10.18605/2175-7275/cereus.v13n4p223-234

<< Recebido em: 14/11/2021 Aceito em: 25/11/2021. >>

ARTIGO ORIGINAL

REVISTA CEREUS ISSN: 2175-7275

Solving the Turbine Balancing Problem Using a Metropolis Algorithm Hybridized with the Hooke-Jeeves Method

Solucionando o problema de balanceamento de uma turbina através de um algoritmo Metropolis hibridizado com o método Hooke-Jeeves

Wagner Figueiredo Sacco[1], Ana Carolina Rios Coelho[2], Marcelo Lisboa Rocha[3]

[1] D.Sc. in Nuclear Engineering, UFRJ. Escola de Engenharia de Petrópolis, Universidade Federal Fluminense.

E-mail: wagner_sacco@id.uff.br

[2] D.Sc. in Computational Modeling, IPRJ/UERJ . Escola de Engenharia de Petrópolis, Universidade Federal Fluminense.

[3] D.Sc. in Electrical Engineering, UFRJ. Curso de Ciência da Computação/UFT e Programa de Pós-Graduação em Modelagem Computacional de Sistemas/UFT

ABSTRACT

Combinatorial optimization problems have been a great challenge for metaheuristics. One of them, the turbine balancing problem, which is NP-hard, is solved here. In order to do so, we use a Metropolis Algorithm, the Particle Collision Algorithm (PCA), hybridized with the well-known Hooke-Jeeves pattern search method. The aim of this algorithm, called Hooke-Jeeves PCA, is to perform a wide search in the solution space using a stochastic optimization method (the PCA) and then scan the promising areas with a local search technique (Hooke-Jeeves). This algorithm is favorably compared against a state-of-the-art metaheuristic, differential evolution. Our results show that Hooke-Jeeves PCA has the potential to be applied to other combinatorial optimization problems.

Keywords: Combinatorial Optimization. Metropolis Algorithm. Hybridization. Hooke-Jeeves Method. Random Keys.

RESUMO

Problemas de otimização combinatória têm sido um grande desafio para metaheurísticas. Um deles, o problema de balanceamento de uma turbina, que é NP-difícil, é resolvido neste artigo. Para fazê-lo, utilizamos um algoritmo de Metropolis, o algoritmo de colisão de partículas (PCA), hibridizado com o notório método de busca padrão de Hooke-Jeeves. A finalidade deste algoritmo, chamado Hooke-Jeeves PCA, é fazer uma busca ampla no espaço de soluções utilizando um método de otimização estocástica (o PCA) e depois explorar as áreas promissoras com uma técnica de busca local (Hooke-Jeeves). Este algoritmo tem um desempenho favorável em relação a uma metaheurística que representa o estado da arte, a evolução diferencial. Nossos resultados demonstram que o Hooke-Jeeves PCA tem potencial para ser aplicado a outros problemas de otimização combinatória.

Palavras-chave: Otimização combinatória. Algoritmo Metropolis. Hibridização. Método de Hooke-Jeeves. Random Keys.

1. INTRODUCTION

Combinatorial optimization problems (PAPADIMITRIOU, STEIGLITZ, 1998) are still interesting for researchers, as they are computationally hard to solve. Besides, there is also practical interest in these problems, since they may lead to a better use of limited resources, minimizing costs and/or maximizing returns (YANASSE, 2013).

One of these problems, which we address here, is the turbine balancing problem (MOSEVICH, 1986). This problem is NP-hard (JOHNSON, GAREY, 1979; ATALLAH, BLANTON, 2010), which, according to ATALLAH and BLANTON (2010) means “a complexity class of problems that are intrinsically harder than those that can be solved by a Turing machine in nondeterministic polynomial time. When a decision version of a combinatorial optimization problem is proven to belong to the class of NP-complete problems, which includes well-known problems such as satisfiability, traveling salesman problem, etc., an optimization version is NP-hard.”

Metaheuristics are general heuristics that can be applied to a wide variety of optimization problems. Recently, there have been many efforts in the application of metaheuristics to combinatorial optimization problems (PERES, CASTELLI, 2021). Following this trend, we apply the Particle Collision Algorithm (SACCO et al., 2006; RIOS-COELHO et al., 2010) to the above-mentioned problem.

The Particle Collision Algorithm (PCA) is a Metropolis-based algorithm (METROPOLIS et al., 1953) that was introduced as an alternative to simulated annealing (KIRKPATRICK et al., 1983). The main motivation behind the PCA was that in spite of being very powerful, the canonical simulated annealing is too sensitive to the choice of free parameters, such as, for example, the annealing schedule and initial temperature (CARTER, 1997). The PCA does not rely on user-supplied parameters other than the number of iterations for the local search phase to perform the optimality search, being thus more robust. This algorithm is loosely inspired by the physics of nuclear particle collision reactions (DUDERSTADT, HAMILTON, 1976), particularly scattering and absorption. Thus, a particle that hits a high-fitness “nucleus” is “absorbed” and explores the boundaries, which means that it generated stochastic solutions in the same region of the search space. On the other hand, a particle that hits a low-fitness region is scattered to another region of the search space. The PCA, on its canonical form or variants, has been successfully applied to many real-world optimization problems (SACCO et al., 2006; KNUPP et al., 2009; ABUHAMDAH, AYOB, 2009; RIOS-COELHO et al., 2010; DA LUZ et

al., 2011; MARTINEZ GONZÁLEZ et al., 2014; ANOCHI, CAMPOS VELHO, 2015; ANOCHI et al., 2021).

In this article, we use a hybridization of the Particle Collision Algorithm and the well-known Hooke-Jeeves local search algorithm (HOOKE, JEEVES, 1961) that was introduced by RIOS-COELHO et al. (2010). The aim of the proposed algorithm, named HJPCA, is to perform a wide search in the solution space using a stochastic optimization method (the PCA) and then scan the promising areas with a local search technique (Hooke-Jeeves). This search is performed iteratively until it reached a stopping criterion.

HJPCA is compared against a state-of-the-art stochastic optimization algorithm: differential evolution (DE; STORN, PRICE, 1997), on its canonical version and on a recent variant (SACCO et al., 2014). Citing SINGH et al. (2021): “Differential Evolution (DE) is a simple to implement population-based heuristic method used for solving optimization problems even if the function is discontinuous or non-differentiable. It is proved to have one of the fastest rates of convergence toward the optima. (...) DE has won top ranks in many IEEE CEC competitions, as it has outperformed its competitors in solving real parameter space optimization problems. DE and its variants have also been applied to solve various engineering optimization problems.”

The remainder of the paper is organized as follows. In the next section, the turbine balancing problem, the optimization method used to solve it, and the implementation and setup are presented. In section 3, the results are shown, then discussed in the following section. Finally, in Section 5, the conclusions are presented.

2. THE PROBLEM, THE OPTIMIZATION METHOD AND IMPLEMENTATION

2.1. The Turbine Balancing Problem

This NP-hard practical problem was originally proposed by MOSEVICH (1986) as a combinatorial optimization problem, but it was also formulated as a quadratic assignment problem (LAPORTE, MERCURE, 1988). Since then, it has been attacked by other researchers, using several formulations and different kinds of turbines (SINCLAIR, 1993; AMIOUNY et al., 2000; PITSOULIS et al., 2001; CHOI, STORER, 2004; SACCO et al., 2014; SACCO, HENDERSON, 2015).

In this work, we solve the case presented by MOSEVICH (1986). The problem consists in balancing the runners of a Francis hydraulic turbine. SINCLAIR (1993) gives a precise description of the problem to be solved: “A hydraulic turbine runner consists essentially of a cylinder with blades attached to its circumference. The turbine rotates as

water flows across the blades. During the manufacturing process the individual blades must be welded into place, equally spaced around the cylinder. To a better undestanding, is possible to see in Figure 1 the Francis hydraulic turbine. The problem encountered during this phase is the static balancing of the completed runner. Because of the complexity of the manufacturing process, the final weights of the blades may differ substantially. The result is an unbalanced runner. Since the runner can rotate at very high revolutions during use, it is crucial that the unbalance be as small as possible, otherwise the bearings on which the runners rotate will wear out very quickly." We must add that, according to MOSEVICH (1986), the variations in final weight mentioned above can be as great as $\pm 5\%$.

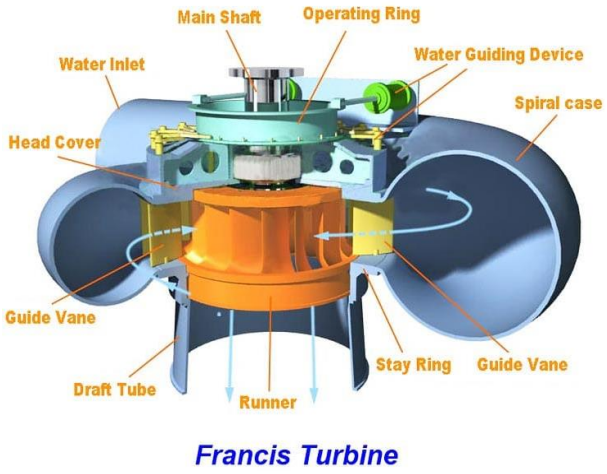

Figure 1. Example of a Francis Turbine (in orange) in a hydraulic framework.

Let us formulate the problem, following Mosevich (1986). The runner is modeled as n equally-spaced weights on a circle of zero mass and radius r equal to the common distance from the blade centers-of-mass to the runner axis. The blade positions are labeled counterclockwise, starting at position $1 = (r, 0)$ in an $x - y$ coordinate system, receiving indexes $k = 1, 2, \ldots, n$. Let P_t be a configuration of blades where $P_t(j) = k$ assigns blade k to position j. First, we define the following variables:

$M^k = \text{mass of blade } k;$

$M_j^k = \text{mass of blade } k \text{ when in position } j;$

$\theta_j = (2\pi/n)(j-1) = \text{angle between position } j \text{ and position } 1,\ j = 1, \ldots, n;$

$$M = \text{total mass of blades} = \sum_{k=1}^{n} M^k.$$

Then, each permutation P_t determines a center of mass $(\bar{x}, \bar{y})$ given by Equations (1) and (2):

$$\bar{x}(P_t) = \frac{1}{M} \sum_{j=1}^{n} M_j^k r \cos\theta_j, \tag{1}$$

$$\bar{y}(P_t) = \frac{1}{M} \sum_{j=1}^{n} M_j^k r \sin\theta_j. \tag{2}$$

Finally, Equation (3) define deviation $\bar{D}$,

$$\bar{D}(P_t) = \sqrt{\mid \bar{x}(P_t) \mid^2 + \mid \bar{y}(P_t) \mid^2}, \tag{3}$$

which is the objective function to be minimized. $\bar{D}(P_t) = 0$ means that a perfect static balance has been reached (MOSEVICH, 1986).

As suggested by Mosevich (1986), we scale the problem making $r = 1.$ We use $n = 14$ blades, as a typical runner has between 14 and 18 blades (MOSEVICH, 1986), and this value of n is one of the most difficult to optimize (LAPORTE, MERCURE, 1988). Regarding the values of M^k, we follow Laporte and Mercure (1988), generating n numbers according to a normal distribution with a mean of 100 and a standard deviation of $5/3,$ so that most M^ks fall within $\pm 5\%$ of the mean. We generated these numbers using a Gaussian Random Number Generator available at RANDOM.ORG (2012).

2.2. The Particle Collision Algorithm combined with Hooke-Jeeves (HJPCA)

The PCA algorithm resembles in its structure that of simulated annealing: first an initial configuration is chosen; then there is a modification of the old configuration into a new one. The qualities (i.e., objective function or fitness values) of the two configurations are compared. A decision is then made on whether the new configuration is “acceptable”. If it is, it serves as the old configuration for the next step (current). If it is not acceptable, the algorithm proceeds with a new change of the old configuration (trial). PCA can also be

considered a Metropolis algorithm, as a trial solution can be accepted with a certain probability. This acceptance may reduce the convergence to local optima.

A summary of PCA combined with Hooke-Jeeves (HJPCA) algorithm is given in Figure 2. The algorithm's default is for maximization problems. For minimization problems, just multiply the objective-function by -1 and invert the ratio in $p_{\text{scattering}}$.

Initialization Step

Generate an initial random solution $current$.

Main Step

Do until termination criterion is reached:

1. Generate a stochastic perturbation $trial$ of the solution $current$:

$$r = \text{rand}(0,1); U = Upper - current; L = current - Lower$$
$$trial \leftarrow current + \{[U * r] - [L * (1 - r)]$$

2. If $\text{Quality}(trial) > \text{Quality}(current)$ let $current \leftarrow trial$ and go to *HookeJeeves.* Otherwise, go to *Scattering*.

End-Do

HookeJeeves Function

1. Apply the HJ method to the solution with $\text{Quality}(current_{\text{HJ}}) = -\text{Quality}(current)$;
2. If $\text{Quality}(trial) > \text{Quality}(current)$ let $current \leftarrow trial$

return

Scattering Function

1. Calculate $p_{\text{scattering}} = 1 - [\text{Quality}(trial)/CurrentBest)$.
2. If $p_{\text{scattering}} > \text{rand}(0,1)$ let $current$ receive a random solution. Otherwise, go to *HookeJeeves*.

return

Figure 2. Algorithm of HJPCA prosed method.

The “stochastic perturbation” in the beginning of the loop consists in random variations in each variable's values within their ranges. Note that in PCA the perturbation mechanism is different from simulated annealing, where generally only a few variables are perturbed at a time (Brooks and Morgan, 1995).

If the quality or fitness of the trial solution configuration is better than the fitness of the current one, then the “particle” is “absorbed”, there is an exploitation of the boundaries searching for an even better solution. In this work, this exploitation is performed using the classical Hooke-Jeeves pattern search method, which performed better than the original scheme proposed in the canonical version of the PCA; see Rios-Coelho et al. (2010) for extensive comparisons. Note that as the Hooke-Jeeves method was conceived as a minimization algorithm, we had to change the signs of the objective-function values in function *HookeJeeves* in order to fit the PCA.

Otherwise, if the quality of the trial is worse than the current's, the “particle” is “scattered”. By scattering we mean that the new configuration receives random values between the lower and upper bounds of each variable. The scattering probability $p_{\text{scattering}}$ is inversely proportional to its fitness function value. Thus, a low-fitness particle will have a greater scattering probability. This mechanism is essential to the success of the algorithm, as it introduces a purely random component, which is responsible to avoid getting stuck at local optima.

2.3. Implementation and Setup

Our tests were performed on an Intel Core i7 PC with 8 Gb RAM running Ubuntu 14.10. Our algorithms were implemented in C++ and compiled with GNU g++. For the stochastic part of the algorithms, we used the pseudorandom number generating algorithm developed by MATSUMOTO and NISHIMURA (1998), the Mersenne Twister, which is available for download at one of its creator's website (MATSUMOTO, 2011).

For DE and its variant, Topographical Clearing Diferential Evolution (TCDE; Sacco et al., 2014), we used the following parameters: population sizes $NP = 100, 200 \text{ and } 500,$ scaling factor $F = 0.5,$ and crossover rate $CR = 0.9,$ which are commonly applied in the literature (for example, VESTERSTROM, THOMSEN, 2004; SACCO et al., 2014; SACCO et al. 2015). For TCDE, we used the number of nearest-neighbors $k = 20,$ as in SACCO et al. (2014).

The Hooke-Jeeves routine inside HJPCA was set up with Δ = 10^{-3}, ε = 10^{-6}, and α = 0.5.

Regarding the turbine balancing problem, as the PCA was conceived as a continuous optimization algorithm (SACCO et al., 2006), first, we need to adapt it for combinatorial optimization. In order to do so, we employ a representation technique named random keys (BEAN, 1994). This mechanism, originally designed for the genetic algorithm, allows us to treat discrete problems as if they were continuous. The solution is translated into a discrete sequence only in the moment of the objective function evaluation. Let us show how it works with a simplified example: a six-component combinatorial optimization problem. Our algorithm works with six continuous variables, all in the range $[0, 1]$. Let us suppose we have a solution $\mathbf{S}_1$, given by

$$\mathbf{S}_1 = (0.18, 0.73, 0.42, 0.87, 0.01, 0.23). \tag{4}$$

Each one of these variables receives an integer index, here in subscripts, corresponding to their order of appearance:

$$\mathbf{S}_1 = (0.18_1, 0.73_2, 0.42_3, 0.87_4, 0.01_5, 0.23_6). \tag{5}$$

Then, these real numbers (the so-called random keys) are sorted: $\mathbf{S}_{1_{\text{sorted}}} = (0.01_5, 0.18_1, 0.2$

The subscripts represent a valid sequence $\mathbf{S}_1^{'}$:

$$\mathbf{S}_1^{'} = (5, 1, 6, 3, 2, 4). \tag{7}$$

Note that, even in an extreme case with repeated real numbers, a valid sequence is produced:

$$\mathbf{S}_2 = (0.93, 0.27, 0.93, 0.45, 0.11, 0.93), \tag{8}$$

$$\mathbf{S}_2 = (0.93_1, 0.27_2, 0.93_3, 0.45_4, 0.11_5, 0.93_6), \tag{9}$$

$$\mathbf{S}_{2_{\text{sorted}}} = (0.11_5, 0.27_2, 0.45_4, 0.93_1, 0.93_3, 0.93_6), \tag{10}$$

$$\mathbf{S}_2^{'} = (5, 2, 4, 1, 3, 6). \tag{11}$$

As the turbine balancing problem has a known global minimum, the algorithms were run using the same termination criterion as, for example, in SIARRY et al. (1997) and RIOS-COELHO et al. (2010), which is ideal for an algorithm's performance assessment:

$$\mid f(\mathbf{x}^*) - f(\mathbf{x}) \mid \leq \varepsilon_1 \mid f(\mathbf{x}^*) \mid + \varepsilon_2, \tag{12}$$

where $f(\mathbf{x}^*)$ is the global minimum, $f(\mathbf{x})$ is the current best, $\varepsilon_1 = 10^{-4}$ corresponds to the relative error and $\varepsilon_2 = 10^{-6}$ corresponds to the absolute error (SIARRY et al., 1997).

As a safeguard, we set a maximum number of objective function evaluations equal to 10^8 for all methods tested herein as a stopping criterion, in case the condition given by Eq. (12) is not achieved.

We performed one-hundred executions for each algorithm, using one-hundred previously selected random seeds in all of them, so that the experiments are unbiased.

3. RESULTS

Table 1 compares the results obtained by DE and its variant TCDE with two population sizes, denoted by PS (not applicable to HJPCA). SR is the success rate for each algorithm and/or setup. Regarding the number of fitness evaluations (NFE), we display the minimum, maximum, and average NFEs taking into account only the successful runs. The last row refers to the relative average (RA), which was obtained dividing the average NFE (AVG) of each algorithm by the largest value achieved by them (MAX(AVG)).

Table 1. Results for the turbine balancing problem.

		DE		TCDE		HJPCA
PS		100	500	100	500	N/A
SR		10/100 (10%)	85/100 (85%)	31/100 (31%)	90/100 (90%)	100/100 (100%)
NFE	Min.	87,310	78,140	102,613	147,673	251,613
	Max.	8,280,554	46,394,433	9,268,706	48,357,305	68,511,282
	Avg.	3,296,290	14,107,599	4,137,400	18,601,053	15,633,556
	RA	0.18	0.76	0.22	1.00	0.84

4. DISCUSSION

Analyzing Table 1, one can note that HJPCA got a success rate of 100%, compared to the top results of 85% for the canonical differential evolution and 90% for its variant. This is remarkable bearing in mind that DE is a top competitor in the field (SINGH et al., 2021).

The large population size (500 individuals) required by DE and variant to obtain satisfactory results demonstrate the great complexity of the turbine balancing problem. In fact, the recommended population size for this optimization method is 100 individuals (VERSTERTROM, THOMSEN, 2004).

In terms of number of function evaluations (NFE), the difficulty of the problem is even clearer. In order to obtain a success rate of 100%, the HJPCA required an average of almost 16 million evaluations, which is 84% of the best DE (TCDE with 500 individuals) cost. In other words, Hooke-Jeeves PCA was more effective in searching the solution space than DE. Possibly the exploration provided by PCA associated with the exploitation promoted by the method of Hooke-Jeeves was responsible for that.

5. CONCLUSIONS

In this work, we applied the Hooke-Jeeves PCA, a hybridization of a Metropolis algorithm with a pattern search method, to an NP-hard combinatorial optimization problem: the turbine balancing problem.

The results obtained by this hybrid metaheuristic, better than those achieved by a state-of-the-art algorithm, differential evolution, show that this method has the potential to be applied to other combinatorial optimization problems, a class that represent some of the most challenging problems in the field (YANASSE, 2013).

As further development, we plan to hybridize the PCA with more recent and or effective local search methods (see BAZARAA et al., 2013). Another future investigation could be the application of the PCA paradigm associated with specific heuristics to the traveling salesman problem (PAPADIMITRIOU, STEIGLITZ, 1998).

REFERENCES

ABUHAMDAH, A., AYOB, M. **Multi-neighbourhood particle collision algorithm for solving course timetabling problems.** In: Data Mining and Optimization, DMO'09, 2nd Conference on, p. 21-27, IEEE, 2009.

AMIOUNY, S. V., BARTHOLDI III, J. J., VANDE VATE, J. H. Heuristics for balancing turbine fans. **Operations Research,** v. 48, p. 591-602, 2000.

ANOCHI, J. A., ALMEIDA, V. A., CAMPOS VELHO, H. F. Machine Learning for Climate Precipitation Prediction Modeling over South America**. Remote Sensing**, v. 13, p.2468, 2021.

ANOCHI, J. A., CAMPOS VELHO, H. F. **Optimization of feedforward neural network by Multiple Particle Collision Algorithm.** In: Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on, p. 128-134, IEEE, 2014.

ATALLAH, M. J., BLANTON, M. **Algorithms and Theory of Computation Handbook (Second Edition) - Special Topics and Techniques.** Boca Raton, FL: CRC Press, 2010.

BAZARAA, M. S., SHERALI, H. D., SHETTY, C. M. **Nonlinear programming: theory and**

algorithms. Hoboken, NJ: John Wiley & Sons, 2013.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. ORSA **Journal on Computing**, v. 6, p. 154-160, 1994.

CARTER, J. N. **Genetic algorithms for incore fuel management and other recent developments in optimisation.** In: Advances in Nuclear Science and Technology, p. 113-154, Springer, US, 1997.

CHOI, W., STORER, R. H. Heuristic algorithms for a turbine-blade-balancing problem. **Computers & Operations Research**, v. 31, p. 1245-1258, 2004.

DA LUZ, E. F., BECCENERI, J. C., CAMPOS VELHO, H. F. **Multiple particle collision algorithm applied to radiative transference and pollutant localization inverse problems.** In: Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW), IEEE International Symposium on, p. 347-351, IEEE, 2011.

DUDERSTADT, J. J., HAMILTON, L. J. **Nuclear Reactor Analysis.** Ann Arbor, MI: Wiley-Interscience, 1976.

GAREY, M. R., JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness.** New York, NY: W.H. Freeman and Company, 1979.

HOOKE, R., JEEVES, T. A. “Direct Search” Solution of Numerical and Statistical Problems. **Journal of the ACM** , v. 8, p. 212-229, 1961.

KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, p. 671-680, 1983.

KNUPP, D. C., SILVA NETO, A. J., SACCO, W. F. Radiative properties estimation with the Luus-Jaakola and the Particle Collision Algorithm**. *C*omputer Modeling in Engineering and Sciences (CMES)**, v. 54, p. 121-145, 2009.

LAPORTE, G., MERCURE, H. Balancing hydraulic turbine runners: A quadratic assignment problem. **European Journal of Operational Research**, v. 35, p. 378-381, 1988.

MATSUMOTO, M. **Mersenne Twister Home Page**. Available at: <http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/emt.html>. Retrieved: 09 nov. 2021.

MARTINEZ GONZÁLEZ, Y., RODRÍGUEZ, J. B. M., SILVA NETO, A. J., RODRIGUES, P. P. G. W. Estimation of open channels hydraulic parameters with the stochastic particle collision algorithm. **Journal of the Brazilian Society of Mechanical Sciences and Engineering**, v. 36, p. 69-77, 2014.

MATSUMOTO, M., NISHIMURA, T. Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on **Modeling and Computer Simulation**, v. 8, p. 3-30, 1998.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. HH., TELLER, E. Equation of state calculations by fast computing machines. **The Journal of Chemical Physics**, v. 21, p. 1087-1092, 1953.

MOSEVICH, J. Balancing hydraulic turbine runners—a discrete combinatorial optimization

problem. **European Journal of Operational Research**, v. 26, p. 202-204, 1986.

PAPADIMITRIOU, C. H., STEIGLITZ, K. **Combinatorial Optimization – Algorithms and Complexity.** Mineola, NY: Dover Publications, 1998.

PERES, F., CASTELLI, M. Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design and Development. **Applied Sciences,** v. 11, 6449, 2021.

PITSOULIS, L. S., PARDALOS, P. M., HEARN, D. W. Approximate solutions to the turbine balancing problem. **European Journal of Operational Research,** v. 130, p. 147-155, 2001.

RANDOM.ORG. **Gaussian Random Number Generator.** Available at: <https://www.random.org/gaussian-distributions/>. Retrieved: 09 nov. 2021.

RIOS-COELHO, A. C., SACCO, W. F., HENDERSON. N. A Metropolis algorithm combined with Hooke-Jeeves local search method applied to global optimization. **Applied Mathematics and Computation**, v. 217, p. 843-853, 2010.

SACCO, W. F., HENDERSON. N. **Balancing exploration and exploitation in differential evolution via variable scaling factors: An application to practical problems.** Progress in Nuclear Energy, v. 83, p. 365-373, 2015.

SACCO, W. F., HENDERSON. N., RIOS-COELHO, A. C. Topographical clearing differential evolution: A new method to solve multimodal optimization problems**. Progress in Nuclear Energy**, v. 71, p. 269-278, 2014.

SACCO, W. F., OLIVEIRA, C. R. E., PEREIRA, C. M. N. A. P. Two stochastic optimization algorithms applied to nuclear reactor core design. **Progress in Nuclear Energy**, v. 48, p. 525-539, 2006.

SIARRY, P., BERTHIAU, G., DURBIN, F., HAUSSY, J. Enhanced simulated annealing for globally minimizing functions of many continuous variables. **ACM Transactions on Mathematical Software**, v. 23, p. 209-228, 1997.

SINCLAIR, M. Comparison of the performance of modern heuristics for combinatorial optimization on real data. **Computers & Operations Research**, v. 20, p. 687-695, 1993.

SINGH, S., TIWARI, A., AGRAWAL, S. **Differential Evolution Algorithm for Multimodal Optimization: A Short Survey.** In: Tiwari, A. et al. (eds), Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing, v. 1393, Springer, Singapore, 2021.

STORN, R., PRICE, K. **Differential evolution–a simple and efficient heuristic for global Optimisation over continuous spaces. Journal of Global Optimization**, v. 11, p. 341-359, 1997.

VESTERTROM, J., THOMSEN, R. **A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems.** In: Evolutionary Computation, 2004. CEC2004. Congress on, v. 2, p. 1980-1987, IEEE, Portland, OR, 2004.

YANASSE, H. H. A review of three decades of research on some combinatorial optimization problems. **Pesquisa Operacional**, v. 33, p. 11-36, 2013.