Multi-Purpose Global Optimization by flexible Simulated-Annealing

by Kai Husmann, Alexander Lange (and Elmar Spiegel)

Abstract An abstract of less than 150 words.

Introduction

As early computer-based optimization methods developed contemporaneously with the first digital computers (Corana et al., 1987), nowadays numerous optimization methods for various purposes are available (Wegener, 2005). One of the main challenges in Operations Research is therefore to match the optimization problem with a reasonable method. According to Kirkpatrick et al. (1983), optimization procedures in general can be distinguished into exact methods and heuristics. In case the loss function of the optimization problem shows quite simple responses, exact methods are often meaningful tools of choice. If all assumptions on model loss and restrictions are met, these methods will obligatorily find the exact solution without need for further parameters. The Linear Simplex-Method (Dantzig et al., 1959) as an example only needs the loss-function and optional restrictions as model input. If, however, any of the model assumptions, e. g. linearity or unimodality, is violated, exact methods are unable to solve the problems validly. In practice, they thus lack applicability whenever a loss function is too complex. With developing computer power heuristics like the Savings-Algorithm (Clarke and Wright, 1964) and metaheuristics became more and more popular. Both enable solving optimization problems of more flexible loss functions. Metaheuristics are a generalization of heuristics with aim to be even more flexible and efficient (Blum and Roli, 2003). Depending on the method of choice, more or less assumptions on the loss function can be neglected. On the other hand heuristics and metaheuristics will always solve problems approximately. Precision of the solution depends on loss, optimization method and further parameterizations of the respective method. There is even no guaranty of approximating the actual optimum since the solution also depends, by contrast to exact methods, on loss, method and parameterization (Blum and Roli, 2003). Heuristic and metaheuristic methods will thus always need additional parameters next to the loss function. Finding the proper model parameters is thus a crucial point when applying heuristic or metaheuristic algorithms. The complexity of parameterization will by trend increase with flexibility of the method. Direct search methods like the Nelder-Mead algorithm are comparatively efficient methods which directly converge to the functions optimum and need relatively less settings (Geiger and Kanzow, 1999). Random search methods are able to cope with multimodal objective functions as they have a random component which allows leaving local optima. The efficiency and accuracy of such models is usually highly sensitive to their parameter specification (Corana et al., 1987).

Simulated Annealing (SA) (Kirkpatrick et al., 1983) is known to be one of the oldest and one of the most flexible and generalized metaheuristic method (Blum and Roli, 2003). It is known to be favorable against most other methods for multimodal loss functions with a very high number of covariates (Corana et al., 1987). The method was applied in many studies among several fields covering e. g. chemistry (Agostini et al., 2006), econometrics (Ingber, 1993) or forest sciences (Baskent and Jordan, 2002). Since its first implementation by Kirkpatrick et al. (1983), many authors have modified the algorithm in order to adopt it for specific problems (e. g. DeSarbo et al., 1989; Goffe et al., 1996) or make it more efficient (e. g. Xiang et al., 2013). It is basically a combination of systematic and a stochastic components. The stochastic part of the algorithm allows keeping worse responses in each iteration step during the optimization process. This enables searching for the global optimum even for multimodal loss functions. As it offers a lot of options, SA can be named as hybrid method between a general optimizer (when default values are chosen) and a problem specific optimization program (Wegener, 2005). Corana et al. (1987) developed a dynamic adoption method for the variation of the stochastic component during the optimization process. Their modification affects the efficiency as well as the accuracy of the SA algorithm. It has potential to substantially improve the method. Pronzato et al. (1984) suggest to decrease the search-domain of the random component with increasing number of iterations. The stochastic component in general is the most sensitive part of the method since it actually determines the loss variables modification during the iterations.

The R software environment provides platform for simple and effective distribution of statistical models to a huge user community (Xiang et al., 2013). Thus, not surprisingly, several optimization packages of high quality can currently be purchased via *Comprehensive R Archive Network* (Theussl and Borchers, 2016) where even the SA method is recently listed 5 times. We, however, believe, a

package which copes with very flexible loss functions and rules for the stochastic component could be an advantageous extension for R. We present the package **optimization** where we implemented a modified version of SA. It is, as far as we know, the first function that combines SA with the extensions of Corana et al. (1987) and Pronzato et al. (1984). Main difference of our approach to existing random-search methods is its flexibility. It allows several user specifications which help to parameterize the model very problem-specific. Next to the high number of parameters that may influence accuracy and speed of the solution, remarkable novelties of our model are the ability of flexibly defining loss function as well as covariate changing rules. The user can thus directly influence the stochastic part of the algorithm. Visual post-hoc inspection of the model convergence facilitate assessment of the solution quality. We show in practical 3 examples where our model can be useful and how it can be parameterized.

Method

Simulated Annealing and its derivations are well document in the scientific literature (e. g. in Hansen, 2012; Kirkpatrick et al., 1983; Xiang et al., 2013). We thus mainly concentrate in our methods description on modified and novel implementations. The classic SA is composed of an inner for and an outer while loop (Kirkpatrick et al., 1983). Since the basic idea of Simulated Annealing is derived from the physical process of metal annealing, the nomenclature of SA comes particularly from metallurgy. The number of iterations in both loops are user predefined.

initialize t, vf with user specifications
calculate $f(x_0)$ with initial parameter vector x_0
while $t > t_{min}$ **do**
  for *i in c(1:* n_{inner}*)* **do**
    $x_j \leftarrow x_{(i-1)}$
    call the variation function to generate x_{i*} in dependence of x_j, rf and t
    check if all entries in x_{i*} are within the boundaries
    if *all* x_i *valid* **then**
      calculate $f(x_{i*})$
    else
      while *any(*x_{i*} *invalid)* **do**
        call the variation function again
        count invalid combinations
      end
    end
    if $f(x_{i*}) < f(x_j)$ **then**
      $x_i \leftarrow x_{i*}; f(x_i) \leftarrow f(x_{i*})$
    else
      calculate Metropolis Probability in dependence of t
      if *random number* $<$ *Metropolis probability* **then**
        store x_j and $f(x_j)$
        $x_i \leftarrow x_{i*}; f(x_i) \leftarrow f(x_{i*})$
      else
        $x_i \leftarrow x_j; f(x_i) \leftarrow f(x_j)$
      end
    end
    if *threshold accepting criterion fulfilled* **then**
      break inner loop
    end
  end
  reduce t for the next iteration
  vf adaptation for the next iteration
end
return optimized parameter vector, function value and some additional informations

Algorithm 1: Pseudocode of the modified Simulated Annealing method in the **optimization** package exemplary for a minimization.

The inner loop of Algorithm 1 is a Markov-Chain in which the responses of different covariate combinations are calculated and compared. The loop is repeated n_{inner} times. After saving the covariate combination of the last inner iteration as x_j, the variation function is called to create a new temporary combination x_{i*}. In the classical SA approach, the covariates are changed by a uniform distributed

random number around x_j (Kirkpatrick et al., 1983) which is also the default in our function. The variation function can be changed by the user. It is allowed to depend on a vector with random factors rf, x_j and the temperature t. By default, the entries in rf determine the lower and upper limit of the uniformly distributed random number relative to the current expression of the covariate. A random factor of 0.1 and a covariate expression of 3 for x_j e.g. leads to a random number between 2.7 and 3.3 for x_i* if the covariate vector has one entry. If all entries in x_{i*} after their variation are within the allowed boundaries, the response is calculated. Otherwise the invalid entries of x_{i*} are drawn again until all entries are valid. According to Corana et al. (1987), the number of invalid trials can be used for dynamical adjustment of the rf. The numbers of invalid trials are thus, distinctively for each covariate, counted and stored. If the return of current variables combination $f(x_{i*})$ is better than $f(x_j)$, x_{i*} and $f(x_{i*})$ are stored. Main idea of the SA is to cope with local optima. For this, even if the return of x_{i*} is worse than the return of x_j there is a chance of keeping x_{i*}. The former optimal covariate combination is stored before it is overwritten. The likelihood of keeping worse responses decreases with decreasing temperature t thus with increasing number of outer loop iterations. The detailed calculation of the Monte-Carlo approach by Metropolis et al. (1953) can i.a. be found in Kirkpatrick et al. (1983). Storing the development of covariates and response can be help improving the performance of SA (Lin et al., 1995; Hansen, 2012). We implemented a modification of the threshold accepting strategy (Dueck and Scheuer, 1990) which enables reducing the total number of calculations. This is archived by simply calculating and storing the improvement as absolute difference of $f(x_i)$ and $f(x_j)$. If the response oscillates for user defined number of repetitions within a user defined threshold, the inner loop breaks.

Main functions of the outer loop are calling the inner loop as well as specifying t and vf for the next iteration of the outer loop. As t obligatory decreases in our derivation of SA, the number of iterations is implicitly user-prespecified by initial temperature t_0 and minimum temperature t_{min}. t is necessary for the stochastic part within the inner loop (Kirkpatrick et al., 1983). As each covariate can have its own random factor, the vector with random factors rf is of the same dimension as x_i. Dividing the counted number of invalid trials of a covariate by the total number of trials of the respective covariate gives the ratio of invalid trials distinctively for each covariate. Summing up the valid and invalid trials of all inner loop repetitions gives the mean ratio of invalid trials per outer loop repetition. According to (Corana et al., 1987), this ratio can be used to find a trade-off between accuracy on the one and size of the search-domain on the other side. They argument that if there were only valid covariate combinations, the search domain could be too small for multimodal function. They suggest ratios between 0.4 and 0.6. If any observed ratio is < 0.4 or > 0.6, the respective random factors are modified following the suggested nonlinear equation by Corana et al. (1987) such that they are dynamically adjusted for the next iteration. Pronzato et al. (1984), who developed the Adaptive Random Search method, propose a time decreasing search-domain. We combined the two methods by linearly shrinking the favorable range of ratios from 0.4-0.6 to 0.04-0.06. If the default variation function is chosen, the search-domain of the covariates reduces with increasing number of outer loop iterations.

May an additional graphical explanation be useful?

The package optimization

The function optim_sa

As the function is mainly written in C++, it requires **Rcpp**. `optim_sa` shall be able to solve very specific optimization problems, several parameters can be defined by the user. Quality of solution and speed of convergence will thus substantially depend on accurate parametrization. In the following, we will explain each parameter briefly giving hints for useful specification.

- `fun`: Loss function to be optimized. The function must depend on a vector of covariates and return one numeric value. There are no assumptions on covariates and return. They are not necessarily continuous. Missing (`NA`) or undefined (`NaN`) returns are allowed as well. Any restriction on the parameter space, e. g. specific invalid covariate values within the boundaries, can be integrated in the loss function directly by simply returning NA. Restrictions on the state space can be defined analogously. We will be more specific on this in the practical example.
- `start`: Obligatory numeric vector with initial covariate combination. It must be ensured that at least one covariate combination leads to a defined numeric response. The loss function response at the start variables combination must therefore be a defined numeric value. This might be relevant when the starting values are determined stochastically.
- `maximization`: Logical value with default `FALSE` which indicates whether the loss function is to be maximized or minimized.

- `trace`: Logical value with default `FALSE`. If `TRUE`, the last inner loop iteration of each outer loop iteration is stored as row in the trace matrix. This might help evaluating the solutions quality. However, storing interim results increases calculation time up to 10 %. Disabling `trace` can thus improve efficiency when the convergence of an optimization problem is known to be stable.
- `lower`: Numeric vector with lower boundaries of the covariates. The boundaries are obligatory since the dynamic `rf` adjustment (Corana et al., 1987; Pronzato et al., 1984) depends on the number of invalid covariate combinations.
- `upper`: Numeric vector with upper boundaries of the covariates.
- `control`: A list with optional further parameters.

All parameters in the list with `control` arguments have a default value. They are pre-parameterized for loss functions of medium complexity. `control` arguments are:

- `vf`: Function that determines the variation of covariates during the iterations. It is allowed to depend on `rf`, `temperature` and a vector of covariates of the current iteration. The variation function is a crucial element of `optim_sa` which enables flexible programming. It is (next to the loss function itself) the second possibility to define restrictions. The parameter space of the optimization program can be defined `vf`. Per default, the covariates are changed by a continuous, uniform distributed random number. It must be considered that defining specific `rf` can increase the calculation time. The default `rf` is a compiled C++ function whereas user specified `rf` must be defined as R functions. User specified are e. g. useful for optimization problems with non-continuous parameter space.
- `rf`: Numeric vector with random factors. The random factors determine the range of the random number in the variation function vf relative to the dimension of the function variables. The rf can be stated separately for each variable. Default is a vector of ones. If `dyn_rf` is enabled, the entries in `rf` change dynamically over time.
- `dyn_rf`: Logical. Default is `TRUE` `rf` change dynamically over time to ensure increasing precision with increasing number of iterations. Default is `TRUE`. `dyn_rf` ensures a relatively wide search domain at the beginning of the optimization process that shrinks over time (Corana et al., 1987; Pronzato et al., 1984). Disabling `dyn_rf` can be useful when `rf` with high performance are known. The development of `rf` is documented in the `trace` matrix. Evaluation of former optimizations with dynamic `rf` can thus help finding reasonable fixed `rf`. Self-specified `vf` may not depend on `rf`. In this cases activating `dyn_rf` makes no sense.
- `t0`: Initial temperature. Default is 1000. The temperature directly influences the likelihood of accepting worse responses thus the stochastic part of the optimization. `t0` should be adopted to the loss function complexity. Higher temperatures lead to higher ability of coping with local optima on the one but also to more time-consuming function calls on the other hand.
- `t_min`: Numeric value that determines the temperature where outer loop stops. Default is 0.1. As there is approximately no chance of leaving local optima in iterations with low temperature `t_min` mainly affects accuracy of the solution. Higher `t_min` yields to lower accuracy and less function calls.
- `nlimit`: Integer value which determines the maximum number of inner loops iterations. Default is 100. If the break criterion `stopac` is not fulfilled, `nlimit` is the exact number of inner loops repetitions.
- `r`: Numeric value that determines the reduction of the temperature at the end of the outer loop. Slower temperature reduction leads to increasing number of function calls. It should be parameterized with respect to `nlimit`. High `nlimit` in combination with low `r` lead to many iterations with the same acceptance likelihood of worse solutions. Low `nlimit` in combination with `r` near 1, by contrast, lead to continuously decreasing acceptance likelihood of worse solutions.
- k:

```
x <- 1:10
result <- myFunction(x)
```

Practical examples

Rosenbrock Function

Simple Integer Programming example. Visual plot inspection.

SVAR

Alex

Forest harvesting schedule optimization

Summary and discussion

It does not guarantee, of course, to find the global minimum, but if the function has many good near-optimal solutions, it should find one. In particular, this method is able to discriminate between "gross behavior" of the function and finer "wrinkles." First, it reaches an area in the function domain where a global minimum should be present, following the gross behavior irrespectively of small local minima found on the way. It then develops finer details, finding a good, near-optimal local minimum, if not the global minimum itself.

This section may contain a figure such as Figure 1.

This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the Instructions for Authors.

Figure 1: The logo of R.

Bibliography

F. P. Agostini, D. D. O. Soares-Pinto, M. A. Moret, C. Osthoff, and P. G. Pascutti. Generalized simulated annealing applied to protein folding studies. *Journal of computational chemistry*, 27(11):1142–1155, 2006. [p1]

E. Z. Baskent and G. A. Jordan. Forest landscape management modeling using simulated annealing. *Forest Ecology and Management*, 165(1):29–45, 2002. [p1]

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003. [p1]

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964. [p1]

A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans. Math. Softw.*, 13(3):262–280, 1987. [p1, 2, 3, 4]

G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, 7(1):58–66, 1959. [p1]

W. S. DeSarbo, R. L. Oliver, and A. Rangaswamy. A simulated annealing methodology for clusterwise linear regression. *Psychometrika*, 54(4):707–736, 1989. [p1]

G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1):161–175, 1990. [p3]

C. Geiger and C. Kanzow. Das nelder-mead-verfahren. *Numerische Verfahren zur Lösung unregestrierter Optimierungsaufgaben*, 1999. [p1]

W. L. Goffe et al. Simann: A global optimization algorithm using simulated annealing. *Studies in Nonlinear Dynamics and Econometrics*, 1(3):169–176, 1996. [p1]

J. H. Hansen. *Modellbasierte Entscheidungsunterstützung für den Forstbetrieb: Optimierung kurzfristiger Nutzungsoptionen und mittelfristiger Strategien unter Verwendung metaheuristischer Verfahren und parallelen Rechnens*. Cuvillier, Göttingen, 1. aufl. edition, 2012. [p2, 3]

L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11): 29–57, 1993. [p1]

S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983. [p1, 2, 3]

C. Lin, K. Haley, and C. Sparks. A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems. *European Journal of Operational Research*, 83(2):330–346, 1995. [p3]

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. [p3]

L. Pronzato, E. Walter, A. Venot, and J.-F. Lebruchec. A general-purpose global optimizer: Implimentation and applications. *Mathematics and Computers in Simulation*, 26(5):412–422, 1984. ISSN 03784754. doi: 10.1016/0378-4754(84)90105-8. [p1, 2, 3, 4]

S. Theussl and H. W. Borchers. CRAN Task View: Optimization and Mathematical Programming. Oct. 2016. URL https://CRAN.R-project.org/view=Optimization. [p1]

I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. In *International Colloquium on Automata, Languages, and Programming*, pages 589–601. Springer, 2005. [p1]

Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. Generalized simulated annealing for global optimization: the gensa package. *The R Journal Volume 5(1):13-29, June 2013*, 2013. [p1, 2]

Author One
Affiliation
Address
Country
author1@work

Author Two
Affiliation
Address
Country
author2@work

Author Three
Affiliation
Address
Country
author3@work