Very Fast Simulated Reannealing

Dionissios T. Hristopulos
School of Electrical and Computer Engineering, Technical University of Crete, Chania, Crete, Greece

Abbreviations

ASA	Adaptive simulated annealing
SA	Simulated annealing
SQ	Simulated quenching
VFSR	Very fast simulated reannealing

Definition

Very fast simulated reannealing (VFSR) is an improved version of simulated annealing (SA). The latter is a global optimization method suitable for complex, non-convex problems. It aims to optimize an objective function $\mathcal{H}(\mathbf{x})$, with respect to the D-dimensional parameter vector $\mathbf{x} = (x_1, \ldots x_D)^\top$. Simulated annealing relies on random Metropolis sampling of the parameter space which mimics the physical annealing process of materials. The annealing algorithm treats $\mathcal{H}(\mathbf{x})$ as a fictitious energy function. The algorithm proposes moves which change the current state $\mathbf{x}$, seeking for the optimum state. The proposed states are controlled by an internal parameter which plays the role of temperature and controls the acceptance rate of the proposed states.

Very fast simulated reannealing employs a fast temperature reduction schedule in combination with periodic resetting of the annealing temperature to higher values. In addition, it allows different temperatures for different parameters, and an adaptive mechanism which tunes temperatures to the sensitivities of the objective function with respect to each parameter. These improvements allow fast convergence of the algorithm to the global optimum. For reasons of conciseness and without loss of generality, it is assumed in the following that the optimization problem refers to the *minimization* of the objective function $\mathcal{H}(\mathbf{x})$.

Overview

Annealing is a physical process which involves heat treatment and is used in metallurgy to produce materials (e.g., steel) with improved mechanical properties. It was presumably crucial in the manufacturing of Damascus steel swords which were famous in antiquity for their toughness, sharpness, and strength. The well-kept secrets of Damascus swordmaking were lost in time as the interest in swords as weapons waned. However, their legend survived in popular culture, inspiring the references to "Valyrian steel" in the popular *Game of Thrones* saga. The secrets of the long-lost art were presumably re-discovered in 1981 by Oleg D. Sherby and Jeffrey Wadsworth at Stanford University. A key factor for the effectiveness of annealing is the heating process which enables the material to escape metastable atomic configurations (corresponding to local optima of the energy) and to find the most stable configuration that corresponds to the global minimum of the energy.

Simulated annealing (SA) is a stochastic optimization algorithm developed by Scott Kirkpatrick and co-workers. It uses randomness in the search for the global minimum of $\mathcal{H}(\mathbf{x})$. The randomness is introduced through (i) a proposal distribution which generates new proposal states $\mathbf{x}_{new}$ of the parameter vector and (ii) through probabilistic decisions whether to accept or reject the proposal states. In contrast, in deterministic optimization methods, every move is fully determined from the current state and the properties of the objective function.

Simulated annealing belongs in a class of models inspired by physical or biological processes; these algorithms are known as *meta-heuristics* (see Ingber (2012) and references therein.) Physical annealing involves a heat treatment process which in SA is simulated using the Metropolis Monte Carlo algorithm (Metropolis et al. 1953). At every iteration of the SA algorithm, a random solution (state) $\mathbf{x}_{\mathrm{new}}$ is generated for the parameters by local perturbation of the current state $\mathbf{x}_{\mathrm{cur}}$. The proposed state is accepted and becomes the new current state according to an acceptance probability; the latter depends on the value of $\mathcal{H}(\mathbf{x}_{\mathrm{new}})$ compared to $\mathcal{H}(\mathbf{x}_{\mathrm{cur}})$ and an internal SA "temperature" variable T. The new state is accepted if the proposal lowers the objective function. On the other hand, even states $\mathbf{x}_{\mathrm{new}}$ such that $\mathcal{H}(\mathbf{x}_{\mathrm{new}}) > \mathcal{H}(\mathbf{x}_{\mathrm{cur}})$ are assigned a non-zero acceptance probability which is higher for higher temperatures.

The SA temperature is initially set to a problem-specific, user-defined high value which allows higher acceptance rates for sub-optimal states. This choice helps the algorithm to effectively explore the parameter space because it allows escaping from local minima of the objective function. The temperature is then gradually lowered in order to "freeze" the system in the minimum energy state. The function $T(k)$, where k is an integer index, describes the evolution of temperature as a function of the "annealing time" k and defines the *SA cooling schedule*. This should be carefully designed to allow convergence of the algorithm to the global minimum (Geman and Geman 1984; Salamon et al. 2002).

SA can be applied to objective functions with arbitrary nonlinearities, discontinuities, and noise. It does not require the evaluation of derivatives of the objective function, and thus it does not get stuck in local minima. In addition, it can handle arbitrary boundary conditions and constraints imposed on the objective function. On the other hand, SA requires tuning various parameters, and the quality of the solutions practically achieved by SA depends on the computational time spent. SA provides a statistical guarantee for finding the (global) optimal solution, provided the correct combination of *state proposal (generating) distribution* and cooling schedule are used (Ingber 2012). The convergence of the algorithm is based on the weak ergodic property of SA which ensures that almost every possible state of the system is visited. Achieving the statistical guarantee can in practice significantly slow down classical SA, since a very gradual cooling schedule is required. To reduce the computational time, often a fast temperature reduction schedule is applied, i.e., $T(k+1) = cT(k)$, where $0 < c < 1$, which amounts to *simulated quenching (SQ)* of the temperature. However, this exponential cooling schedule does not guarantee convergence to the global optimum.

Very fast simulated reannealing (VFSR) is also known as adaptive simulated annealing (ASA) (Ingber 1989, 2012). ASA is the currently preferred term, while VFSR was used initially to emphasize the fast convergence of the method compared to the standard Boltzmann annealing approach. ASA employs a generating probability distribution for proposal states which allows an optimal cooling schedule. This setup enables the algorithm to explore efficiently the parameter space. *Reannealing* refers to a periodical resetting of the temperature to higher (than the current) value after a number of proposal states have been accepted. Then, the search begins again at the higher temperature. This strategy helps the algorithm to avoid getting trapped at local minima. Finally, ASA allows for different temperatures in each direction of the parameter space. The parameter temperatures determine the width of the generating distribution for each parameter, thus enabling *the cooling schedule to be adapted* according to the sensitivity of the objective function in each direction. ASA maintains SA's statistical guarantee of finding the global minimum for a given objective function, but it also features significantly improved convergence speed. It has been demonstrated that ASA is competitive with respect to other non-gradient-based global optimization methods such as genetic algorithms and Taboo search (Chen and Luk 1999; Ingber 2012).

Methodology

The SA algorithm is a method for global – possibly constrained – optimization of general, nonlinear, real-valued objective functions $\mathcal{H}(\mathbf{x})$ where $\mathbf{x} \in \chi \subset \mathbb{R}^D$ is a vector of parameters that takes values in the space χ:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\chi} \mathcal{H}(\mathbf{x}), \text{potenially with constraints on } \mathbf{x}.$$

The standard SA algorithm is based on the Markov chain Monte Carlo method (Geman and Geman 1984). It involves homogeneous Markov chains of finite length which are generated at progressively lower temperatures. The following parameters should thus be specified: (i) A sufficiently high initial temperature T_0; (ii) a final "freezing" temperature T_f (alternatively a different stopping criterion); (iii) the length of the Markov chains; (iv) the procedure for generating a proposal state $\mathbf{x}_{\mathrm{new}}$ "neighboring" the current state $\mathbf{x}_{\mathrm{cur}}$; (v) the acceptance criterion which determines if the proposal state $\mathbf{x}_{\mathrm{new}}$ is admitted; and (vi) a rule for temperature reduction (annealing schedule).

Boltzmann Simulated Annealing

The SA algorithm is initialized with a guess for the parameter vector $\mathbf{x}_0$. The simulation proceeds by iteratively proposing new states $\mathbf{x}_{\mathrm{new}}$ based on the *Metropolis algorithm* (Metropolis et al. 1953). These proposal states are generated by perturbing the current state $\mathbf{x}_{\mathrm{cur}}$. For example, in Boltzmann annealing, this can be done by drawing the proposal state $\mathbf{x}_{\mathrm{new}}$ from the proposal distribution

$$g(\mathbf{x}_{\text{new}}|\mathbf{x}_{\text{cur}}) = \frac{1}{(2\pi T)^{D/2}} \exp\left[-\|\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{cur}}\|^2/2T\right].$$

For every proposed move, the difference $\Delta\mathcal{H} = \mathcal{H}(\mathbf{x}_{\text{new}}) - \mathcal{H}(\mathbf{x}_{\text{cur}})$ between the current state, $\mathbf{x}_{\text{cur}}$, and the proposed state, $\mathbf{x}_{\text{new}}$, is evaluated. If $\Delta\mathcal{H} < 0$, the proposed state is accepted as the current state. On the other hand, if $\Delta\mathcal{H} > 0$, the acceptance probability is given by the exponential expression $P_{\text{acc}} = 1/[1 + \exp \Delta\mathcal{H}/T]$ (Salamon et al. 2002; Ingber 2012). The decision whether to accept $\mathbf{x}_{\text{new}}$ is implemented by generating a random number $r{\sim}U(0, 1)$ from the uniform distribution between 0 and 1; if $r \leq P_{\text{acc}}$, the proposed state is accepted ($\mathbf{x}_{\text{cur}}$ is updated to $\mathbf{x}_{\text{new}}$), otherwise the present state is retained as the current state.

The above procedure is repeated a number of times before the temperature is lowered. The *acceptance rate* is equal to the percentage of proposal states that are accepted. The initial temperature is selected to allow high acceptance rate (e.g., ≈80%) so that the algorithm can move between different local optima. The temperature reduction is determined by the *annealing schedule* which is a key factor for the performance of the SA algorithm.

The annealing schedule depends on the form of the generating function used: to ensure that the global minimum of $\mathcal{H}(\mathbf{x})$ is reached, it must be guaranteed that all states $\mathbf{x} \in \chi$ can be visited an infinite number of times during the annealing process. In the case of Boltzmann annealing, this condition requires a cooling schedule no faster than logarithmic, i.e., $T_k = T_0 \ln k_0/ \ln k$, where $k_{\max} \geq k \geq k_0$. The integer index k counts the simulation *annealing time* and $k_0 > 1$ is an arbitrary initial counter value (Ingber 2012). The final temperature, $T_{k_{\max}}$, should be low enough to trap the objective function in the global optimum state.

Fast (Cauchy) Simulated Annealing

Boltzmann SA requires a slow temperature reduction schedule as determined by the logarithm function. In practical applications, an *exponential schedule* $T_{k+1} = cT_k$ where $0 < c < 1$ is often followed. However, this fast temperature reduction enforces *simulated quenching* which drives the system too fast toward the final temperature. The exponential annealing schedule offers computational gains, but it does not fulfill the requirement of weak ergodicity (infinite number of visits to each state). Hence, it does not guarantee convergence of SA to the global minimum.

A *fast annealing* schedule which lowers the temperature according to $T_k = T_0/k$ is suitable for the *Cauchy generating distribution*:

$$g(\mathbf{x}_{\text{new}}|\mathbf{x}_{\text{cur}}) = \frac{T}{\left(\|\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{cur}}\|^2 + T^2\right)^{(D+1)/2}}.$$

Fast cooling works in this case due to the "heavy tail" of the Cauchy distribution which carries more weight in the tail than the Gaussian distribution used in Boltzmann SA (Ingber 2012; Salamon et al. 2002). The resulting higher probability density for proposal states considerably different than the current state allows the algorithm to visit efficiently all the probable states in the parameter space.

Very Fast Simulated Reannealing

ASA includes three main ingredients similar to classical annealing: the generating distribution of proposal states, the acceptance probability, and the annealing schedule. ASA uses an *acceptance temperature* $T_{\text{acc}}(k_a)$ which controls the acceptance rate of proposed moves and a set of D temperatures $\{T_i(k_i)\}_{i=1}^{D}$ which control the width of the generating distribution for each parameter individually.

In addition to these three ingredients, ASA includes a *reannealing scheme* which rescales all the temperatures after a certain number of steps so that they adapt to the current state of $\mathcal{H}(\mathbf{x})$. Reannealing adjusts the rate of change of the annealing schedule independently for each parameter. This helps the algorithm adapt to changing sensitivities of the objective function as it explores the parameter space, encountering points (in D-dimensional space) with very different local geometry, where $\mathcal{H}$ may change rapidly with respect to some parameters but considerably more slowly with respect to others. The mains steps of ASA are outlined in the text below and in Algorithm 1.

Generation of proposal states: If the parameters x_i are constrained within $[A_i, B_i]$, where A_i and B_i are, respectively, the lower and upper bounds, the ASA proposal states are generated by means of the following steps (Ingber 1989; Chen and Luk 1999; Ingber 2012):

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{cur}} + \Delta\mathbf{x}, \tag{1a}$$

$$\Delta x_i = y_i(B_i - A_i), \quad i = 1, \ldots, D, \tag{1b}$$

$$y_i = \text{sign}\left(u_i - \frac{1}{2}\right) T_i(k_i) \left[\left(1 + \frac{1}{T_i(k_i)}\right)^{|2u_i - 1|} - 1\right] \tag{1c}$$

$$\text{where} \quad u_i \sim U[0, 1], \tag{1d}$$

is a random variable uniformly distributed between zero and one. Consequently, the random variable y_i is centered around zero and takes values in the interval $[-1, 1]$. The integer indices k_i are used to count time. Certain values of y_i can

Very Fast Simulated Reannealing, Algorithm 1 ASA main steps. The integers n and N count the accepted and generated states, respectively. N_s: # proposed states between annealing steps. N_{acc}: # accepted proposals between reannealing steps.

```
Input: Initialize: $\mathbf{x}_0 \in \mathcal{X}$, $T_{acc}(0) \leftarrow \mathcal{H}(\mathbf{x}_0)$, $k_i \leftarrow 1$, $k_a \leftarrow 1$, $\mathbf{x}_{cur} \leftarrow \mathbf{x}_0$,
       $T_i(0) \leftarrow 1$, $n \leftarrow 0$, $N \leftarrow 0$
while termination condition is not met do Annealing loop
    Generate proposal state $\mathbf{x}_{new}$ using Eq. (1) ;
    $N \leftarrow N + 1$ (increase generated states counter) ;
    if $\mathcal{H}(\mathbf{x}_{new}) < \mathcal{H}(\mathbf{x}_{cur})$ then Update current state
        $\mathbf{x}_{cur} \leftarrow \mathbf{x}_{new}; n \leftarrow n + 1$ (increase accepted states counter)
    else
        Calculate acceptance probability $P_{acc}$ from Eq. (2) ;
        if $P_{acc} > r \sim U(0,1)$ then Update current state
            $\mathbf{x}_{cur} \leftarrow \mathbf{x}_{new}; n \leftarrow n + 1$ (increase accepted states counter)
        end
    end
    if $n > N_{acc}$ then
        Reannealing based on Eqs. (3)-(5) ;
        $n \leftarrow 0$ (reset accepted states counter)
    else
        Go to Step 2
    end
    if $N > N_s$ then
        Annealing based on Eqs. (6) ;
        $N \leftarrow 0$ (reset generated states counter)
    end
end
```

yield proposed parameters outside the range $[A_i, B_i]$; these values should be discarded (Ingber 1989). Lower values of temperature force y_i to concentrate around zero, while high temperature values lead to an almost uniform distribution of y_i between -1 and 1.

Acceptance probability: The *acceptance probability* of a proposed state is given by (Chen and Luk 1999)

$$P_{acc} = \frac{1}{1 + e^{\Delta\mathcal{H}/T_{acc}(k_a)}}. \tag{2}$$

In Eqs. (1) and (2), the set of integer indices $\{k_i\}_{i=1}^{D} \cup \{k_a\}$ represents different *annealing times*. ASA uses one time index per parameter, so that the reannealing process can adjust the annealing time differently for each parameter. This multi-dimensional annealing schedule enables adaptation of the proposal states to different sensitivities of $\mathcal{H}(\mathbf{x})$ with respect to the parameters x_i, $i = 1, \ldots, D$.

Reannealing: Every time a number N_{acc} of proposal states have been accepted, *reannealing* is performed. This procedure adjusts the annealing temperatures and times to the local geometry of the parameter space. The *sensitivities* $\{s_i\}_{i=1}^{D}$ are calculated as follows:

$$s_i = \left|\frac{\partial\mathcal{H}(\tilde{\mathbf{x}})}{\partial x_i}\right| \approx \left|\frac{\mathcal{H}(\tilde{\mathbf{x}} + a\widehat{\mathrm{e}}_i) - \mathcal{H}(\tilde{\mathbf{x}})}{a}\right|, \quad i = 1, \ldots, D, \tag{3}$$

where $\tilde{\mathrm{x}}$ is the *current optimal parameter vector*, a is a small increment, and $\widehat{\mathrm{e}}_i$ is a D-dimensional unit vector in the i-th direction of parameter space, i.e., $\{\widehat{\mathrm{e}}_i\}_k = \delta_{i,k}$ ($\delta_{i,k} = 1$ for $i = k$ and $\delta_{i,k} = 0$ for $i \neq k$ being the Kronecker delta).

Reannealing adjusts the temperatures and annealing times as follows ($\leftarrow$ implies assignment of the value on the right side of $\leftarrow$ to the variable appearing on the left side)

$$T_i(k_i) \leftarrow \frac{s_{max}}{s_i} T_i(k_i), \quad s_{max} = \max_{i=1,\ldots,D} s_i, \tag{4a}$$

$$k_i \leftarrow \left[\frac{1}{c} \log\left(\frac{T_i(0)}{T_i(k_i)}\right)\right]^D, \quad i = 1, \ldots, D, \tag{4b}$$

where $c > 0$ is a user-defined parameter that adjusts the rate of reannealing and $T_i(0)$ is usually set to unity (Chen and Luk 1999).

Similarly, the acceptance temperature is rescaled according to

$$T_{\text{acc}}(k_a) \leftarrow \mathcal{H}(\tilde{\mathbf{x}}), \quad T_{\text{acc}}(0) \leftarrow \mathcal{H}(\mathbf{x}_a), \tag{5a}$$

$$k_a \leftarrow \left[\frac{1}{c}\log\left(\frac{T_{\text{acc}}(0)}{T_{\text{acc}}(k_a)}\right)\right]^D, \tag{5b}$$

where $\mathbf{x}_a$ is the *last accepted state*.

The reannealing procedure enables ASA to decrease the temperatures T_i along the high-sensitivity directions of $\mathcal{H}(\mathbf{x})$, thus allowing smaller steps in these directions, and to increase T_i along the low-sensitivity directions, thus allowing larger jumps.

Annealing: After a number of steps N_s have been completed, the *annealing* procedure takes place. The annealing times increase by one, and the annealing temperatures are modified according to Eq. (6). The annealing schedules for the temperature parameters follow the stretched exponential expression

$$\begin{aligned} T_{\text{acc}}(k_a) &= T_{\text{acc}}(0)\exp\left(-ck_a^{1/D}\right), \\ T_i(k_i) &= T_i(0)\exp\left(-ck_i^{1/D}\right), i = 1, \ldots, D, \end{aligned} \tag{6}$$

where $T_i(k_i)$ is the current temperature for the i-th parameter (Ingber 1989).

Termination: The above operations are repeated until the algorithm terminates. Different termination criteria can be used in ASA depending on the available computational resources and a priori knowledge of $\mathcal{H}$. Such criteria include the following: (i) an average change of $\mathcal{H}(\mathbf{x})$ (over a number of accepted proposal states) lower than a specified tolerance; (ii) exceedance of a maximum number of iterations (generated proposal states); (iii) exceedance of a maximum number (usually proportional to D) of $\mathcal{H}$ evaluations; (iv) exceedance of a maximum computational running time; and (v) attainment of a certain target minimum value for $\mathcal{H}$.

Initialization: An initial value, $T_{\text{acc}}(0)$, should be assigned to the acceptance parameter. One possible choice it to set $T_{\text{acc}}(0) = \mathcal{H}(\mathbf{x}_0)$ where $\mathbf{x}_0$ is the random initial parameter vector (Chen and Luk 1999). A more flexible approach matches $T_{\text{acc}}(0)$ with a selected acceptance probability, e.g., $P_{\text{acc}} = 0.25$ (Iglesias-Marzoa et al. 2015). The optimal values of N_s and N_{acc} do not seem to depend crucially on the specific problem (Chen and Luk 1999). Often an adequate choice for N_{acc} is on the order of tens or hundreds and for N_s on the order of hundreds or thousands. Higher values may be necessary to adequately explore high-dimensional and topologically complex parameter spaces. A good choice for the annealing rate control parameter is $c \approx 1 - 10$. In general, higher values lead to faster temperature convergence at the risk that the algorithm may get stuck near a local minimum. Lower values lead to slower temperature reduction and therefore increase the computational time. Given ASA's adaptive ability, the choices for c, N_{acc}, N_s do not critically influence the algorithm's performance but they have an impact on the computational time. For problems where ASA is used for the first time, it is a good idea to experiment with different choices (Iglesias-Marzoa et al. 2015).

Applications

An introduction to SA for spatial data applications is found in Hristopulos (2020). Geostatistical applications are reviewed in Pyrcz and Deutsch (2014). An informative review of ASA (VFSR) is given in Ingber (2012), while a mathematical treatment of SA is presented in Salamon et al. (2002). ASA has been successfully used for geophysical inversion (Sen and Stoffa 1996). The ASA algorithm and its application in various signal processing problems are reviewed in Chen and Luk (1999). A detailed investigation of ASA's application for determining the radial velocities of binary star systems and exoplanets is given in Iglesias-Marzoa et al. (2015).

Software Implementations

In Matlab, SA is implemented by means of the command simulannealbnd. In R, the packages `optimization` and `optim_sa` provide SA capabilities. In Python, this functionality can be found in the `scipy.optimize` module. A C-language code for ASA (VFSR) developed by Lester Ingber is available from his personal website.

Summary and Conclusions

ASA (VFSR) is a computationally efficient implementation of the simulated annealing algorithm which employs a fast (stretched-exponential) annealing schedule in combination with a reannealing program which periodically resets the temperature. ASA is an adaptive algorithm: both the annealing and reannealing schedules take into account the sensitivity of the objective function in different directions of parameter space. Thanks to this adaptive ability, ASA is an effective stochastic global optimization method which has been successfully applied to complex, nonlinear objective functions that may involve many local minima. Estimates of parameter uncertainty can be obtained by means of the Fisher matrix and Markov chain Monte Carlo methods.

ASA converges faster than the classical SA algorithm, it allows each parameter to adapt individually to the local topology of the objective function, and it involves a faster schedule for temperature reduction (Iglesias-Marzoa et al. 2015). The counterbalance of these advantages is that ASA is more complex than the classical SA algorithm, it involves more internal parameters, and the stretched exponential expression for temperature reduction requires using double precision arithmetic and checking of the exponents to avoid numerical problems. Overall, ASA has found several applications in the geosciences (Sen and Stoffa 1996; Pyrcz and Deutsch 2014; Hristopulos 2020).

Cross-References

- ► Markov Chain Monte Carlo
- ► Optimization Methods
- ► Simulated Annealing
- ► Simulation

Bibliography

Chen S, Luk BL (1999) Adaptive simulated annealing for optimization in signal processing applications. Signal Process 79(1):117–128. https://doi.org/10.1016/S0165-1684(99)00084-5

Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans Pattern Anal Mach Intell 6(6):721–741. https://doi.org/10.1109/TPAMI.1984.4767596

Hristopulos DT (2020) Random fields for spatial data modeling: a primer for scientists and engineers. Springer, Dordrecht. https://doi.org/10.1007/978-94-024-1918-4

Iglesias-Marzoa R, López-Morales M, Arévalo Morales MJ (2015) The rvfit code: a detailed adaptive simulated annealing code for fitting binaries and exoplanets radial velocities. Publ Astron Soc Pac 127(952):567–582. https://doi.org/10.1086/682056

Ingber L (1989) Very fast simulated re-annealing. Math Comput Model 12(8):967–973. https://doi.org/10.1016/0895-7177(89)90202-1

Ingber L (2012) Adaptive simulated annealing. In: Hime A, Ingber L, Petraglia A, Petraglia MR, Machado MAS (eds) Stochastic global optimization and its applications with fuzzy adaptive simulated annealing. Springer, Heidelberg, pp 33–62. https://doi.org/10.1007/978-3-642-27479-4

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087–1092. https://doi.org/10.1063/1.1699114

Pyrcz MJ, Deutsch CV (2014) Geostatistical reservoir modeling, 2nd edn. Oxford University Press, New York

Salamon P, Sibani P, Frost R (2002) Facts, conjectures, and improvements for simulated annealing. SIAM monographs on mathematical modeling and computation, SIAM, Philadelphia. https://doi.org/10.1137/1.9780898718300

Sen MK, Stoffa PL (1996) Bayesian inference, Gibbs' sampler and uncertainty estimation in geophysical inversion. Geophys Prospect 44(2):313–350. https://doi.org/10.1111/j.1365-2478.1996.tb00152.x