USING THE TEACHING-LEARNING BASED OPTIMIZATION ALGORITHM IN TROUBLESHOOTING OPTIMIZATION PROBLEMS

Leonardo Ramos Rodrigues*, Takashi Yoneyama†, Roberto Kawakami Harrop Galvão†

*Electronics Division
Institute of Aeronautics and Space - IAE
São José dos Campos, São Paulo, Brazil

†Electronic Engineering Department
Aeronautics Institute of Technology - ITA
São José dos Campos, São Paulo, Brazil

Emails: leonardolrr@iae.cta.br, takashi@ita.br, kawakami@ita.br

Abstract— A troubleshooting model describes the failure modes, the repair actions and the diagnostic questions available to fix a failed component or system. A troubleshooting strategy is a sequence of repair actions and diagnostic questions that must be carried out in order to fix the failed system. In this paper, an optimization method to find a troubleshooting strategy that minimizes the expected cost of repair (ECR) is presented. Each action and question has an associated cost. Cluster costs are used to model the common preparation work shared by subsets of actions and questions. The proposed method uses the Teaching-Learning Based Optimization (TLBO) algorithm, which is a novel population oriented meta-heuristic that simulates the teaching-learning process observed in a classroom. Numerical experiments are carried out in order to illustrate the application of the proposed method in three different troubleshooting models with different complexity levels.

Keywords— TLBO, Combinatorial Optimization, Troubleshooting.

Resumo— Um modelo de solução de problemas descreve os modos de falha, as ações de reparo e as questões de diagnóstico disponíveis para consertar o componente ou sistema falhado. Uma estratégia de solução de problemas é uma sequência de ações de reparo e questões de diagnóstico que devem ser realizadas para consertar o sistema falhado. Neste artigo, um método de otimização para encontrar uma estratégia de solução de problemas que minimize o custo esperado de reparo (CER) é apresentado. Cada ação de reparo e questão de diagnóstico possui um custo associado. Custos de grupo são usados para modelar as tarefas de preparação que algumas ações de reparo e questões de diagnóstico têm em comum. O método proposto usa o algoritmo de Otimização Baseada em Ensino-Aprendizagem (OBEP), que se trata de uma meta-heurística baseada em população que simula o processo de ensino-aprendizagem observado em salas de aula. Experimentos numéricos são realizados para ilustrar a aplicação do método proposto em três modelos que possuem diferentes níveis de complexidade.

Palavras-chave— TLBO, Otimização Combinatória, Resolução de Problemas.

1 Introduction

The problem of identifying the faulty component in a complex system has become a topic of great interest for academy researchers and industry practitioners due to its potential benefits (Lín, 2014). In order to identify the root cause of a complex system failure, a maintenance investigation is commonly required. Troubleshooting is the process responsible for identifying and repairing the faulty component.

A troubleshooting problem can be solved in polynomial time under quite restrictive assumptions (Lín, 2014). When these assumptions are not valid, the problem becomes NP-hard (Vomlelová, 2003). In these situations, an alternative to find good solutions in an acceptable time is to use meta-heuristic algorithms. Many papers using different algorithms to solve troubleshooting problems can be found in the literature (Langseth and Jensen, 2001), (Jensen et al., 2001), (Vomlelová and Vomlel, 2003), (Ottosen, 2011), (Vianna et al., 2016).

In this paper, an optimization method to solve a troubleshooting problem in order to minimize the expected cost of repair (ECR) is presented. The proposed method uses the Teaching-Learning Based Optimization (TLBO) algorithm, which is a novel population oriented meta-heuristic that simulates the teaching-learning process observed in a classroom (Rao et al., 2011).

Although many papers have been recently published on applications of the TLBO algorithm, there are only a few reports concerning the use of TLBO in combinatorial problems (Baykasoğlu et al., 2014), (Dokeroglu, 2015), (Patil, 2016). Therefore, the proposed application to troubleshooting may also be regarded as a contribution to the TBLO literature.

The remaining sections of this paper are organized as follows. Section 2 describes the troubleshooting problem under consideration. Section 3 presents the basic principles of the TLBO algorithm. Section 4 presents the proposed method to solve the troubleshooting problem. Section 5 illustrates the application of the proposed method in three different troubleshooting models. Concluding remarks are presented in section 6.

2 Problem Description

In the troubleshooting optimization problem considered in this paper, a probabilistic troubleshooting model of a component or system is given. The troubleshooting model describes the failure modes, the repair actions and the diagnostic questions available to fix the component or system. The costs associated to each repair action and diagnostic question are also provided.

Some actions and questions may share a common initialization or preparation work. For instance, it may be necessary to disassemble part of the system in order to perform some of the actions and questions. In order to model these situations, the troubleshooting model also contains clusters, which are subsets of actions and questions of the model. To perform an action or question belonging to a cluster, the cost of the cluster is incurred. Once the cost of the cluster is paid, any actions and questions belonging to that cluster can be performed (Langseth and Jensen, 2001).

Assuming that the system is failed, the problem consists in finding a sequence of actions and questions to fix the faulty system in order to minimize the expected cost of repair (ECR). Each sequence of actions and questions carried out to solve a troubleshooting problem is called a troubleshooting strategy, which will be denoted by S. Figure 1 shows an example of a troubleshooting model containing all the basic elements: Failure modes (F_1, F_2, F_3 and F_4), repair actions (A_1, A_2 and A_3), a diagnostic question (Q_1), the associated costs ($C(A_1)$, $C(Q_1)$, $C(A_2)$ and $C(A_3)$) and a cluster cost ($C(K_1)$).

Failure modes are the ways in which the system might fail. In this paper, it is assumed that there is one and only one failure mode present in the system. Actions have two possible outcomes: either the system is fixed after the action has been performed, or it remains failed. Questions do not fix the system, but isolate a subset of failure modes. Some failure modes may require more than one action to be performed in order to fix the system.

For instance, consider the troubleshooting model shown in Figure 1. In order to fix a failure caused by failure mode F_1, it is sufficient to perform action A_1. However, in order to fix a failure caused by failure mode F_4, both actions A_2 and A_3 must be performed. The execution of question Q_1 makes it possible to know whether the failure mode which caused the system failure belongs to $\{F_1,F_2\}$ or not. When question Q_1 or action A_2 are performed, the cost of cluster K_1, denoted by $C(K_1)$, is incurred.

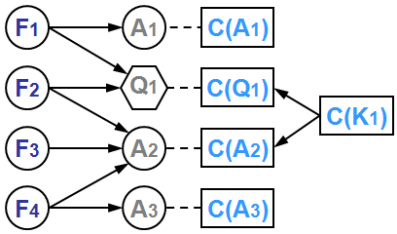

Figure 1: Example of a troubleshooting model

A troubleshooting strategy S is evaluated by its Expected Cost of Repair (ECR), which is defined according to Equation (1):

$$ECR(S) = \sum_{i=1}^{N_f} p_i \cdot C_i(S) \tag{1}$$

where N_f is the number of failure modes, p_i is the probability associated with failure mode i and $C_i(S)$ is the cost to fix a failure caused by failure mode F_i using troubleshooting strategy S, calculated according to Equation (2):

$$\begin{aligned} C_i(S) &= \sum_{j=1}^{N_a} C(A_j) \cdot \alpha_j(S) + \\ &+ \sum_{v=1}^{N_q} C(Q_v) \cdot \beta_v(S) + \\ &+ \sum_{z=1}^{N_k} C(K_z) \cdot \gamma_z(S) \end{aligned} \tag{2}$$

where N_a, N_q and N_k are the number of actions, questions and clusters in the troubleshooting model, respectively. The terms $C(A_j)$, with $j \in \{1,\dots N_a\}$, $C(Q_v)$, with $v \in \{1,\dots N_q\}$ and $C(K_z)$, with $z \in \{1,\dots N_k\}$ are costs associated to each action, question and cluster, respectively. Moreover, $\alpha_j(S)$, $\beta_v(S)$ and $\gamma_z(S)$ are binary variables that assume value 1 if the associated action, question or cluster cost is incurred and zero otherwise.

For instance, consider that the system described by the troubleshooting model presented in Figure 1 is failed due to failure mode F_4. Also, consider that troubleshooting strategy $S = \{A_2, Q_1, A_1, A_3\}$ is used to fix the system. In this example, the first step is to perform action A_2. Costs $C(A_2)$ and $C(K_1)$ are then incurred. The system is still failed, so the next step is to perform question Q_1. Cost $C(Q_1)$ is incurred. Since cluster K_1 was already activated by action A_2, no additional cost is incurred in this step. The execution of question Q_1 provides the information that the failure mode which caused the system failure belongs to $\{F_3,F_4\}$. Following the sequence of troubleshooting strategy S, the next step would be to perform action A_1. However, action A_1 can fix a failure caused by failure mode F_1 only, and it is known that the failure mode which caused the system failure belongs to $\{F_3,F_4\}$. For this reason, action A_1 is skipped and the next step becomes to perform action A_3, which fixes the

system, with cost $C(A_3)$. The total cost associated to this troubleshooting strategy to fix a failure caused by failure mode F_4 is then given by $C_4(S) = C(A_2) + C(K_1) + C(Q_1) + C(A_3)$.

3 Teaching-Learning Based Optimization Algorithm

The TLBO algorithm has been recently proposed in the literature as a novel population oriented meta-heuristic based on the teaching-learning process observed in a classroom (Rao et al., 2011). This algorithm simulates the influence of a teacher on the output of a group of students in a class. The TLBO algorithm is divided into two main parts: the Teacher Phase and the Student Phase, which is also known as the Learner Phase (Rao and Patel, 2013). During the Teacher Phase, students learn from the teacher, while in the Learner Phase students learn through the interaction among themselves.

There is a solution X associated with each student, which corresponds to a possible solution to the optimization problem under consideration. Also, there is a result $f(X)$ associated with each solution (or student), which can be obtained by evaluating the solution X using the objective function f. In the troubleshooting problem considered in this paper, a solution X corresponds to a troubleshooting strategy S and the result $f(X)$ corresponds to $ECR(S)$.

The student with the best solution in the population is considered as the Teacher. The steps for implementing the TLBO algorithm are presented in Figure 2 (Rao et al., 2011). The Teacher Phase and the Student Phase are described in the next sections.

3.1 Teacher Phase

In this phase, the algorithm simulates the learning of the students from the teacher (best solution). During this phase, the teacher makes an effort to increase the mean result of the class.

Consider a group of n students. Let M_i be the mean solution of the students and T_i be the teacher at iteration i. The teacher T_i will make an effort to move M_i to its own level. Knowledge is gained based on the quality of the teacher and the quality of students in the class. The difference D_i between the solution of the teacher, X_{Ti}, and the mean solution of the students, M_i, can be expressed according to Equation (3):

$$D_i = r_i(X_{Ti} - T_F \cdot M_i) \tag{3}$$

where r_i is a random number in the range $[0, 1]$ for iteration i and T_F is a teaching factor for iteration i, which is randomly set to either 1 or 2 according to Equation (4):

$$T_F = round(1 + rand(0,1)) \tag{4}$$

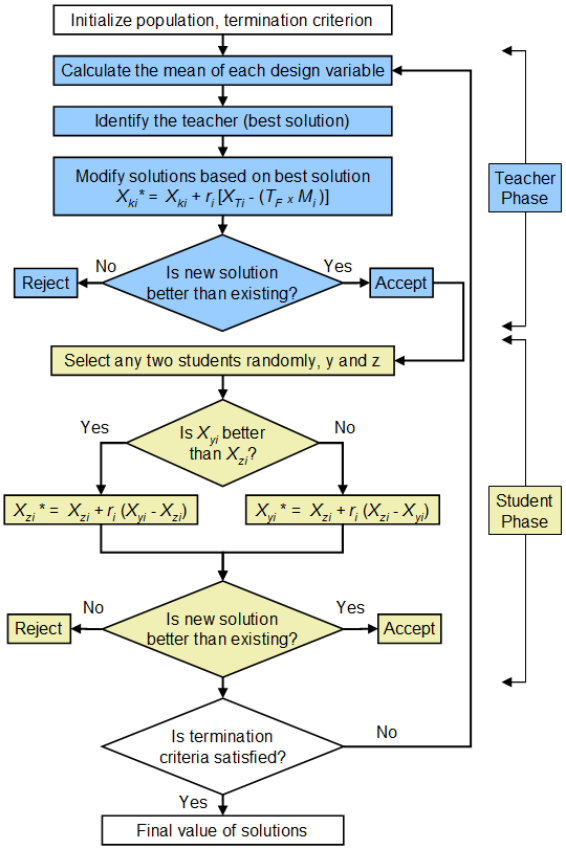

Figure 2: TLBO algorithm

Based on the difference D_i, the existing solution of student k in iteration i, X_{ki}, with $k \in \{1,2,\ldots,n\}$, is updated in the teacher phase according to Equation (5):

$$X_{ki}^{\star} = X_{ki} + D_i \tag{5}$$

where $X_{ki}^{\star}$ is the updated value of X_{ki}.

If $f(X_{ki}^{\star})$ is better than $f(X_{ki})$, $X_{ki}^{\star}$ is accepted and replaces X_{ki}. Otherwise, $X_{ki}^{\star}$ is discarded.

3.2 Student Phase

In this phase, the algorithm simulates the learning of the students through interaction with one another. During this phase, students gain knowledge by discussing with other students who have more knowledge (Rao and Patel, 2013).

Consider a pair of students y and z. Let X_{yi} and X_{zi} be the solutions of students y and z at iteration i, respectively. If $f(X_{yi})$ is better than $f(X_{zi})$, the solution of student z is updated according to Equation (6). Then, $X_{zi}^{\star}$ will replace X_{zi} if $f(X_{zi}^{\star})$ is better than $f(X_{zi})$.

Similarly, if $f(X_{zi})$ is better than $f(X_{yi})$, the solution of student y is updated according to Equation (7). Then, $X_{yi}^{\star}$ will replace X_{yi} if $f(X_{yi}^{\star})$ is better than $f(X_{yi})$.

$$X_{zi}^{\star} = X_{zi} + r_i(X_{yi} - X_{zi}) \tag{6}$$

$$X_{yi}^{\star} = X_{yi} + r_i(X_{zi} - X_{yi}) \tag{7}$$

At the end of each iteration, the stop criteria are checked. Different stop criteria may be adopted. Some of the most commonly adopted stop criteria are the maximum number of iterations, the maximum number of successive iterations without any improvement and the maximum simulation time.

In this paper, the the maximum number of iterations is adopted as the stop criterion, as in other applications of TLBO in combinatorial problems (Baykasoğlu et al., 2014), (Patil, 2016).

4 Proposed Optimization Method

In this section, the representation used to express the solution of each student is presented. The solution representation adopted in this paper is based on the priority vectors concept used by Baykasoğlu et al. (2014).

Let $X_k = \{x_k(1), x_k(2), \ldots, x_k(N_a+N_q)\}$ be a real-number vector that represents the solution of student k, where $x_k(j)$ denotes the priority value of the j-th activity (action or question) in the solution of student k. Each priority value x_k is initialized using Equation (8):

$$x_k = LB + rand[0,1](UB - LB) \qquad (8)$$

where $rand[0,1]$ is a random value uniformly distributed between zero and one, LB is the lower bound for the initial value of x_k and UB is the upper bound for the initial value of x_k.

In order to evaluate the solution, the priority vector must be converted into a troubleshooting strategy S. This conversion is made by sorting the actions and questions in decreasing order of the priority values in vector X_k. Once the troubleshooting strategy S is obtained, the result of the student, $ECR(S)$, is calculated using Equation (1).

To illustrate the conversion of a priority vector to a troubleshooting strategy, consider a troubleshooting model containing 4 actions and 2 questions. The priority vector of student k, X_k, is presented in Table 1. The largest value of X_k (8.40) is associated to question Q_1, which will be the first activity to be performed in the solution of student k. Following a decreasing order of priority, the troubleshooting strategy S of student k will be $S = \{Q_1, A_1, Q_2, A_4, A_3, A_2\}$.

Table 1: Conversion of a priority vector to a troubleshooting strategy

j	1	2	3	4	5	6
A/Q	A_1	A_2	A_3	A_4	Q_1	Q_2
$x_k(j)$	5.47	1.38	1.49	2.57	8.40	3.54

5 Numerical Experiments

This section presents the results obtained in numerical experiments carried out to illustrate the application of the proposed method in three different troubleshooting models. Table 2 summarizes the main characteristics of each model used in the experiments, which are similar to those employed in a previous work involving the use of a simulated annealing strategy for optimization (Vianna et al., 2016). The topology of Models 1, 2 and 3 are shown in Figures 3, 4 and 5, respectively.

Table 2: Main characteristics of troubleshooting models

	Model 1	Model 2	Model 3
Failures	6	10	15
Actions	5	10	16
Questions	2	4	6
Clusters	2	4	6

Table 3 presents the probability associated with each failure mode in each troubleshooting model. Table 4 shows the costs associated with each action, question and cluster.

5.1 Parameter Settings

The performance of meta-heuristic algorithms is highly dependent on parameter values. For the TLBO algorithm, the parameters to be defined are the population size, PS, and maximum number of generations, GN. In the absence of a mathematical model to determine the optimal values for the algorithm parameters, they are commonly determined empirically. In this paper, a design of experiments (DOE) approach was used to find out good settings for the parameters. The DOE is an investigative method used to evaluate the effect of multiple factors upon a process (Montgomery, 2005).

Troubleshooting model 2 was used in the DOE approach because it has an intermediate complexity level. Five different possible values were considered for GN: 100, 200, 300, 400 and 500. Four different values were considered for PS: 15, 30, 45 and 60. A full factorial experimental layout was used. For each combination, 30 runs were carried out.

In order to determine which parameter effects are significant, a statistical analysis of variance (ANOVA) was carried out. The results are presented in Table 5. Each of the main effects and interactions are considered significant at the level of 5%. In this table, df are the degrees of freedom, SS is the sum of squares, MS is the mean square, F is the F-test statistics and p is the probability value used to test the null hypothesis that a parameter effect is not significant.

Table 3: Failure modes probabilities for each model

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
Model 1	10%	5%	20%	30%	15%	20%	-	-	-	-	-	-	-	-	-
Model 2	1%	5%	10%	15%	8%	4%	13%	17%	16%	11%	-	-	-	-	-
Model 3	1%	5%	10%	15%	2%	4%	13%	14%	1%	11%	2%	7%	10%	1%	4%

Table 4: Actions, questions and clusters costs

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$C(A_i)$	10	20	30	40	50	40	35	5	20	15	20	30	70	10	20	15
$C(Q_i)$	5	10	30	10	30	20	-	-	-	-	-	-	-	-	-	-
$C(K_i)$	10	40	20	50	10	30	-	-	-	-	-	-	-	-	-	-

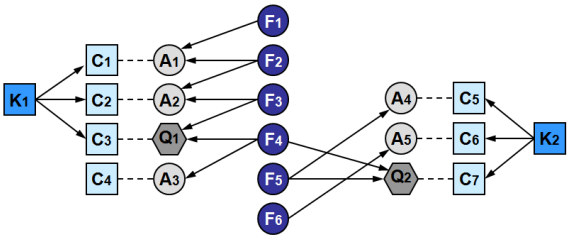

Figure 3: Troubleshooting model 1

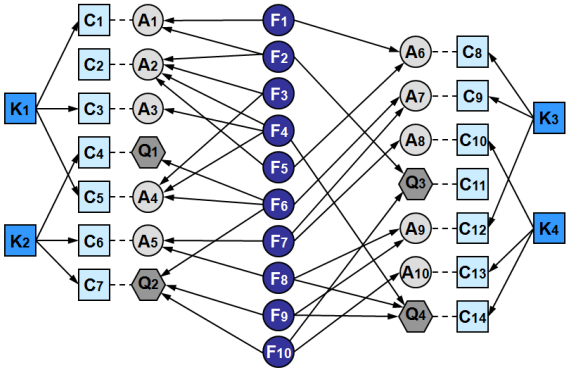

Figure 4: Troubleshooting model 2

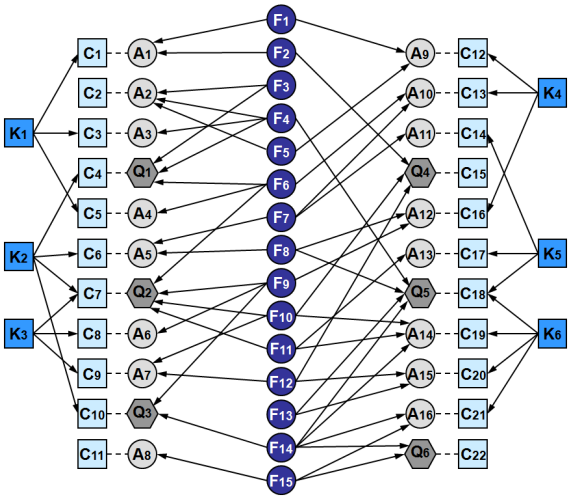

Figure 5: Troubleshooting model 3

Table 5: ANOVA results

Source	SS	df	MS	F	p
GN	11.37	4	2.84	3.81	0.0318
PS	34.07	3	11.36	15.22	0.0002
Error	8.95	12	0.75		
Total	54.39	19			

Tables 6 and 7 show the average ECR computed for each value of GN and PS, respectively. On the basis of these results, the final values for GN and PS were chosen as 300 and 60, respectively.

Table 8 shows the results for all troubleshooting models using the chosen values for GN and PS. For each model, 30 runs were carried out. In this table, BKS is the Best Known Solution for each model. For troubleshooting models 1 and 2, an exhaustive search was carried out in order to find the optimal solution. For troubleshooting model 3, however, performing an exhaustive search would be unfeasible due to the number of possible solutions. Although the BKS for model 3 is not guaranteed to be the optimal solution, extensive simulations were carried out with model 3 in order to provide confidence that its BKS is a high quality solution.

Table 6: Average ECR for each GN value

GN	100	200	300	400	500
Avg. ECR	203.1	202.8	201.3	201.6	201.5

Table 7: Average ECR for each PS value

PS	15	30	45	60
Avg. ECR	204.1	202.1	201.0	200.9

6 Conclusions

A method for solving troubleshooting optimization problems based on the TLBO algorithm was proposed.

Table 8: Simulation results

	Model 1	Model 2	Model 3
Best	78.50	199.00	206.10
Worst	80.50	214.95	232.10
Avg.	78.62	200.84	213.39
Std. dev.	0.45	3.37	6.83
BKS	78.50(*opt.*)	199.00(*opt.*)	206.10

The gaps between the average solution obtained with the proposed method and the BKS for models 1 and 2 were small: 0.15% and 0.92%, respectively. For troubleshooting model 3 the gap was larger, 3.54%. However, this result was achieved after 36,000 function evaluations only. Considering the whole search space of 1.124×10^{21} possible solutions for troubleshooting model 3, it represents a very small portion of the search space.

The proposed method may be used to update the recommended troubleshooting strategy for a system based on failure data history collected from the field. The best troubleshooting strategy in terms of ECR is highly dependent on the probability associated with each failure mode. Many factors may affect these probabilities during system lifetime such as changes in the operational pattern, maintenance interventions and environmental issues. Detecting those changes and update the recommended troubleshooting strategy accordingly can bring benefits in terms of repair cost and system availability.

In this paper, the TLBO algorithm was used in its basic version. Some changes in the basic TLBO algorithm can be made in order to improve its performance. One possible suggestion for future research is to develop algorithms to generate the initial population in order to cover a larger portion of the search space, instead of using a random population generation. Another suggestion for future investigation is to set the population size as a function of the model complexity.

Acknowledgments

This work was supported by FAPESP (grant 2011/17610-0) and CNPq (research fellowships 303714/2014-0, 303450/2013-4).

References

Baykasoğlu, A., Hamzadayi, A. and Köse, S. Y. (2014). Testing the performance of teaching learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases, *Information Sciences* **276**: 204–218.

Dokeroglu, T. (2015). Hybrid teaching-learning-based optimization algorithms for the quadratic assignment problem, *Computers & Industrial Engineering* **85**: 86–101.

Jensen, F. V., Kjærulff, U., Kristiansen, B., Langseth, H., Skaanning, C., Vomlel, J. and Vomlelová, M. (2001). The SACSO methodology for troubleshooting complex systems, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **15**(4): 321–333.

Langseth, H. and Jensen, F. V. (2001). Heuristics for two extensions of basic troubleshooting, *Frontiers in Artificial Intelligence and Applications* **66**: 80–89.

Lín, V. (2014). Decision-theoretic troubleshooting: Hardness of approximation, *International Journal of Approximate Reasoning* **55**: 977–988.

Montgomery, D. C. (2005). *Design and Analysis of Experiments*, 6th edn, John Wiley & Sons, Inc.

Ottosen, T. J. (2011). *Solutions and Heuristics for Troubleshooting with Dependent Actions and Conditional Costs*, PhD thesis, Aalborg University.

Patil, A. B. (2016). Teaching learning based optimization: Application and variation, *Proceedings of the International Conference on Computing, communication and Energy Systems*, Sangli, pp. 1–5.

Rao, R. V. and Patel, V. (2013). An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems, *Scientia Iranica* **20**: 710–720.

Rao, R. V., Vakharia, D. P. and Savsani, V. J. (2011). Teaching-learning-based optimization:a novel method for constrained mechanical design optimization problems, *Computer-Aided Design* **43**: 303–315.

Vianna, W. O. L., Rodrigues, L. R., Yoneyama, T. and Mattos, D. I. (2016). Troubleshooting optimization using multi-start simulated annealing, *10th Annual IEEE Systems Conference, Orlando, FL, USA, 18-21 April 2016.*

Vomlelová, M. (2003). Complexity of decision-theoretic troubleshooting, *International Journal of Intelligent Systems* **18**(2): 267–277.

Vomlelová, M. and Vomlel, J. (2003). Troubleshooting: NP-hardness and solution methods, *Soft Computing* **7**(5): 357–368.