

# Simulated Annealing (Recozimento Simulado)

UFMG

10/11/2025

- 1 Simulação de Recozimento
  - Introdução
  - Processo de Recozimento
  - Aplicabilidade
  - Vantagens e Desvantagens
  - Teoria
  - Algoritmo Geral de Recozimento Simulado
- 2 Implementação
- 3 Resultados
- 4 Referências Bibliográficas

# Recozimento Simulado - Introdução

Também chamado de método meta-heurístico, este tem como intuito resolver problemas de otimização de grande complexidade. Este algoritmo foi introduzido por volta de 1980 por três pesquisadores, sendo eles: Kirkpatrick, Gelatt e Vecchi. O método traz soluções sub-ótimas, ou seja, eficientes e sem grande esforço computacional.

Tem como objetivo encontrar resultados satisfatórios, ainda que não exatos, sendo que muitas vezes estes resultados são inviáveis e/ou difíceis de serem alcançados.

Se enquadra na classe *Markov chain Monte Carlo* (MCMC) e, por ter caráter estocástico alinhado à sua capacidade de escapar de mínimos locais, este é um ótimo candidato para resolução de problemas complexos de otimização.

# Processo de Recozimento

O conceito foi introduzido da ideia de recozimento, onde um sólido é levado para um estado de baixa energia após aumentar sua temperatura.

O processo consiste em duas etapas, sendo elas:

- "Derretimento" da sua estrutura ao levá-lo a uma temperatura muito elevada.
- Resfriamento esquematizado buscando atingir um estado sólido de energia mínima.

Uma vez em estado líquido as partículas do material exposto são distribuídas de forma aleatória.

O estado mínimo de energia só é alcançado com uma temperatura inicial que seja suficientemente alta e um tempo de resfriamento que seja logo o suficiente.

O algoritmo **Simulated Annealing** tem aplicabilidade em várias áreas e aqui serão expostas algumas delas:

- Engenharia de software
- *Machine Learning*
- Teoria de filas
- Processos de manufatura
- logística e transporte com otimização

# Vantagens e Desvantagens

## Vantagens

- Pode lidar com modelos não lineares complexos, dados caóticos e desordenados, sendo uma técnica robusta no geral;
- É flexível para aproximar do ótimo global;
- O algoritmo é considerado versátil por não se prender às propriedades do modelo;
- O método é facilmente ajustado ("tunado"), ou seja, buscando melhorar seu desempenho.

## Desvantagens

- Necessita de muitas escolhas (e parâmetros) para torná-lo um algoritmo real;
- O trabalho para adaptar (tailoring) e ajustar os parâmetros do algoritmo pode ser bastante delicado;
- A precisão dos números usados na implementação do SA pode ter um efeito significativo na qualidade do resultado do método;
- Não é um algoritmo simples e convencional a ser implementado.

O estudo será centrado em estimar os parâmetros primeiramente de uma distribuição Weibull com a seguinte função densidade:

$$f(x) = \frac{\beta}{\eta} \left( \frac{x - \gamma}{\eta} \right)^{\beta-1} e^{\left(-\frac{x-\gamma}{\eta}\right)^{\beta}}; \beta > 0, \eta > \gamma \geq 0. \quad (1)$$

Tenha ciência também que sua acumulada tem o seguinte resultado:

$$F(x) = 1 - e^{\left(-\frac{x-\gamma}{\eta}\right)^{\beta}},$$

sendo  $\beta$ ,  $\eta$ ,  $\gamma$  os parâmetros de forma, escala e locação, respectivamente.

A distribuição Weibull tri-paramétrica é raramente usada devido à dificuldade de se estimar estes parâmetros. Portanto, o uso de SA será feito para tentar estimar estes parâmetros. Aqui o algoritmo será fundamental para maximizar a função de verossimilhança.



# Estimação por EMV

Temos ciência que a função de máxima verossimilhança é descrita da seguinte forma:

$$L(x) = \prod_{i=1}^n f_{x_i}(x_i|\theta)$$

Com isso, a log-verossimilhança é a seguinte:

$$\text{Ln}(L(x_1, \dots, x_n, \beta, \eta, \gamma)) = n \text{Ln} \left( \frac{\beta}{\eta} \right) + \sum_{i=1}^n \left( - \left( \frac{x_i - \gamma}{\eta} \right)^{\beta} + (\beta - 1) \text{Ln} \left( \frac{x_i - \gamma}{\eta} \right) \right). \quad (2)$$

Para encontrar as estimativas dos parâmetros devemos realizar as derivadas parciais e igualar a zero, da seguinte forma:

$$\frac{\partial}{\partial \theta} \text{Ln}(L(x)) = 0$$

Fica evidente que esta é uma função bem difícil de derivar, necessitando derivar gradiente ou mesmo aplicar métodos numéricos, portanto prosseguiremos para a ideia geral do algoritmo SA

A principal vantagem deste método perante os demais é fato de que este, ao se empregar uma busca aleatória, **não se prende a pontos de mínimos locais**. O algoritmo aceita mudanças que aceitam a diminuição e também o aumento da função objetivo  $f$ . O seu aumento é aceito com probabilidade:  $p = e^{-\frac{\Delta}{T}}$  sendo  $\theta$  o aumento e  $T$  o parâmetro de controle, analogamente conhecido como temperatura do sistema. A sua implementação é simples:

- Uma representação de soluções possíveis,
- Um gerador de mudanças aleatórias nas soluções,
- Um meio de avaliar as funções do problema, e
- Um esquema de resfriamento (annealing schedule) – uma temperatura inicial e regras para diminuí-la à medida que a busca progride.

Por ser um algoritmo meta-heurístico, urge a importância de o mesmo ter a possibilidade de aceitar uma solução pior, evitando que o algoritmo caia em um máximo local e permaneça por lá. Claramente a probabilidade de aceitar uma pior solução diminui na mesma medida que a temperatura cai em cada um dos outros ciclos. A performance do algoritmo portanto é definida dentro de parâmetros de controle, sendo eles:

- A temperatura inicial ( $T_0$ ) deve ser grande o suficiente para que na primeira iteração do algoritmo, a probabilidade de aceitação de um cenário ruim seja, pelo menos 80%;
- É utilizada a função de redução, sendo esta geométrica compreendida por:  
 $T_i = CT_{i-1}$ , onde  $C < 1$ , constante e se encontra usualmente entre 0.75 e 0.95;
- O tamanho de cada nível de temperatura determina o número de soluções geradas em uma certa temperatura  $T$ ;
- O critério de parada define quando o sistema encontrou o nível de energia desejado. Em suma, isso define:
  - O total de números de soluções geradas.
  - A temperatura a cada nível de energia desejado.
  - A razão de aceitação (entre o número de soluções aceitas e geradas).

# Algoritmo SA para Maximização da Verossimilhança

Estimativa de  $\beta, \eta, \gamma$

Passo	Ação / Descrição
1	<b>Obter Amostra</b> aleatória (tamanho grande o suficiente).
2	<b>Determinar Parâmetros de Controle</b> ( $T_0, T_f, C, l$ ).
3	<b>Gerar Solução Inicial</b> aleatória ( $a, b, c$ ).
4	<b>Computar Verossimilhança Inicial</b> ( $L$ ) na solução ( $a, b, c$ ).
<b>Loop Externo (Resfriamento)</b>	
5	<b>Enquanto</b> $T > T_f$ : $T \leftarrow CT$
<b>Loop Interno (Equilíbrio)</b>	
6	<b>Para</b> $i = 1$ a $l$ :
6.1	<b>Gerar Vizinho:</b> Gerar $a_1, b_1, c_1$ vizinhos de $a, b, c$ .
6.2	<b>Calcular Verossimilhança Vizinha</b> ( $L_0$ ) em ( $a_1, b_1, c_1$ ).
6.3	<b>Avaliar Parâmetros (Critério de Metropolis):</b>
6.3.1	<b>Se</b> $L_0 > L$ (Melhora): $a \leftarrow a_1, b \leftarrow b_1, c \leftarrow c_1, \text{ e } L \leftarrow L_0$
6.3.2	<b>Senão</b> (Piora): Gerar $u \sim \text{Uni}(0, 1)$ .

6.3.2.1	<p><b>Se</b> <math>u &lt; e^{\frac{(L_0-L)}{T}}</math> (Aceitação Probabilística):</p> $a \leftarrow a_1, b \leftarrow b_1, c \leftarrow c_1$
(Fim do Loop Interno)	
(Fim do Loop Externo)	
7	<b>Resultado:</b> Imprimir $a, b, c$ e $L$ .
<b>Nota:</b> $a, b, c$ são estimativas de $\beta, \eta, \gamma$ . $F$ é uma função de custo/energia (ex: $-\ln(L)$ ).	

# Código implementado

```
loglik_weibull3 <- function(params, x) {  
  b <- params[1] # beta (forma)  
  g <- params[2] # eta (escala)  
  c <- params[3] # gamma (locação)  
  
  if (b <= 0 || g <= 0 || any(x <= c)) {  
    return(-Inf)  
  }  
  
  z <- (x - c) / g  
  # Fórmula Log-Verossimilhança  
  ll <- length(x) * (log(b) - log(g)) + sum((b - 1) * log(z) - (z^b))  
  return(ll)  
}
```

```
propose_neighbor <- function(curr, x, sds = c(0.1, 0.1, 0.1)) {  
  attempt <- 0  
  min_x <- min(x)  
  
  repeat {  
    attempt <- attempt + 1  
  
    b1 <- curr[1] + rnorm(1, mean = 0, sd = sds[1])  
    g1 <- curr[2] + rnorm(1, mean = 0, sd = sds[2])  
    c1 <- curr[3] + rnorm(1, mean = 0, sd = sds[3])  
  
    if (b1 <= 0) {  
      b1 <- abs(b1) + 1e-6  
    }  
    if (g1 <= 0) {  
      g1 <- abs(g1) + 1e-6  
    }  
  
    if (c1 < min_x - 1e-8) {  
      return(c(b1, g1, c1))  
    }  
  }  
}
```

```
    if (attempt > 50) {  
      sds[3] <- sds[3] * 1.5  
    }  
  
    if (attempt > 1000) {  
      return(c(b1, g1, min_x - 1e-6))  
    }  
  }  
}  
  
# Algoritmo Simulated Annealing para Maximização  
sa_weibull3 <- function(  
  x,  
  init = NULL,  
  T0 = 100,  
  Tf = 0.001,  
  cooling_rate = 0.99,  
  L = 5,  
  sds = c(0.1, 0.1, 0.1)  
) {  
  n <- length(x)
```



```
if (is.null(init)) {  
  
  init <- c(1, sd(x), min(x) - 0.1)  
}  
  
curr <- init  
curr_ll <- loglik_weibull3(curr, x)  
best <- curr  
best_ll <- curr_ll  
T <- T0  
  
#  
while (curr_ll == -Inf) {  
  curr <- propose_neighbor(curr, x, sds = sds * 10)  
  curr_ll <- loglik_weibull3(curr, x)  
  best <- curr  
  best_ll <- curr_ll  
}  
  
history <- data.frame(eval = 1, ll = curr_ll)  
eval_count <- 1
```

```
# Loop SA
while (T > Tf) {
  for (i in 1:L) {
    prop <- propose_neighbor(curr, x, sds = sds)
    prop_ll <- loglik_weibull3(prop, x)

    if (prop_ll > curr_ll) {
      curr <- prop
      curr_ll <- prop_ll
    } else {
      delta_ll <- prop_ll - curr_ll

      if (runif(1) < exp(delta_ll / T)) {
        curr <- prop
        curr_ll <- prop_ll
      }
    }

    if (curr_ll > best_ll) {
      best <- curr
      best_ll <- curr_ll
    }
  }
}
```

```

        eval_count <- eval_count + 1
        history <- rbind(history, data.frame(eval = eval_count, ll = curr_ll))
    }
    T <- cooling_rate * T
}

return(list(
    best_params = best,
    best_ll = best_ll,
    history = history,
    evals = eval_count
))
}

# Função de geração SA
run_and_save_example <- function(example_num, real_params) {
    sample_sizes <- c(2500, 1000, 500, 100)
    T0_val <- 100
    Tf_val <- 0.001
    cooling_rate_val <- 0.99
    L_val <- 5

    results_table <- data.frame(

```

```

"N" = integer(),
"beta_est" = numeric(),
"eta_est" = numeric(),
"gamma_est" = numeric(),
"LL_real" = numeric(),
"LL_est" = numeric(),
"Time_s" = numeric()
)
plot_data <- list()

# Loop pontual
for (N in sample_sizes) {
  # 1. Geração da Amostra (Weibull 3P)
  data_x <- rweibull(N, shape = real_params[1], scale = real_params[2]) +
    real_params[3]

  # LL - Parametro Real
  ll_real <- loglik_weibull3(real_params, data_x)

  # Execução SA
  start_time <- Sys.time()

  # Inicialização (Passada explicitamente para run_and_save_example)

```

```
init_params <- c(1.5, sd(data_x), min(data_x) - 0.1)

sa_result <- sa_weibull3(
  x = data_x,
  init = init_params,
  T0 = T0_val,
  Tf = Tf_val,
  cooling_rate = cooling_rate_val,
  L = L_val
)
end_time <- Sys.time()
run_time <- as.numeric(difftime(end_time, start_time, units = "secs"))

# Armazenamento
best_p <- sa_result$best_params
best_ll <- sa_result$best_ll

new_row <- data.frame(
  "N" = N,
  "beta_est" = best_p[1],
  "eta_est" = best_p[2],
  "gamma_est" = best_p[3],
  "LL_real" = ll_real,
```

```
"LL_est" = best_ll,  
"Time_s" = run_time  
)  
results_table <- rbind(results_table, new_row) #{...}  
}
```

# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 2$ ,  $\eta = 2$ ,  $\gamma = 2$ )

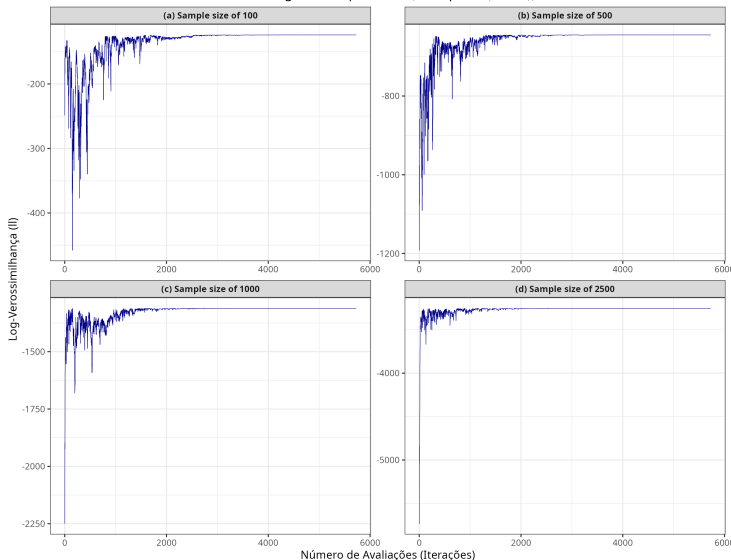
Weibull parameters		(2, 2, 2)			
Measure	2500	1000	500	100	
Estimated parameters ( $\beta$ , $\eta$ , $\gamma$ )	(1.9991, 2.0158, 2.0138)	(1.9777, 2.0351, 1.9797)	(2.1459, 2.0908, 1.9071)	(2.3615, 2.1445, 1.8567)	
Likelihood function at the real values	-3254.7346	-1311.5570	-646.9332	-125.4270	
Likelihood function at the estimated value	-3252.9872	-1311.0612	-644.8159	-124.0860	
Run time (s)	4.7858	2.9836	2.3574	2.1606	

**Table:** Tabela de Resultados para o Exemplo 1: Weibull (2, 2, 2) via Simulated Annealing

# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 2$ ,  $\eta = 2$ ,  $\gamma = 2$ )

Convergência SA para MLE (Exemplo 1: (2, 2, 2))





# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 3$ ,  $\eta = 5$ ,  $\gamma = 7$ )

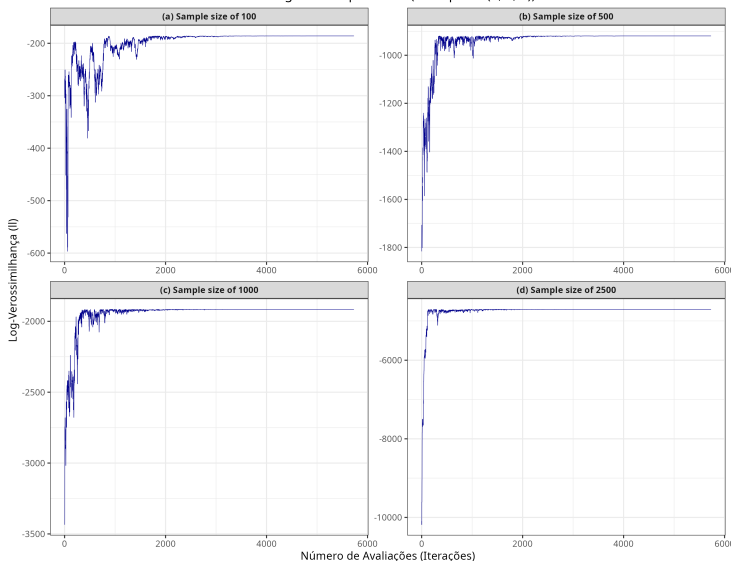
Weibull parameters		(3, 5, 7)			
Measure	2500	1000	500	100	
Estimated parameters ( $\beta, \eta, \gamma$ )	(2.9692, 4.8821, 7.0640)	(2.6933, 4.6690, 7.2710)	(2.8394, 4.5196, 7.3546)	(2.1266, 3.6834, 8.1256)	
Likelihood function at the real values	-4705.6693	-1917.4969	-922.3498	-188.0940	
Likelihood function at the estimated value	-4704.4097	-1915.2503	-919.5619	-185.9989	
Run time (s)	3.5846	2.8512	3.0746	2.8736	

**Table:** Tabela de Resultados para o Exemplo 2: Weibull (3, 5, 7) via Simulated Annealing

# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 3$ ,  $\eta = 5$ ,  $\gamma = 7$ )

Convergência SA para MLE (Exemplo 2: (3, 5, 7))



# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 8$ ,  $\eta = 4$ ,  $\gamma = 6$ )

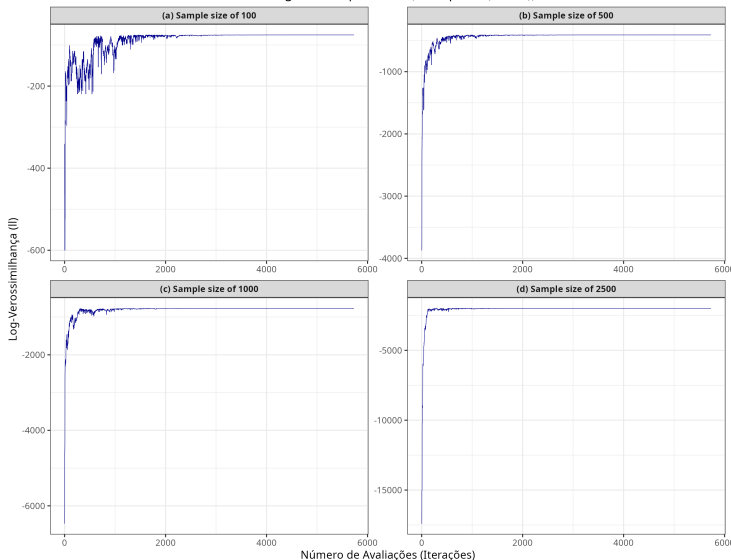
Weibull parameters		(8, 4, 6)			
Measure	2500	1000	500	100	
Estimated parameters ( $\beta$ , $\eta$ , $\gamma$ )	(7.5211, 3.7231, 6.2683)	(8.5488, 4.0806, 5.9406)	(7.6652, 3.8755, 6.1333)	(4.3496, 2.1853, 7.8769)	
Likelihood function at the real values	-1993.8436	-770.9186	-409.9350	-78.5454	
Likelihood function at the estimated value	-1992.6522	-768.5618	-409.7919	-75.4730	
Run time (s)	3.6161	3.0264	2.5860	2.3906	

**Table:** Tabela de Resultados para o Exemplo 3: Weibull (8, 4, 6) via Simulated Annealing

# Resultados da Estimação por Recozimento Simulado

Weibull ( $\beta = 8$ ,  $\eta = 4$ ,  $\gamma = 6$ )

Convergência SA para MLE (Exemplo 3: (8, 4, 6))



# Referências

- ABBasi, B.; EshrAsh Jahromi, A. H.; ARKAT, J.; Hosseinkouchack, M. **Estimating the parameters of Weibull distribution using simulated annealing algorithm**. Applied Mathematics and Computation, v. 183, p. 85-93, 2006.
- ALBERT, C.; KÜNSCH, H. R.; SCHEIDEGGER, A. **A simulated annealing approach to approximate Bayes computations**. Stat Comput, v. 25, n. 6, p. 1217–1232, 2015.
- LEIJOTO, H. D. **Avaliação de Mecanismo Probabilístico de Cooling Schedule para o Algoritmo Simulated Annealing**. Monografia de Graduação, Departamento de Estatística, Universidade Federal de Ouro Preto (UFOP), 2024.
- LUNA, A. H. **Introdução aos Métodos de Monte Carlo Avançados**. Relatório Técnico RTE\_01\_2019, Departamento de Estatística-UFMG.
- Lee, Sangmin e Seoung Bum Kim: **Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning**. IEEE Transactions on Knowledge and Data Engineering, 2019.
- Shen, Yadi, Yingchao Dong, Xiaoxia Han, Jinde Wu, Kun Xue, Meizhu Jin, GangXie e Xinying Xu: **Prediction model for methanation reaction conditions based on a state transition simulated annealing algorithm optimized extreme learning machine**. International Journal of Hydrogen Energy.
- Spinellis, D, C Papadopoulos e J M Smith: **Large production line optimization using simulated annealing**. International Journal of Production Research. 2000.