Pertanika J. Sci. & Technol. 26 (3): 1401 - 1412 (2018)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

MetaheuristicOpt: An R Package for Optimisation Based on Meta-Heuristics Algorithms

Lala Septem Riza*, Iip, Eddy Prasetyo Nugroho and Munir

Department of Computer Science Education, Universitas Pendidikan Indonesia, JL. Setiabudhi, 40154, Bandung, Indonesia

ABSTRACT

Optimisation, which is a method to obtain optimal or near-optimal values of objective functions, has been widely used to make a decision in many problem domains, such as engineering, chemical, business, etc. This research is aimed to build an R package that implements 11 methods based on meta-heuristics methods that are inspired by natural phenomena and animal behaviours. Here, R programming language is considered since it is a popular programming language for data science. In this version of the package, 11 meta-heuristic algorithms are implemented, namely particle swarm optimisation (PSO), ant lion optimizer (ALO), grey wolf optimizer (GWO), dragonfly algorithm (DA), firefly algorithm (FFA), genetic algorithm (GA), grasshopper optimisation algorithm (GOA), moth flame optimizer (MFO), sine cosine algorithm (SCA), whale optimisation algorithm (WOA), and harmony search (HS). The methods have proven to be reliable and stable. To validate the package, the study presents 13 benchmarking functions in our experiments such as sphere model, Schwefel's Problem 2.22, Generalised Rosenbrock's Function and Step Function. Based on the experiments, package metaheuristicOpt produces optimal solutions as indicated by references proposing respective algorithms.

Keywords: Meta-heuristics algorithm, optimisation, R programming language, software library, Swarm intelligence

INTRODUCTION

Humans make decision every day based on its optimal output. For example, to visit a specific place from our current location, we choose the best by considering many factors, such as distance, width of street, traffic condition, type of transportation, safety, and many others. Many complicated problems are solved by optimisation methods, such as molecular biology (Festa, 2007), electrics measurements

ARTICLE INFO

Article history:
Received: 20 October 2017
Accepted: 20 June 2018

E-mail addresses:
lala.s.riza@upi.edu (Lala Septem Riza)
iip@student.upi.edu (Lala Septem Riza)
eddypn@upi.edu (Eddy Prasetyo Nugroho)
munir@upi.edu (Munir)
*Corresponding Author

ISSN: 0128-7680

and electrical engineering (Sergeyev, Daponte, Grimaldi, & Molinaro, 1999), optimal solar sail steering (Dachwald, 2004).

Additionally, to solve various real-world problems, many researchers have proposed many methods. These algorithms can be divided into two groups according to the precision of solutions, such as exact and approximate methods (Talbi, 2009). Algorithms need a software library so that scientists and engineers can utilise them easily. These packages are very helpful for others since not every scientist and engineer has the capability to make a program and sometimes they just need to solve their problems without needing to know the methods in detail.

Therefore, this research is aimed at developing a software library (i.e., a package) that implements 11 algorithms for dealing with optimisation tasks. These algorithms are included in the approximate method, which is population based meta-heuristics. The following algorithms are considered in this paper: particle swarm optimisation (PSO), ant lion optimizer (ALO), grey wolf optimizer (GWO), dragonfly algorithm (DA), firefly algorithm (FA), genetic algorithm (GA), grasshopper optimisation algorithm (GOA), moth flame optimizer (MFO), sine cosine algorithm (SCA), whale optimisation algorithm (WOA), and harmony search (HS).

The package developed was written in R programming language (Ihaka & Gentleman, 1996). It is not only programming language but also an ecosystem that provides over 8000 packages for implementing many methods such as machine learning, natural language processing, optimisation, etc. For example, in machine-learning domain, we can find the following packages: frbs (Riza, Bergmeir, Herrera, & Benítez Sánchez, 2015), RoughSets (Riza et al., 2014), and gradDescent (Riza, Nasrulloh, Junaeti, Zain, & Nandiyanto, 2016a). For dealing with the optimisation task, we can find several packages, such as Rmalschains and DEoptim. Additionally, R programming language is considered in this research since according to a survey conducted by KDnuggets (Piatetsky, 2017), it is most popular programming language for data science in 2013, 2014, 2015, and 2016. It means the package can be possibly used by many users in data science.

The paper is organised as follows. Population based meta-heuristics algorithms is discussed in Section 2 while Section 3 explains the package architecture developed in this research. The experimental design to validate the proposed package is shown in Section 4 while Section 5 discusses results and Section 6 summarises and concludes the paper.

METHODS

Population Based Meta-Heuristics Algorithms for Optimisation

Basically, optimisation is a process to find best solutions from sets of alternatives. In other words, given an objective function (f) to be minimised (i.e., a minimisation problem) or maximised (i.e., a maximisation problem), we need to find x_0 such that $f(x_0) \leq f(x)$ for all x in sets of solutions for minimisation or such that $f(x_0) \geq f(x)$ for all x in sets of solutions in the search space for maximisation. Detailed information regarding an introduction to optimisation can be found in (Pedregal, 2006). Furthermore, there are many approaches that can be used for dealing with optimisation tasks. According to the research conducted by Talbi (2009), these methods can be divided into two kinds: exact methods and approximate methods. The approximate methods mean that optimal solutions are not guaranteed to be obtained as in

the exact algorithms. In this research, the focus is on approximate methods, especially on population based meta-heuristics.

Recently, many methods related to population based meta-heuristics have been introduced. A method that utilises meta-heuristics and selects a solution from a population can be stated as population based meta-heuristics. They are mostly inspired by natural phenomena. Figure 1 shows general pseudo code applied in population based meta-heuristic methods.

Input: Objective function (f(x))

Output: Best solution

Algorithm:

1. Generate initial population (P_0)
2. Evaluate each candidate solution in P_0
3. Select the best solution
4. **while** $t < maxIteration$
    a. Update new population (P_t) by considering defined operators
    b. Evaluate each candidate solution in P_t
    c. Update the best solution
    d. $t = t + 1$
5. **end while**

Figure 1. General pseudo code in population meta-heuristics methods

This research considers 11 algorithms as follows:

1. Particle Swarm Optimisation (PSO) (Poli, Kennedy, & Blackwell, 2007): It was proposed by Kennedy and Eberhart in 1995. In order to update a new population, the method uses equations for updating on the velocity and position of particles. An example of PSO implementations is for determining pressure distribution on water pipeline networks (Riza, Azmi, Rahman, & Sidarto, 2016b).
2. Ant Lion Optimiser (ALO) (Mirjalili, 2015a): It is inspired by smart behaviors of antlion (i.e., Myrmeleontidae) when hunting ants. In this method, random walk is used to update a new position, which is a new population.
3. Grey Wolf Optimiser (GWO) (Mirjalili, Mirjalili, & Lewis, 2014): It is inspired by behaviors of grey wolves (i.e., Canis lupus) in hunting techniques: searching, encircling, and attacking. The wolves involved are the leaders (i.e., alpha), second level (i.e., beta), and the lowest level (i.e., omega).
4. Dragonfly Algorithm (DA) (Mirjalili, 2016a): In this method, the following concepts are considered, as follows: separation, alignment, cohesion, resources, and predator, to update a new population.

5. Firefly Algorithm (FFA) (Yang, 2009): It is inspired by tropical firefly. Two essential rules are involved in these methods: to calculate attractiveness s and to evaluate new solutions and update light intensity/brightness. These methods have been used for dealing with many optimisation tasks, such as in calculating pressure distribution on water pipeline networks (Riza, Kusnendar, Hays, & Sidarto, 2016c).
6. Genetic Algorithm (GA) (Goldberg & Holland, 1988): It is a method based on genetic evolution that has several operators: mutation, crossover, and selection processes.
7. Grasshopper Optimisation Algorithm (GOA) (Saremi, Mirjalili, & Lewis, 2017): It is inspired by Grasshoppers' behaviours, such as attraction and repulsion. These behaviors are expressed by two equations: social forces and updating position.
8. Moth Flame Optimizer (MFO) (Mirjalili, 2015b): It is inspired by the moth's natural navigation techniques as seen in nature called transverse orientation. It is used to design a mathematical model of spiral flying path of moths around artificial lights (flames).
9. Sine Cosine Algorithm (SCA) (Mirjalili, 2016b): It is based on characteristics of profiles of sinus and cosines functions. So, to explore and exploit and then to generate a new population, two functions in sinus and cosines are used.
10. Whale Optimisation Algorithm (WOA) (Mirjalili & Lewis, 2016): It is inspired by the social behaviors of humpback whales, especially on the bubble-net hunting strategy. There are some concepts involved to obtain best solution, as follows: encircling prey, bubble-net attacking method (exploitation phase), and search for prey (exploration phase).
11. Harmony Search (HS) (Geem, Kim, & Loganathan, 2001): It is inspired by the improvisation of music players to produce harmony in music.

These algorithms are considered because they have solved many tasks. Additionally, these algorithms are state of the art on optimisation methods.

Metaheuristicopt: Package Architecture, User Guide, and Example

The package proposed to implement 11 methods based on meta-heuristics methods for dealing with optimisation problems is called metaheuristicOpt. It is submitted into the Comprehensive R Archive Network (CRAN), and retrieved at https://cran.r-project.org/package=metaheuristicOpt. Additionally, users can download it free. The package contains several functions where each name is defined as the abbreviation of the methods as shown in Figure 2. It can be seen there are 11 functions representing the considered methods and 1 main function, which is `metaOpt()`, as the highest level of function. It has the following signature:

```
metaOpt(FUN, optimType = "MIN", algorithm = "PSO",
numVar, rangeVar, control = list(), seed = NULL)
```

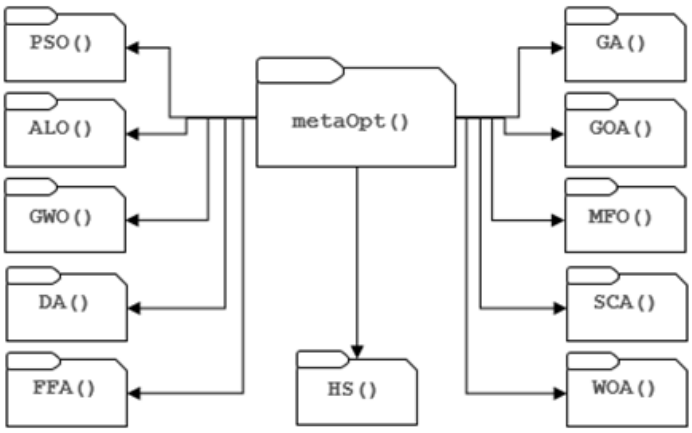

Figure 2. Functions included in the metaheuristicOpt package

So, the function has several arguments as follows:

- `FUN`: an objective function or cost function. It can be seen that the package can be used for continue problems with a single objective function.
- `optimType`: a string value that represents one of two optimisation types: "`MIN`" and "`MAX`".
- `algorithm`: a vector or single string value representing one of 11 implemented algorithms, such as "`PSO`" for Particle Swarm Optimization.
- `numVar`: a positive integer to determine the number of variables.
- `rangeVar`: a matrix (`2 x n`) containing the range of variables.
- `control`: a list containing all specific arguments of the chosen algorithm.

Detailed description regarding the arguments can be found in the manual of the package at the website.

In order to use the package metaheuristicOpt, we firstly need to install it from CRAN, and then we load it as explained in (Riza et al., 2014; Riza et al., 2015; Riza et al., 2016a). The following is an example of a user guide to run the package. For example, we need to minimise the McCormic function defined as (Surjanovic & Bingham, 2017):

Minimise: $f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$
with $-1.5 \leq x_1 \leq 4$ and $-3 \leq x_2 \leq 4$

Figure 3 shows the code showing how to execute the function `metaOpt()` and its arguments. In this case, we use GWO algorithm. The results show that the optimum value is -1.913223 at -0.5473604 and -1.54727 for x_1 and x_2.

```
R> mcCormic <- function(x){
+   F <- sin(x[1]+x[2])+(x[1]-x[2])^2-
1.5*x[1]+2.5*x[2]+1
+   return(F)
+  }
R> optimType <- "MIN"
R> algorithm <- "GWO"
R> numVar <- 2
R> rangeVar <- matrix(c(-1.5, 4, -3, 4), nrow=2)
R> control <- list(numPopulation=30, maxIter=300)
R> result <- metaOpt(mcCormic, optimType, algorithm,
numVar, rangeVar, control, seed=1)
# to show the result
R> result
$result
          var1     var2
GWO -0.5473604 -1.54727

$optimumValue
    optimum_value
GWO     -1.913223

$timeElapsed
    user system elapsed
GWO 1.31      0    1.31
```

Figure 3. An example to use the package metaheuristicOpt

Experimental Design

To validate the package, in this research we did experimentations that involved 13 benchmark functions on optimisation. These functions can be classified into two types of optimisations: unimodal and multimodal, for minimisation tasks. The following is a list of these functions:

- Unimodal: sphere model (F1), Schwefel's problem 2.22 (F2), Schwefel's problem 1.2 (F3), Schwefel's problem 2.21 (F4), generalised Rosenbrock's (F5), step function (F6), and quartic function with noise (F7).
- Multimodal: generalised Schwefel's problem 2.26 (F8), generalised Rastrigin's function (F9), Ackley's function (F10), generalised Griewank function (F11), generalised penalized function 1 (F12), and generalised penalized function 2 (F13).

All these functions can be found in Yao, Liu, & Lin, (1999). For example, F8 to F13 are shown in Figure 4, and we can see these functions are complicated and have many solutions.

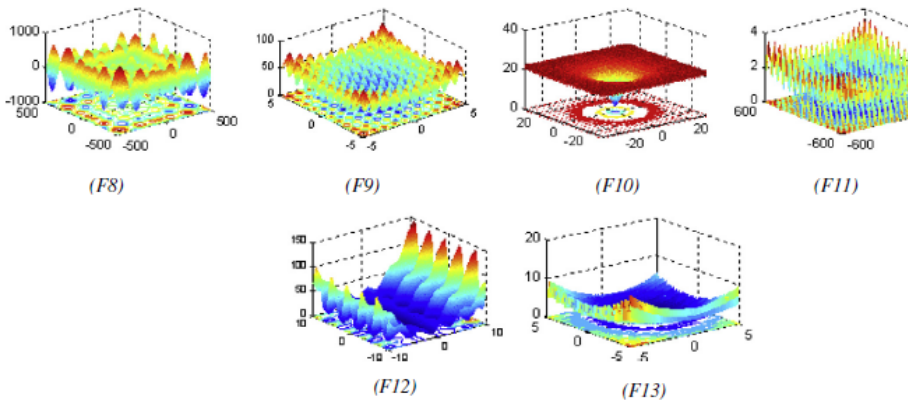

Figure 4. Multimodal benchmark function in 2D. From "Evolutionary programming made faster", by X. Yao, Y. Liu, & G. Lin, 1999, *IEEE Transactions on Evolutionary Computation, 3*(2), pp. 82-102. Copyright 1999 by IEEE Computational Intelligence Society

Furthermore, before performing simulations over all benchmark functions, we need to define all parameters required on each algorithm. In these simulations, we make some sensitivity analysis on the population size (i.e., `numPopulation`) and maximum iteration (i.e., `maxIter`) with the following values: {10, 30, 50, and 100} and {100, 500, and 1000} respectively. For other parameters, we assign the values with the default ones. So, it is obvious that we simulate 1716 times for all combinations of the algorithms, the benchmark functions and parameter values.

RESULTS AND DISCUSSION

By using scenarios based on the previous section, 1716 simulations were performed. The best solution for of each function on all the objective functions were compared and shown in Table 1. The best solution refers to values of variables that provide minimum values of the objective functions. It can be seen that PSO provides best solutions on the function F5, F6, F12, and F13 while the best solutions for F3, F4, F9, and F11 are by GWO. Additionally, best solutions of F1, F2, F8, F9, and F10 are presented by WOA and the rest are obtained by HS. In short, the algorithms mostly provide reasonable results for all benchmark functions. It should be noted that the experiments are not intended to test the algorithms, but to verify and validate that the package has been run very well when simulating the benchmarking functions. To validate the algorithms, interested readers can refer the articles proposing the corresponding algorithms.

Furthermore, comparison between the package metaheuristics and other software libraries available in CRAN is shown in Table 2. According to the table, we can state the package metaheuristicOpt offers more numbers of methods than the others. From the perspective of documentation, we have provided a comprehensive manual for common users along with many examples in the CRAN website. Lastly, the package uploaded in CRAN has the license of the GNU General Public License, so R users can also modify and improve the package freely.

Table 1
Comparison on minimum values of the objective functions of all algorithms

Benchmark Function	Algorithms										
	PSO	ALO	GWO	DA	FFA	GA	GOA	MFO	SCA	WOA	HS
F1	7.31E-25	7.06E-09	2.84E-125	7.05E-02	6.68E+02	3.26E+00	6.38E-06	4.56E-19	5.35E-16	2.37E-134	3.53E-05
F2	5.50E-14	6.68E+00	2.14E-71	2.48E+00	7.72E-01	4.22E-01	1.13E-03	3.76E-13	1.07E-08	3.29E-75	1.24E-02
F3	1.80E-17	2.25E-03	1.17E-62	3.14E+02	1.44E+03	9.96E+02	8.81E-02	5.32E-03	1.04E-04	1.23E+02	1.78E+02
F4	6.07E-11	3.10E+00	1.86E-41	2.05E+01	4.82E+01	4.52E+00	1.90E-01	7.24E+00	1.46E-05	1.98E+01	1.60E+00
F5	8.20E-02	1.63E+01	7.19E+00	2.59E+03	1.43E+03	5.92E+01	2.86E+01	7.21E-01	7.37E+00	7.18E+00	2.42E-01
F6	3.13E-25	2.76E-09	1.09E-06	3.80E+01	7.07E+02	9.82E+00	2.43E-06	9.04E-19	4.29E-01	7.74E-04	3.28E-05
F7	6.56E-01	6.19E-01	7.58E-01	9.21E-01	1.03E+00	1.03E+00	9.90E-01	9.44E-01	4.94E-01	3.54E-01	2.55E-01
F8	-1614.01	-2285.15	-2233.52	-2728.70	-2610.03	-4165.29	-2985.69	-3479.19	-2249.93	-4188.70	-4188.68
F9	6.96E+00	3.08E+01	0.00E+00	4.14E+01	7.10E+01	1.70E+00	6.77E+01	4.88E+01	2.56E-09	0.00E+00	5.34E-03
F10	1.11E-14	2.58E+00	7.55E-15	3.03E+00	8.20E+00	9.90E-02	2.32E+00	2.36E-10	2.90E-08	4.44E-16	6.34E-03
F11	8.03E-01	2.29E-01	2.45E-02	1.14E-01	1.20E+02	1.05E+00	2.23E-01	8.36E-02	4.51E-02	1.46E-01	1.18E-01
F12	1.21E-26	8.73E+00	1.57E-07	4.53E-01	2.69E+01	3.93E-01	2.58E-01	1.17E-21	1.57E-01	1.92E-02	1.05E-06
F13	1.45E-27	7.58E-08	1.54E-06	2.59E+04	2.34E+01	2.27E-01	2.11E-02	1.10E-02	5.68E-01	2.20E-02	1.60E-02

Table 2
Comparison metaheuristicOpt with other software libraries

No	Software Libraries	Numbers of Method	Algorithms	Support on Parallel Computing	References
1	MetaheuristicOpt	11	PSO, ALO, GWO), DA, FFA, GA, GOA, MFO, SCA, WOA, and HS	No	-
2	DEoptim	1	Differential Evolution	Yes	(Mullen, Ardia, Gil, Windover, & Cline, 2011)
3	hydroPSO	5	PSO and its variants: Standard PSO 2011, Standard PSO 2007, ipso, fips, and canonical PSO	Yes	(Zambrano-Bigiarini & Rojas, 2013)
4	Rmalschains	4	CMA-ES The Covariance Matrix Adaptation Evolution Strategy, SW A Solis Wets solver, SSW Subgrouping Solis Wets, Simplex	No	(Bergmeir, Molina Cabrera, & Benítez Sánchez, 2016)
5	NMOF	5	Differential Evolution, Genetic Algorithm, Local Search, Particle Swarm Optimisation, Threshold Accepting	Yes	(Gilli, Maringer, & Schumann, 2011)

CONCLUSION

The package metaheuristicOpt, providing the implementation of 11 algorithms included in population based meta-heuristics for optimisation, has been developed. It can be downloaded from CRAN at https://cran.r-project.org/package=metaheuristicOpt. Moreover, 13 benchmark functions in optimisation are used to test the package. The results show that all solutions obtained are reasonable. Therefore, the package can be used as an alternative software library for dealing with optimisation tasks. Moreover, in the future, the researchers plan to improve the package by adding other methods. Supporting parallel computing can be also considered in the next research.

REFERENCES

Bergmeir, C. N., Molina Cabrera, D., & Benítez Sánchez, J. M. (2016). Memetic algorithms with local search chains in R: The Rmalschains package. *Journal of Statistical Software, 75*(4), 1-33.

Dachwald, B. (2004). Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol. *Journal of Guidance, Control, and Dynamics, 27*(1), 66-72.

Festa, P. (2007). On some optimization problems in molecular biology. *Mathematical Biosciences, 207*(2), 219-234.

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation, 76*(2), 60-68.

Gilli, M., Maringer, D., & Schumann, E. (2011). *Numerical methods and optimization in finance.* USA: Academic Press is an imprint of Elsevier.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95-99.

Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*(3), 299-314.

Mirjalili, S. (2015a). The ant lion optimizer. *Advances in Engineering Software, 83*, 80-98.

Mirjalili, S. (2015b). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228-249.

Mirjalili, S. (2016a). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*(4), 1053-1073.

Mirjalili, S. (2016b). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120-133.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51-67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Mullen, K., Ardia, D., Gil, D., Windover, D., & Cline, J. (2011). 'DEoptim': An R package for global optimization by differential evolution. *Journal of Statistical Software, 40*(6), 1-26.

Pedregal, P. (2006). *Introduction to optimization.* New York, NY: Springer Science and Business Media.

Piatetsky, G. (2017). *New leader, trends, and surprises in analytics, data science, machine learning software poll*. KDnuggets™. Retrieved from http://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning-software-leaders.html.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence, 1*(1), 33-57.

Riza, L. S., Azmi, A. F., Rahman, E. F., & Sidarto, K. A. (2016b). Particle swarm optimization for calculating pressure on water distribution systems. In Y. Tan, Y. Shi, & B. Niu (Eds.), *Proceedings of the International Conference in Swarm Intelligence* (pp. 381-391). Bali, Indonesia: Springer International Publishing.

Riza, L. S., Bergmeir, C. N., Herrera, F., & Benítez Sánchez, J. M. (2015). FRBS: Fuzzy rule-based systems for classification and regression in R. *Journal of Statistical Software. 65*(6), 1-30.

Riza, L. S., Janusz, A., Bergmeir, C., Cornelis, C., Herrera, F., Ślezak, D., & Benítez, J. M. (2014). Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets". *Information Sciences, 287*, 68-89.

Riza, L. S., Kusnendar, J., Hays, R. N., & Sidarto, K. A. (2016c). Determining the pressure distribution on water pipeline networks using the firefly algorithm. In D. Al-Dabass, T. Achalakul, S. Prom-On, & R. Sarochawikasi (Eds.), *Proceedings of the 7th International Conference on Intelligent Systems, Modelling and Simulation* (pp. 31-36). Bangkok, Thailand: Institute of Electrical and Electronics Engineers, Inc.

Riza, L. S., Nasrulloh, I. F., Junaeti, E., Zain, R., & Nandiyanto, A. B. D. (2016a). gradDescentR: An R package implementing gradient descent and its variants for regression tasks. In *Proceedings of the International Conference on Information Technology, Information Systems and Electrical Engineering* (pp. 125-129). Yogyakarta, Indonesia: Institute of Electrical and Electronics Engineers, Inc.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30-47.

Sergeyev, Y. D., Daponte, P., Grimaldi, D., & Molinaro, A. (1999). Two methods for solving optimization problems arising in electronic measurements and electrical engineering. *SIAM Journal on Optimization, 10*(1), 1-21.

Surjanovic, S., & Bingham, D. (2015). *McCormick function.* Retrieved August 1, 2017, from https://www.sfu.ca/~ssurjano/mccorm.html.

Talbi, E. G. (2009). *Metaheuristics: From design to implementation.* Lille, Prancis: John Wiley & Sons.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe & T. Zeugmann (Eds.), International symposium on stochastic algorithms (pp. 169-178). Sapporo, Japan: Springer.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82-102.

Zambrano-Bigiarini, M., & Rojas, R. (2013). A model-independent particle swarm optimisation software for model calibration. *Environmental Modelling and Software, 43*, 5-25.