm{Unif}(0, 1)$
      if $u < A(g, g')$ **then**
        $g \leftarrow g' \Rightarrow$ accept the proposal
      else
        Leave g unmodified
      end if
    end for
    Store g and $f(g)$ from the end of the run
  end for
    return g with the highest overall score

temperature drops sufficiently, it is reasonable to continue with GA directly. Another improvement is due to Lee and Kim, 2020; the authors take SAGA and further develop PSAGA, a related technique which uses multiple parallel runs and memoization to improve the sampling efficiency. Due to the restricted scope of this report, we shall not expand on SAGA and PSAGA further.

3 Simulation study

In order to investigate the ability of SA to learn meaningful relationships among variables in a data set, we perform a simulation study. Figure 2 shows the DAG according to which the data has been simulated, and which must be learned. It is realistic to what one may encounter in a biological application.

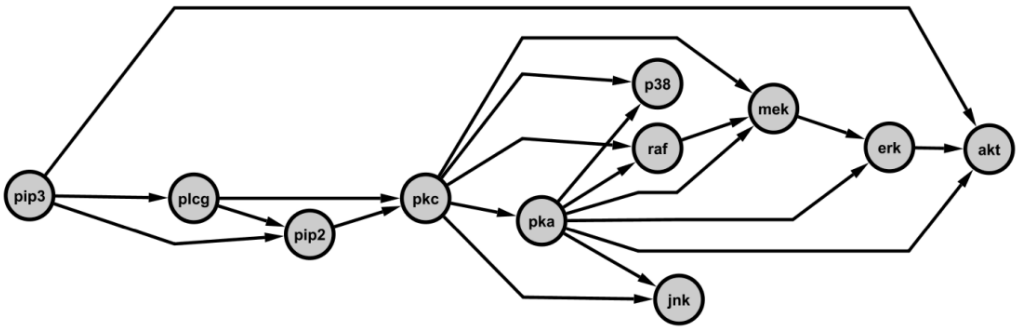


Figure 2: The DAG that describes the RAF protein activation cascade; taken from the slides of Lecture 9. This DAG has been used for generating the simulated data sets

Data generation. The function which generates data has been provided during the course and used with no modifications. Each node's data is constructed as a linear combination of its parents' data, with a bit of Gaussian noise added (conditional dependencies are thus ensured); the data for nodes with no parents is generated directly from a normal distribution. Each node's data is Z standardized.

Scoring. In order to judge the goodness of fit of a DAG to the data, we use the BGe score. There are multiple reasons:

- The underlying assumption that the data is normally distributed holds true by design

- It allows for efficient computations of BGe scores for neighbour graphs via single-edge operations, provided that the score of the initial graph is known
- Equivalent DAGs are automatically assigned identical scores

3.1 The baselines

Structure MCMC. strMCMC, in its basic form, has been extensively studied throughout the course. As such, we can omit its theory and instead focus on its use within this project. The R implementation of strMCMC has been provided during the course and used with minimal modifications (a pre-allocation of memory for efficiency). Informed by the instructions for the third assignment, we discard the first 50% of the MCMC samples to account for the burn-in period. We further thin the sample by taking only every 100th MCMC sample, to account for the sample auto-correlation. The total number of MCMC iterations, per run, has been set to 10^5. The result is an MCMC sample from the posterior graph distribution with 500 i.i.d. DAGs.

The quality of the MCMC samples has been assessed by two methods: trace plots and marginal edge posterior scatter plots ("run i versus run j"). Figures 7, 8 show the results for this study. The trace plots indicate that the Markov Chain's distribution converged to the stationary distribution, as there are no large jumps or different plateaus at which the BGe score settles. However, based on a subjective reading of the marginal edge posteriors, the number of MCMC samples is at most adequate. The agreement between runs becomes better with higher m, justifying the intuitive notion that results based on larger data sets are more trustworthy. It is expected that 10^6 or even 10^7 iterations may bring significant improvements, but because of time constraints 10^5 has been chosen.

Gradient Ascent. The principle behind GA with randomized restarts is shown in Algorithm 2. The algorithm requires a $\mathbb{R}^{n\times m}$ matrix containing the data to be learned from, a variable which bounds the maximum number of GA iterations, and a variable which gives how many times GA is repeated from randomly initialized starting points. If the local optimum is not found within *restarts* iterations, the algorithm will return the best DAG found so far and a variable which signals that convergence has not been achieved. The GA algorithm has been implemented manually, with the help of code provided during the course. Appendix A contains more details about the algorithms.

The convergence speed of GA to local optima has been investigated in the following way. Firstly, two data sets with $m = 20, 50, 100$ data points respectively have been generated. On each data set, 30 GA runs have been employed, followed by recording the number of GA steps until convergence. The results showed that for each data set, the median number of steps fluctuated around 25; the maximum number of steps, among all runs, never exceeded 40. It is thus feasible to run GA for many restarts.

While the above may indicate that GA is a fast method, it is important to note that the true computational cost is hidden in the evaluation of all neighbour graphs. In contrast to strMCMC, where each step requires operations on only one randomly sampled neighbour, GA must evaluate all the neighbours. When the number of nodes n is large, GA can become infeasible.

3.2 Simulated Annealing

The SA algorithm has two important hyperparameters: *the temperature schedule* (which includes the number of iterations N) and *the number of restarts*. Appropriate values for these hyperparameters are investigated prior to running SA in comparison to GA/strMCMC.

3.2.1 Temperature schedules

In the course of this report, we investigate two types of evolutions:

- *Exponential* with parameter a, denoted by Exp. (a). The formula is: $T_i = (1 + 10^{-a})^{-i}$
- *Power* with parameter a, denoted by Pow. (a). For a SA run with N iterations, the formula is: $T_i = \left(1 - \frac{i}{N+1}\right)^a$

Algorithm 2 Gradient Ascent with randomized restarts

Require: *data*, *max_iter*, *restarts*
  $n \leftarrow$ number of nodes to be learned
  $m \leftarrow$ number of data points
  for $i \leftarrow 1$; $i \leq$ restarts; $i \leftarrow i + 1$ **do** ▷ Repeat GA learning *restarts* times
    Generate a random incidence matrix *inc_mat_start*
    Initialize best DAG so far: *inc_mat_best* $\leftarrow$ *inc_mat_start*
    Compute the BGe score for *inc_mat_best*
    $j \leftarrow 1$
    while (Local) maximum not found & $j \leq$ max_iter **do** ▷ Search through neighbours
      neigh_list $\leftarrow$ all neighbour graphs of *inc_mat_best* via single-edge operations
      for Each neighbour in *neigh_list* **do**
        Compute the BGe of the neighbour
        Store the score
      end for
      Find the neighbour with the best score *neigh_best*
      if Score of *neigh_best* $>$ score of *inc_mat_best* **then**
        Update *inc_mat_best* $\leftarrow$ *neigh_best*
        Update *inc_mat_best*'s score
        $j \leftarrow j + 1$
      else
        inc_mat_best is the local optimum
        Store *inc_mat_best* and its score
        Terminate the i's GA learning instance
      end if
    end while
  end for
    return The *inc_mat_best* with the highest overall score

In particular, we use $a = 2, 4$ respectively. Figure 3 shows the temperature schedules for a run with $N = 2000$ iterations.

Properties. For the exponential schedule, the parameter a controls how quickly T decays. Higher values lead to *slower* decay, which should theoretically help with convergence to better-scoring graphs. On a logarithmic plot, the decay speed is constant with the iterations. For the power schedule, we observe a significantly different behaviour: the temperature decay is slow in the beginning, but very accelerated towards the end of the run. The implication is that the power schedule allows SA to explore the landscape of DAGs for a significant portion of the run, after which the evolution is quickly frozen when the iterations approach N. Another property of the power schedule is that the temperature depends on the ratio i/N, and not on the iteration number alone. Larger N allow for a longer exploration period, but not for probing lower absolute temperatures efficiently. The lowest log-temperature is $-a \log(N + 1)$; hence, N mainly controls the length of the exploration period, while a controls the decay speed and the lowest temperature achieved.

Preliminary investigations design. A key component of a well-designed SA run is the number of iterations N. If it is too low, then the SA is still in the exploration phase and the resulting DAGs are not in the high-scoring regions; if it is too large, then computation resources are wasted. There are two questions to be answered: (1) when do SA runs cease to evolve with the iterations and (2) how do the scores change with the temperature schedules? To answer the questions, we generate two data sets for each $m = 20, 50, 100$ respectively. On each data set, we run SA with 5 restarts and for all temperature schedules ($N = 2000$). For one data set at $m = 50$, we also run SA with $N = 60\ 000$ iterations. Some trace plots of SA runs are shown in Figures 4, 9.

Preliminary investigations results. The absolute value of the log-BGe scores differ a lot in between simulated data sets, e.g. by hundreds of points. However, when the shift is accounted for, *the differences between average scores across methods are remarkably similar*. The general trend ($N = 2000$):

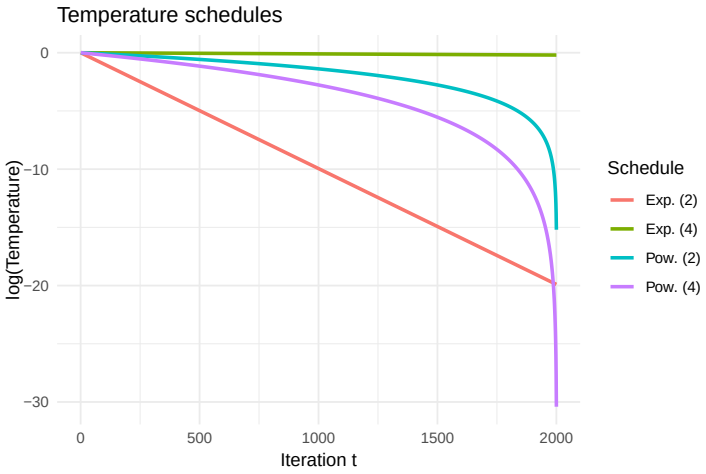


Figure 3: The temperature schedules used for most preliminary investigations of SA convergence

- Pow. (2) and Pow. (4) have similarly good performances on average. Their variances are hard to judge (no one has a variance consistently smaller than the other)
- Exp. (2) is slightly worse than Pow. (2) and Pow. (4) (by no more than 3 points), but often within the overlap of the t-test 95% confidence intervals
- Exp. (4) is significantly worse than Exp. (2) (by 7-10 points), with no overlap between the t-test 95% confidence intervals. This is explained by the fact that, graphically, we see that the SA run did not escape the exploratory period

When $N = 60\ 000$, the scores of the temperature schedules increase.

- Pow. (2), Pow. (4), the increase of the averages is 1-3 points; the increase of the maxima among a sample is < 1 point
- Exp. (2) is relatively unchanged
- Exp. (4) improves drastically, by 8 points
- Exp. (4) is similarly good to Pow. (2) and Pow. (4), while Exp. (2) is a worse. It is to be expected, as Exp. (2) escapes the exploratory period much faster than the others and does not have time to find the regions with high scores

We may ask if the increase of the average scores justifies the larger computational cost; to do so, we can compare the best performers of $N = 2000$ with Exp. (4) for $N = 60\ 000$ (i.e. the method that benefitted the most). On the same data set, the difference in averages is ≈ 2 points, and in maxima is < 1 point. Considering the added time (from 20 seconds/run to 10 minutes/run), the answer is **no**. The extra computation time should rather be invested in more restarts.

The temperatures where SA iterations cease to evolve fluctuate across data sets, but it was possible to identify some approximate upper bounds; the values are given in Table 1. The values are consistent with the properties described earlier in the section. Pow. (4) decays faster than Pow. (2), so SA's effective evolution terminates at an earlier time point; the justification for Exp. (2) is similar.

Temperature schedule for the final runs. Based on the above, it was decided to use $N = 1000$ iterations for Exp. (2), $N = 46\ 000$ for Exp. (4), $N = 5000$ for Pow. (2) and Pow. (4) but with an early termination at $0.9N$ and $0.65N$ respectively.

3.2.2 Number of restarts

In the design of GA and SA with restarts, only the DAG with the optimal score among all runs is chosen. Henceforth it is important to determine an appropriate number of restarts, after which the probability of

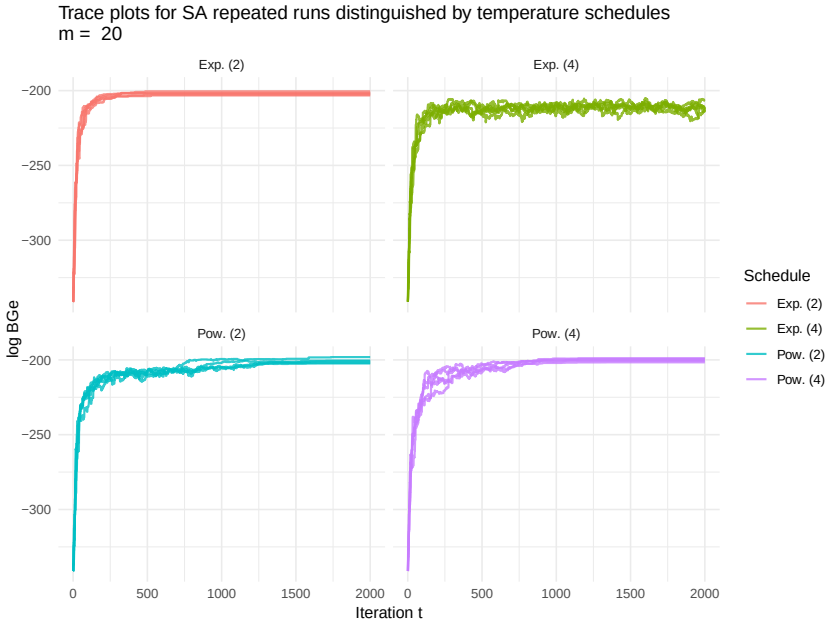


Figure 4: Trace plots of the 5 SA runs, for each temperature schedule ($N = 2000$), for one data set of size $m = 20$. The other figures (not shown) are similar

Temp. schedule	Exp. (2)	Exp. (4)	Pow. (2)	Pow. (4)
$\approx \log T_{\text{stop}}$	-10	-4.6	-4.2	-4
$\approx N_{\text{stop}}$	1000	46 000	-	-
$\approx \left(\frac{i}{N}\right)_{\text{stop}}$	-	-	0.9	0.65

Table 1: The approximate temperatures and the corresponding iterations at which the evolution of SA freezes. Changes are not guaranteed to stop, but they become so rare that the extra computation time is not justified

obtaining a new maximum and the expected increase of the score of the new maximum over the previous is sufficiently small. In this report, we estimate them via non-parametric bootstrapping; the details are given in Appendix B.

Results. A single data set for each $m = 20, 50, 100$ has been generated. On each data set, GA and SA with $N = 2000$ have each been run 31 times. The results of the run are summarized in Figure 5. Overall, the expected improvement of the maximum shows clearly the diminishing returns nature with the number of restarts. Generally, the higher m is, the stronger this effect becomes.

For 6 restarts, the probability that a new score is a maximum is $\approx 1/7 \approx 0.14$. It was observed that the maximum among 6 restarts improves the score of the DAG by 60-80% of the total improvement possible (up to 30 restarts), relative to the average value of a *single* run. This difference, in absolute log-BGe points, is sufficient to cover the differences in average log-BGe values for the scores of all methods except Exp. (4). In other words, the expected improvement due to 6 restarts is sufficient to change the ranking of GA or SA in terms of log-BGe.

While there is no objective criterion on how many restarts to have, it was decided that 6 restarts compromise well between expected improvements and computation time.

3.3 Comparison of methods

Design. In order to compare the ability of each method to learn the underlying relationships, the Area Under the Receiver-Operator Curve (AUROC) has been used. 20 data sets have been generated for each $m = 20, 50, 100$. On each data set, the following learning methods have been employed with the hyperparameters described in the earlier sections: strMCMC, GA and SA (four temperature schedules).

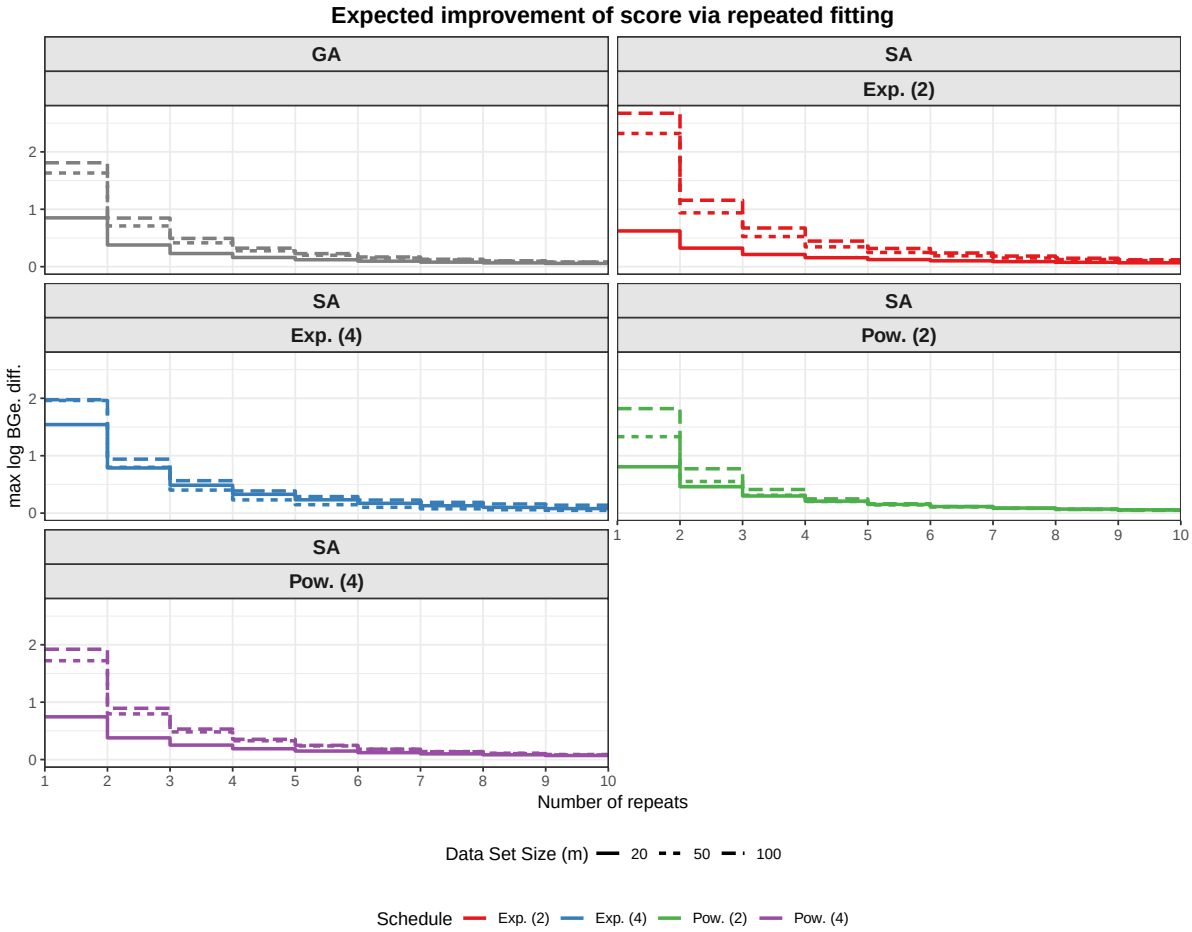


Figure 5: The bootstrap estimate for the difference of expected values of the maxima in a sample of size K, based on 31 scores. At $K = t$, it gives the difference between the maxima of a sample size t and a sample size $t + 1$

To determine whether the distribution of AUROC values differs significantly between methods, we have relied on t-tests. In particular:

- If the t-test 95% confidence intervals between two methods do not overlap, then they are significantly different
- If the confidence intervals overlap, an additional t-test is performed. It two methods belong to the same m, we use a *paired* sample test. Otherwise we rely on the *unpaired* sample test

Results. The distribution of AUROC values, in the form of boxplots, is shown in Figure 10 (Appendix). The t-test 95% confidence intervals based on the AUROC distribution are shown in Figure 6. The findings can be summarized as follows:

- The baseline strMCMC is always the superior method by a large margin: both in terms of AUROC, and in terms of the achieved outcome for the amount of computations required. We can remove it from the discussion
- GA has the lowest average AUROC for all m. However, judging from the paired t-test, the difference between the other methods is not statistically significant. To a threshold of 5%, the only cases where GA was significantly inferior is for $m = 100$ relative to Exp. (4) and Pow. (4)
- To a threshold of 10%, there is no statistically significant difference between different temperature schedules for SA. Only two exceptions: Pow. (2) - Pow. (4) $m = 100$ (p-val $= 0.0092$) and Exp. (2) - Exp. (4) $m = 20$ (p-val $= 0.087$)

- With 3 exceptions out of 12, increasing the available data always led to a statistically significant (5%) improvement in the AUROC scores. The intuitive notion that more data helps with inference holds true

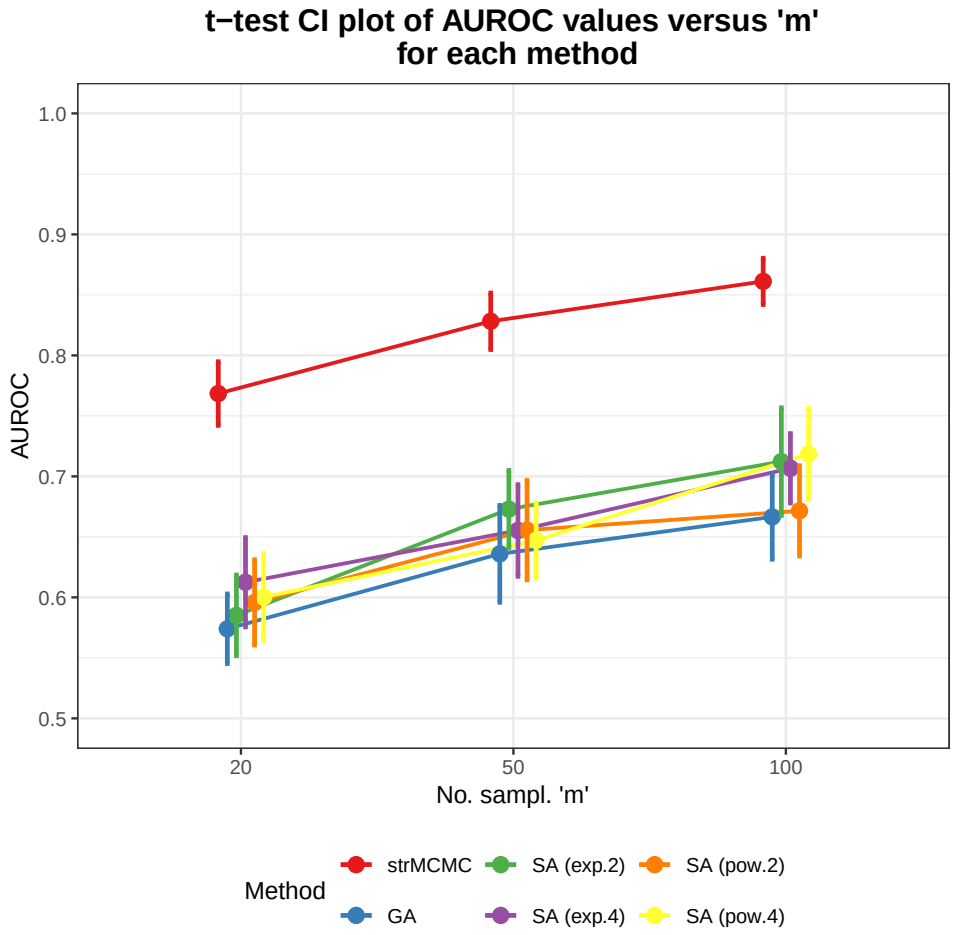


Figure 6: t-test 95% confidence intervals of the AUROC values shown in Figure 10

3.4 Discussion

Differences among temperature schedules. It is surprising that there was no systematic statistical difference between the AUROC scores of different temperature schedules, even more so considering that Exp. (4) is a slowly-decaying schedule that was given a large number of iterations compared to the others. With the investigated temperature schedules, we conclude that the choice is hardly relevant. There could be several implications for this:

- the chosen temperature schedules are inadequate. One could use an exponential schedule with larger a or more iterations N
- other "hidden" hyperparameters to the search, such as the proposal function Q in the Metropolis-Hastings scheme or the scoring metric, are not optimal for SA in this problem. One could wonder if a score such as the BIC would lead to better AUROC results and a demarcation among temperature schedules

SA relative to the rest. When compared with the two baselines (GA, strMCMC), SA is in between. However, SA is very close in strength to GA to the point of no statistically significant difference. Hence, it is difficult to justify why one should opt for the expensive SA when GA performs just as well; a possible situation is when the number of variables n is very large, such that an evaluation of all neighbours

becomes prohibitive. But in that situation, a natural question arises: would it not be better to invest the computational resources in strMCMC?

strMCMC is a significantly superior method to SA, at least according to this simulation. This difference could be fundamental: a great strength of strMCMC is that every sampled graph contributes to the scoring of edges, whereas SA discards everything but the final result. However, it is also important to keep in mind that each strMCMC run was assigned more iterations (10^5 compared to $0.46 \cdot 10^5$ for SA with Exp. (4)). Could it have made a difference? In my opinion, the answer is probably no because SA is not a method that benefits directly from more iterations per run, like strMCMC does. Furthermore, the fact that SA has been re-run 6 times partly accounts for this discrepancy.

4 Conclusion

SA provides a very interesting framework in the context of combinatorial optimization, with clever mathematics that blurs the line between frequentist and Bayesian structure learning. The simulation study, however, did not indicate that SA should be prioritized over GA or strMCMC. SA's weakness may stem from an inadequate choice of temperature schedules, as it was clearly warned in Koller and Friedman, 2009. With an appropriate choice of temperature schedule, SA could become competitive at least relative to GA, especially for n large. Furthermore, implementational improvements from literature (namely SAGA and PSAGA) allow SA to still be a reasonable choice for structure learning.

References

Adabor, E. S., Acquaah-Mensah, G. K., & Oduro, F. T. (2015). SAGA: A hybrid search algorithm for bayesian network structure learning of transcriptional regulatory networks. *Journal of Biomedical Informatics*, *53*, 27–35. https://doi.org/https://doi.org/10.1016/j.jbi.2014.08.010

Granville, V., Krivánek, M., & Rasson, J.-P. (1994). Simulated annealing: A proof of convergence. *IEEE Trans. Pattern Anal. Mach. Intell.*, *16*, 652–656. https://api.semanticscholar.org/CorpusID:7865982

Hammond, J., & Smith, V. A. (2025). Bayesian networks for network inference in biology. *Journal of The Royal Society Interface*, *22*(226), 20240893. https://doi.org/10.1098/rsif.2024.0893

Janžura, M., & Nielsen, J. (2006). A simulated annealing-based method for learning Bayesian networks from statistical data: Research articles. *Int. J. Intell. Syst.*, *21*(3), 335–348.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models*. The MIT press.

Lee, S., & Kim, S. B. (2020). Parallel simulated annealing with a greedy algorithm for bayesian network structure learning. *IEEE Transactions on Knowledge and Data Engineering*, *32*(6), 1157–1166. https://doi.org/10.1109/TKDE.2019.2899096

A More about the algorithms

Exploring all neighbour graphs of a DAG. A neighbour graph is any valid DAG that can be reached from the starting DAG by a single edge operation (reversal, addition or deletion). The function which generates all neighbour graphs of a given DAG, namely `GET_ALL_NEIGHBOURS`, has been greatly inspired by the function which has been provided during the course: `PROPOSE_NEW_DAG`.

- Similarities:
  - Both functions identify all valid single edge operations through efficient matrix computations
  - Given an operation and edge, the mechanisms by which the incidence and ancestor matrices are updated are identical
- Differences:
  - `PROPOSE_NEW_DAG` samples both an operation and an edge. The number of neighbour graphs is important, but the exact edges and operations that describe the other neighbours are irrelevant. It returns a single neighbour graph
  - `GET_ALL_NEIGHBOURS` systematically goes through each valid operation and edge, and it returns the corresponding neighbour. The number of neighbour graphs is irrelevant

The implementation of the above search routine for this report did not have efficiency as a priority. In the context of GA searches, it is possible to increase the efficiency through memoization. However, the latter poses an implementational challenge rather than a mathematical one.

Random DAG initialization. A key component of GA searches is the randomly generated starting point. The mechanism by which to generate a random DAG posed an interesting challenge, as it was not possible to rely on the already-provided `PROPOSE_NEW_DAG`. The latter selects a random neighbour DAG, whereas we would like the possibility to obtain any DAG from the entire search space. *Disclosure*: in what follows, the DAG generation mechanism is my own work. The R implementation and a refinement on a hyperparameter has been done with the help of Google Gemini. However, the soundness of the implementation has been verified manually.

The random sampling of DAGs can be decomposed in two tasks: a (uniform) sampling of the *topological order* of nodes, and a sampling of the *edges* that connect the nodes in topological order. A great advantage of this decomposition is that the result is guaranteed to be a valid DAG. Furthermore, we can quickly deduce that there is no bias with regards to the labelling of nodes. Sampling mechanism:

- *Topological order*. We randomly and uniformly select a permutation σ of the first n integers (where $n = 11$ is the number of nodes). We identify that permutation with the topological order
- *Edges*. Let U be an $\mathbb{R}^{n\times n}$ upper triangular matrix and $p \in (0, 1)$. We first sample U:
  - each non-zero element is sampled from the Unif$(0, 1)$ distribution independently
  - elements that are $\leq p$ are set to 1. The rest are set to 0
  - interpretation: if $U_{ij} = 1$, the edge $\sigma(i) \to \sigma(j)$ is present. Otherwise it is absent

The effect is that each edge is sampled independently, with the probability of it appearing equal to p. The average number of edges in the DAGs sampled is $p \cdot n(n-1)/2$. Hence, if p is large, then the algorithm has a preference for dense networks. Conversely, a low p results in sparse DAGs appearing with a higher frequency. For this report, we have set $p = 2/n$ such that the expected number of edges is $n - 1$. The result is that the GA algorithm searches mainly through DAGs where the number of edges is of the same order as the number of nodes.

p could be viewed through the lens of Bayesian statistics, playing the role of a prior on the complexity of the underlying network. It is possible that AUROC results for GA may improve with another choice of p, e.g. by relying on expert knowledge and trying to improve a "gold standard" network. However, it is difficult to imagine how one could systematically and optimally choose p when no prior knowledge is available.

B Bootstrap estimate of new maxima

Let us fix a data set, and consider one stochastic technique (e.g. GA or SA with one temperature schedule). After running the techniques N times, we obtain a sample $X_1, X_2, \ldots, X_N$ of scores for the resulting DAG of the search. Because we do not change the parameters of the search, the sample is i.i.d. When N is much smaller than the total number of DAGs with n nodes, the probability that two scores are equal is negligible. For $K < N$, we estimate

$$\mathbb{P}\left(X_{K+1} > \max\left(X_1, \ldots, X_K\right)\right) \quad \text{and} \quad E\left[\max\left(X_1, \ldots, X_{K+1}\right)\right] - E\left[\max\left(X_1, \ldots, X_K\right)\right]$$

Let us denote the realizations of X_i by x_i; without loss of generality, let us say that x_i are sorted non-decreasing via i. For non-parametric bootstrapping, the probability that $X = x_i$ equals $1/N$. The bootstrap estimate of the probability is

$$\begin{aligned}
\mathbb{P}(X_{K+1} > \max\left(X_1, \ldots, X_K\right)) &= \sum_{x \in f(S)} \mathbb{P}\left(X_{K+1} = x\right) \cdot \mathbb{P}\left(x > \max\left(X_1, \ldots, X_K\right)\right) \\
&= \sum_{x \in f(S)} \mathbb{P}\left(X_{K+1} = x\right) \cdot \mathbb{P}\left(x > X_1, \ldots, X_K\right) = \sum_{x \in f(S)} \mathbb{P}\left(X_{K+1} = x\right) \cdot \prod_{i=1}^{K} \mathbb{P}\left(X_i < x\right) \\
&= \sum_{x \in f(S)} \mathbb{P}\left(X_1 = x\right) \cdot \mathbb{P}\left(X_1 < x\right)^K \approx \sum_{x \in \{x_1, \ldots, x_N\}} \mathbb{P}\left(X_1 = x\right) \cdot \mathbb{P}\left(X_1 < x\right)^K \\
&= \sum_{i=1}^{N} \frac{1}{N} \left(\frac{i-1}{N}\right)^K \to \int_0^1 x^K dx = \frac{1}{K+1} \quad \text{as } N \to \infty
\end{aligned}$$

where in the first line we use the law of total probability, in the second we use the independence of the sample, and in the third we use the identical distribution and the bootstrap estimate. In the approximation, we rely the fact that the probability of ties is negligible.

For the expected value of the maximum, we recall the the CDF of the maximum of a sample is related to the CDF of one value via

$$\mathbb{P}\left(\max\left(X_1, \ldots, X_K\right) \leq x\right) = \mathbb{P}\left(X_1 \leq x\right)^K \coloneqq F(x)^K$$

The non-parametric bootstrap estimate of the CDF is

$$\hat{F}(x) = \frac{1}{N} \sum_{i=1}^{N} 1(x_i \leq x)$$

The expected value of the maximum becomes

$$\begin{aligned}
E\left[\max\left(X_1, \ldots, X_K\right)\right] &= \sum_{x \in f(S)} x \cdot \mathbb{P}\left(\max\left(X_1, \ldots, X_K\right) = x\right) = \sum_{x \in f(S)} x \cdot \left(F(x)^K - F(x-)^K\right) \\
&\approx \sum_{x \in \{x_1, \ldots, x_N\}} x \cdot \left(\hat{F}(x)^K - \hat{F}(x-)^K\right)
\end{aligned}$$

It is interesting that the limit probability does not depend on the realizations of the scores. This indicates that the probability, as a metric for judging the number of restarts, is "too robust" to changes in the values of the realizations. Henceforth, the use of the difference in expected maxima is justified. The nature of the distinction is not too dissimilar to that of the median and the mean, and their robustness to outliers.

C Additional figures

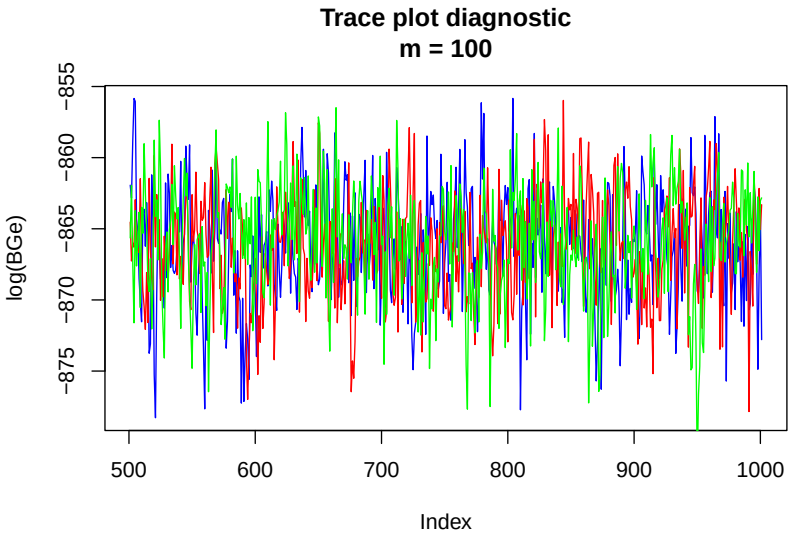
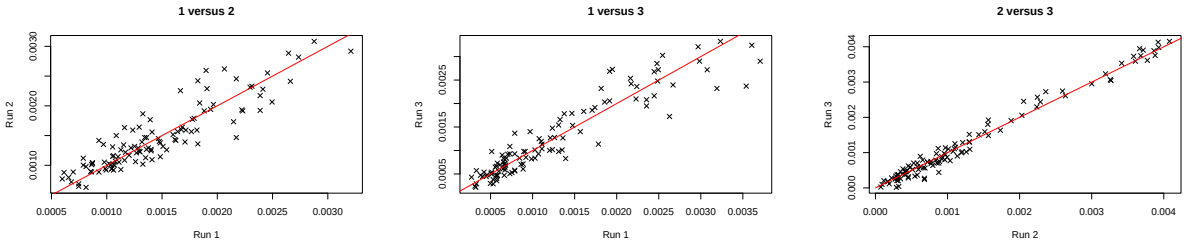


Figure 7: Trace plot of 3 MCMC runs for a data set with $m = 100$ data points. The figures for $m = 20, 50$ (not shown) are similar



(a) $m = 20$ (b) $m = 50$ (c) $m = 100$

Figure 8: Marginal edge posterior scatter plots on data sets with $m = 20, 50, 100$ data points respectively. Each figure displays the worst agreement between the MCMC runs, given a data set

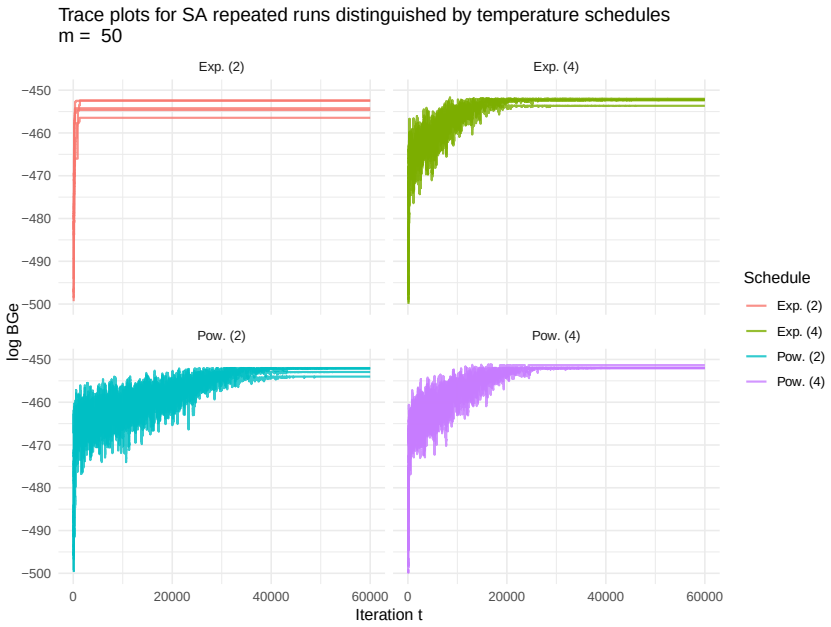


Figure 9: Trace plots of the 5 SA runs, for each temperature schedule ($N = 60\ 000$), for one data set of size $m = 50$

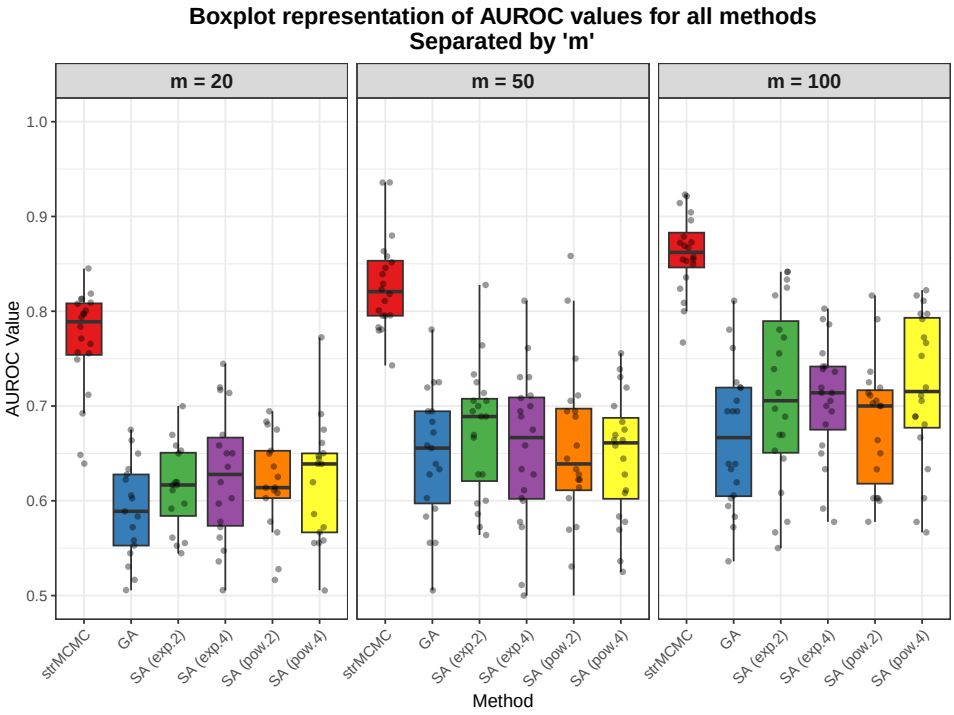


Figure 10: Boxplot of the AUROC values for each method, distinguished by the size of the data sets generated