PARAMETER TUNING OF THE TEACHING-LEARNING BASED OPTIMIZATION ALGORITHM APPLIED TO TROUBLESHOOTING

Lucas Sousa de Oliveira
Instituto Tecnológico de Aeronáutica - ITA
Praça Marechal Eduardo Gomes, 50, São José dos Campos, SP, Brasil, 12228-900
lsoliveira459@gmail.com

Takashi Yoneyama
Instituto Tecnológico de Aeronáutica - ITA
Praça Marechal Eduardo Gomes, 50, São José dos Campos, SP, Brasil, 12228-900
takashi@ita.br

Leonardo Ramos Rodrigues
Instituto de Aeronáutica e Espaço - IAE
Praça Marechal Eduardo Gomes, 50, São José dos Campos, SP, Brasil, 12228-904
leonardolrr@iae.cta.br

ABSTRACT

Teaching-Learning Based Optimization (TLBO) is a novel metaheuristic, population-based optimization method that simulates the teaching environment. It has recently shown great results in different applications due to its efficiency and accuracy. Parameter tuning is recognized as a crucial step when designing an optimization algorithm. The performance of optimization algorithm is very sensitive to parameter values. Therefore, finding a good parameter setting can be the difference between success and failure. In this context, TLBO arises as a good alternative because it simplifies the parameter tuning and control processes due to its reduced number of parameters. The aim of this paper is to propose a parameter tuning method for the TLBO algorithm, considering both the accuracy and the speed. Numerical experiments are carried out to illustrate the proposed parameter tuning method in troubleshooting optimization problems.

KEYWORDS. TLBO. Parameter Tuning. Combinatorial Optimization. Troubleshooting.

Paper topics: MH - Metaheuristics; OC - Combinatorial Optimization

1. Introduction

Evolutionary algorithms (EA) are powerful in the way they efficiently provide good solutions for real-world problems [Simon, 2013]. They have been successfully used when the search space is too large, the problem is too complex, the evaluation function is too complicated, the solution is too constrained, and/or there is no hint of how to solve the problem [Michalewicz and Fogel, 2004]. These algorithms though suffer from their generality, i.e., their lack of restrictions on their parameters. Therefore, it is always necessary to finely tune the parameters such that computational resources and algorithm accuracy are not affected. The Genetic Algorithm, for example, has a high sensitivity to mutation probability, crossover probability and population size [Pinel et al., 2012; Srinivas et al., 2014].

The selection of good parameters is usually done by one of these two classes of methods: parameter tuning, or parameter control [Eiben and Smit, 2012]. The former denotes the set of methods which define the parameters before the optimization is run, while the latter denotes the methods in which the parameters are dynamically defined during the execution of the optimization algorithm. Parameter control methods can still be categorized into deterministic, adaptive, or self-adaptive.

It is important to note that it can hardly be said that a set of good parameters for a given problem will be generally good. Parameters are usually mutually sensitive and very dependent on the models and problems at hand. Those reasons strengthen the need for parameter control, even though it is harder to achieve [Wolpert and Macready, 1997; Nannen and Eiben, 2006].

An algorithm which has been recently proposed by Rao et al. [2011], the Teacher-Learner Based Optimization algorithm (TLBO), needs considerably less parameters: only the population size, besides the stop criteria. It is based on the teaching-learning process observed in a classroom and simulates the influence of a teacher on the output of a group of students in a class.

TLBO has achieved remarkable performances in different types of problems such as constrained [Rao and Patel, 2012], and unconstrained [Rao and Patel, 2013] optimization problems. TLBO has also been successfully used in other combinatorial optimization problems such as the flow shop and the job shop scheduling problems [Baykasoğlu et al., 2014], the set covering problem [Crawford et al., 2015] and the disassembly sequence planning problem Xia et al. [2013]. This paper aims to explore the optimization capabilities of the TLBO algorithm and the relationship between its parameters and computational time and accuracy.

The optimization problem used to guide this study is the troubleshooting optimization problem described in section 3. Troubleshooting optimization problem is a combinatorial optimization problem which is known to be NP-hard for most of real applications [Vomlelová, 2003].

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

2. Teaching-Learning Based Optimization Algorithm

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 J. In the troubleshooting problem considered in this paper, a solution X corresponds to a troubleshooting strategy S and the associated result $f(X)$ corresponds to its Expected Cost of Repair, denoted by $ECR(S)$ [Vianna et al., 2016].

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 1. The Teacher Phase and the Student Phase are described in the next sections.

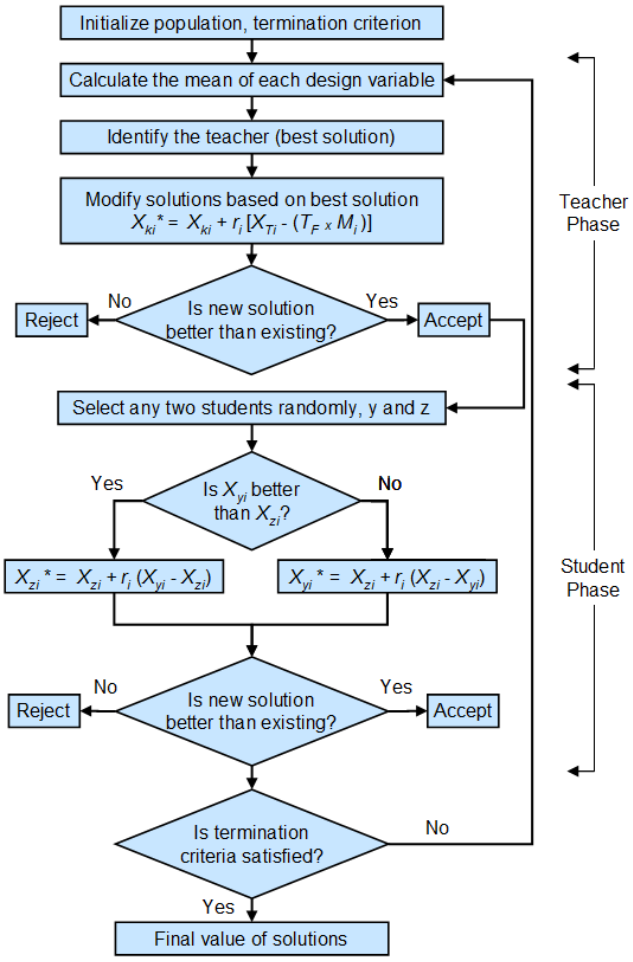

Figure 1: TLBO algorithm

2.1. 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 (1):

$$D_i = r_i(X_{Ti} - T_F \cdot M_i) \tag{1}$$

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 (2):

$$T_F = round(1 + rand(0,1)) \tag{2}$$

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 (3):

$$X^{\star}_{ki} = X_{ki} + D_i \tag{3}$$

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

If $f(X^{\star}_{ki})$ is better than $f(X_{ki})$, $X^{\star}_{ki}$ is accepted and replaces X_{ki}. Otherwise, $X^{\star}_{ki}$ is discarded and X_{ki} is not changed for the next iteration.

2.2. 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 (4). Then, if $f(X^{\star}_{zi})$ is better than $f(X_{zi})$, $X^{\star}_{zi}$ is accepted and replaces X_{zi}. Otherwise, $X^{\star}_{zi}$ is discarded and X_{zi} is not changed for the next iteration.

Similarly, if $f(X_{zi})$ is better than $f(X_{yi})$, the solution of student y is updated according to Equation (5). Then, if $f(X^{\star}_{yi})$ is better than $f(X_{yi})$, $X^{\star}_{yi}$ is accepted and replaces X_{yi}. Otherwise, $X^{\star}_{yi}$ is discarded and X_{yi} is not changed for the next iteration.

$$X^{\star}_{zi} = X_{zi} + r_i(X_{yi} - X_{zi}) \tag{4}$$

$$X^{\star}_{yi} = X_{yi} + r_i(X_{zi} - X_{yi}) \tag{5}$$

At the end of each iteration, the stop criteria must be 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 maximum number of iterations is adopted as the stop criterion, as in other applications of TLBO in combinatorial problems [Baykasoğlu et al., 2014], [Crawford et al., 2015], [Patil, 2016].

3. Troubleshooting Problem

Troubleshooting is the name given for the sequence of actions performed in order to fix a given system. These actions are commonly separated into two categories: diagnostic and repair [Vomlelová, 2003].

The first type of action, usually less expensive and time consuming, can be inconclusive, i.e., it may represent some cost for no advancement toward an actual solution. Repair actions, on the other hand, are a step in the direction of solving the problem. Several actions might be required to fix a single fault, as well as several faults might have occurred simultaneously.

These types of problems are very practical and common in our daily lives. The decision-theoretic troubleshooting comes as a way to model and study our way of making decisions to optimize the troubleshooting process and thus minimize the costs involved.

A common way to represent this problem is through oriented graphs, such as the one presented in Figure 2. Nodes represent the possible failure modes (F_i), diagnose questions (Q_i) and repair actions (A_i), while the oriented arcs represent the connections between failure modes and actions [Vianna et al., 2016].

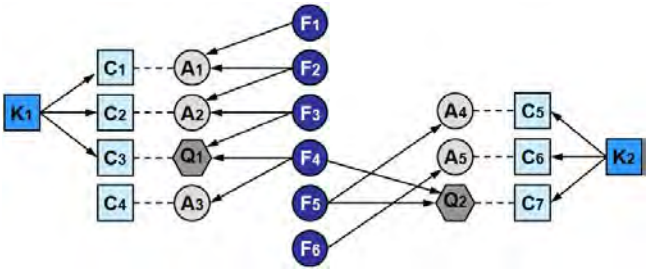

Figure 2: Troubleshooting model 1 - Bayesian network representing a simple troubleshooting problem.

Costs are represented by C_i, while cost clusters are represented by K_i. A fault caused by failure mode F_i is only fixed when all repair actions connected to it are executed. A diagnostic question isolates a subset of possible failure modes. Each action and question has an associated cost, which is incurred if the action or question is executed. A cluster cost is incurred if at least one action or question connected to it is executed.

Some assumptions need to be made in order to ensure the optimality of the sequence found [Langseth and Jensen, 2001]:

1. **Single Fault:** only one fault can be present at the system at a time;
2. **Perfect Repair:** repair actions are always effective;
3. **Fixed Cost:** cost does not vary with time;
4. **No Questions:** no diagnose actions are allowed; and
5. **Independent Actions:** each action addresses exactly one fault.

Without these restrictions, there is no polynomial approximation algorithm for such problem [LÃn, 2014]. Hence, the usage of a metaheuristic optimization algorithm is justified. In this paper, only the first three assumptions are considered such as to limit the complexity of the problem at hand without distancing it too much from real world premises.

The solution for a troubleshooting problem is given in terms of a troubleshooting strategy. There is a sequence of actions (either repair or diagnostic) that minimize the Expected Cost of Repair (ECR) of a system according to the probability of each fault and the cost of each action. Figure 3 shows an example of a troubleshooting strategy.

4. Numerical Experiments

4.1. Experiment Setting

This section presents numerical experiments to illustrate the application of the proposed method in three different troubleshooting models. Table 1 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 troubleshooting models 1, 2 and 3 are shown in Figures 2, 4 and 5, respectively.

Table 2 presents the probability associated with each failure mode in each troubleshooting model. Table 3 shows the costs associated with each action, question and cluster.

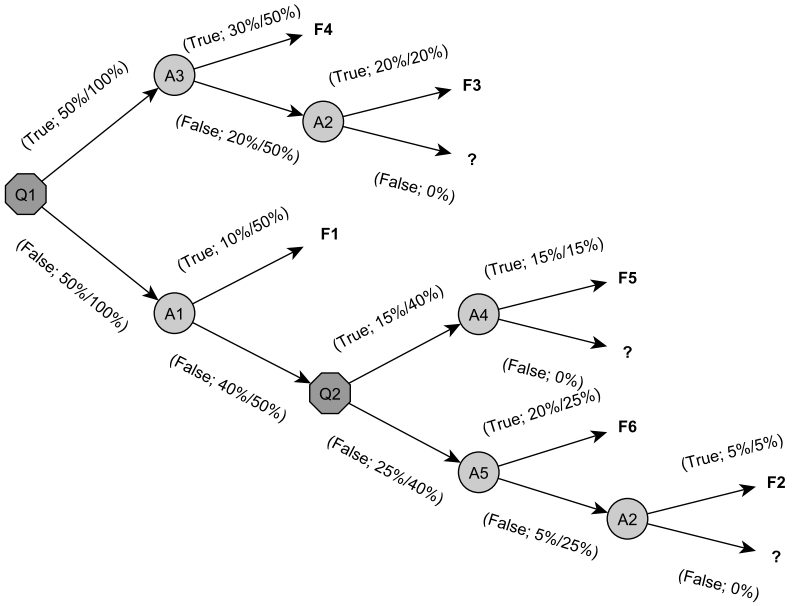

Figure 3: Troubleshooting strategy for the model presented in Figure 2. Edges indicate whether the action was successful (true) or unsuccessful (false) and the probability this path is taken according to Table 2. The costs at the path ends are computed based on Table 3. The Estimated Cost of Repair for this strategy is $\text{ECR}(\text{solution}) = 45 \cdot 30\% + 65 \cdot 20\% + 25 \cdot 10\% + 115 \cdot 15\% + 125 \cdot 20\% + 145 \cdot 5\% = 78.5$.

Table 1: 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

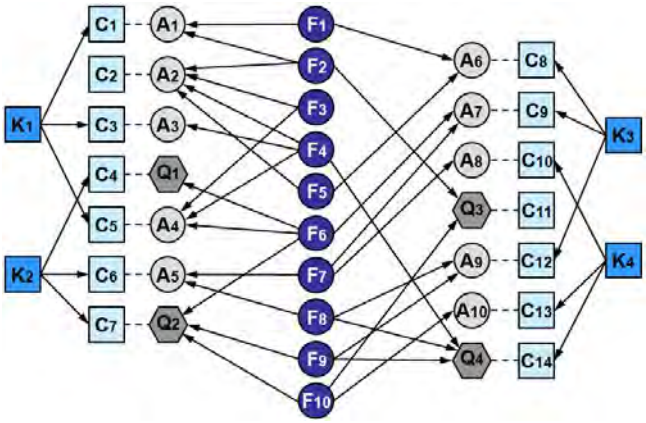

Figure 4: Troubleshooting model 2

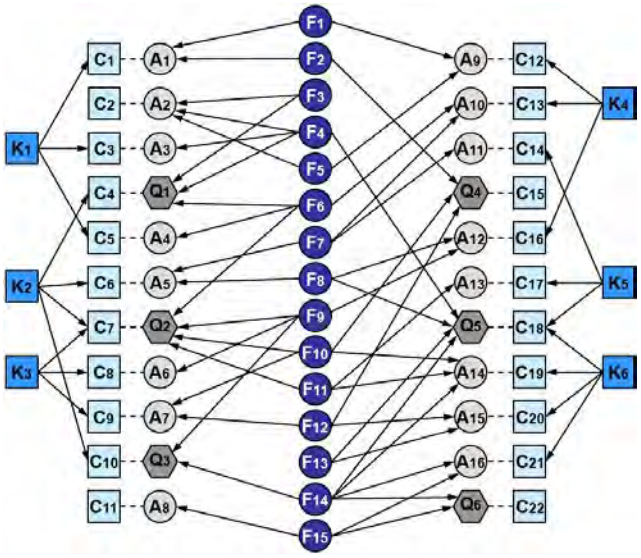

Figure 5: Troubleshooting model 3

Table 2: Failure modes probabilities for each model

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

Table 3: 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	-	-	-	-	-	-	-	-	-	-

4.2. Optimization Conditions

These models were optimized through a Matlab ® implementation of TLBO. The experiments were run on a computer with a 4-core, 8-thread Intel Core i7 2.4 GHz processor and 12 GB of RAM. Both of these critical resources were monitored during the execution and did not reach 100%. Six parallel threads were used to speed up the simulations.

The population size was set to one of 15 values varying from 2 to 250 in a logarithmic, non-repeating progression. The maximum number of iterations allowed was set to 5,000, as a very large condition, so as to simulate an infinite amount of iterations. The statistic stability of the experiment was verified by a Monte Carlo approach with 50 repetitions for each parameter combination. Their ECR and simulation time for each iteration were registered for each repetition, allowing the analysis to be made post-simulation.

4.3. Results

As expected, the results presented in Figure 6 show that the algorithm converges faster for larger populations. This behavior was observed in all three models. Model 1 converges to the final solution before 10 iterations for populations larger than 89, for example. Also, note from Figure 6 that the curves become closer to each other as the model size increases. It can be assumed from this result that population size keeps having a smaller impact on algorithm accuracy as population increases.

Figure 7 shows that the probability of the optimization to converge for model 1 is very low for population of 2, but reaches high levels of success (greater than 80%) for more than 8 learners through 1000 iterations. The same level of results are reached when 89 learners were used for model 2. Model 3 was too complex to return any statistically meaningful results for any setup proposed during the study, but it was kept as a limit case.

Figure 8 shows how the simulation time grows with population size for each model and how its standard deviation grows. The R^2 (for linear regressions) shows a high probability that the time grows linearly with population size. The standard deviation also shows the algorithm's simulation time becomes more unpredictable even though it's envelope remains linearly contained.

Combining all information in Figures 6, 7 and 8, it becomes obvious the influence the parameters have on the optimization outcome. Setting the parameters poorly may not only reduce the chances of getting to the optimal problem solution, but also consume a lot of time. Figure 9 summarizes the results of this paper, and provides a way to find better parameters for models in the range considered. Note that the algorithm is nonlinear, and as such, the contour plots might present noisy results. Its purpose as a guidance is not reduced by this fact though.

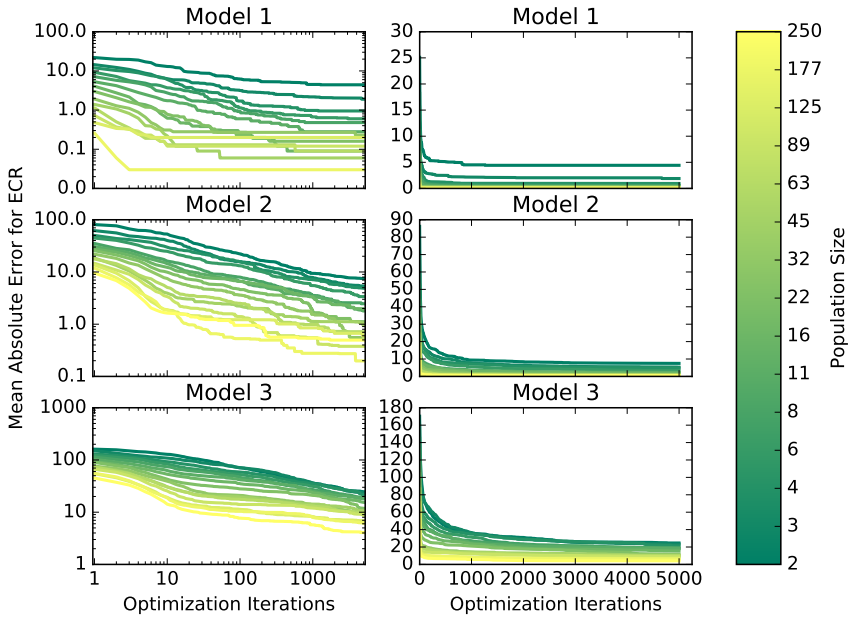

Figure 6: Mean Absolute Error of the Expected Cost of Repair for each model and each population size. Log-log axis used on the left to make it possible to visualize all models on same axis ranges. Linear-linear axis used on the right to show the exponential convergence of the algorithm.

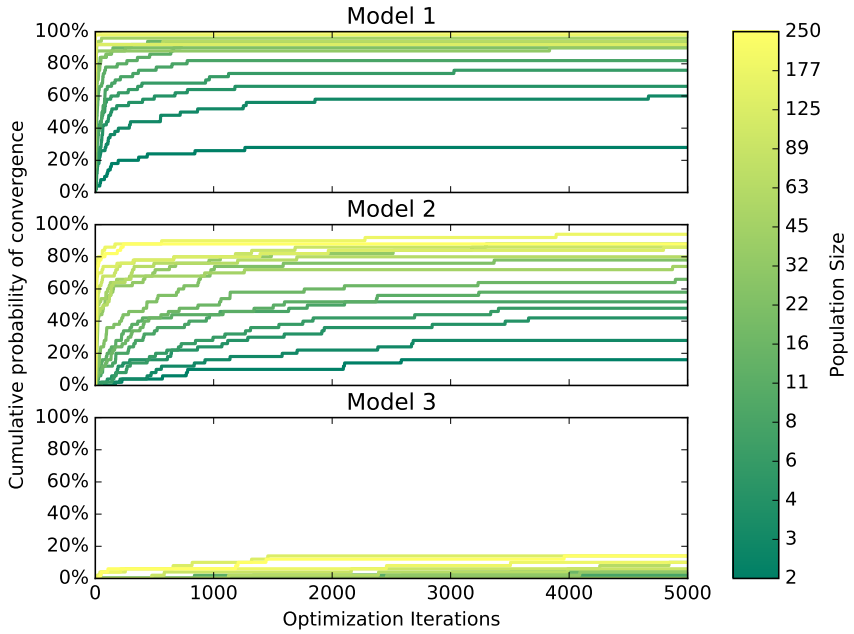

Figure 7: Cumulative probability that the simulation converged for a given amount of iterations and population size.

5. Conclusions

There is an optimal parameter setting for each troubleshooting model that runs through the TLBO algorithm. In this paper, it was shown that the time needed to find a stable solution for a troubleshooting optimization problem varies linearly with the population size. The Mean Absolute Error (MAE) decays exponentially with the same parameter.

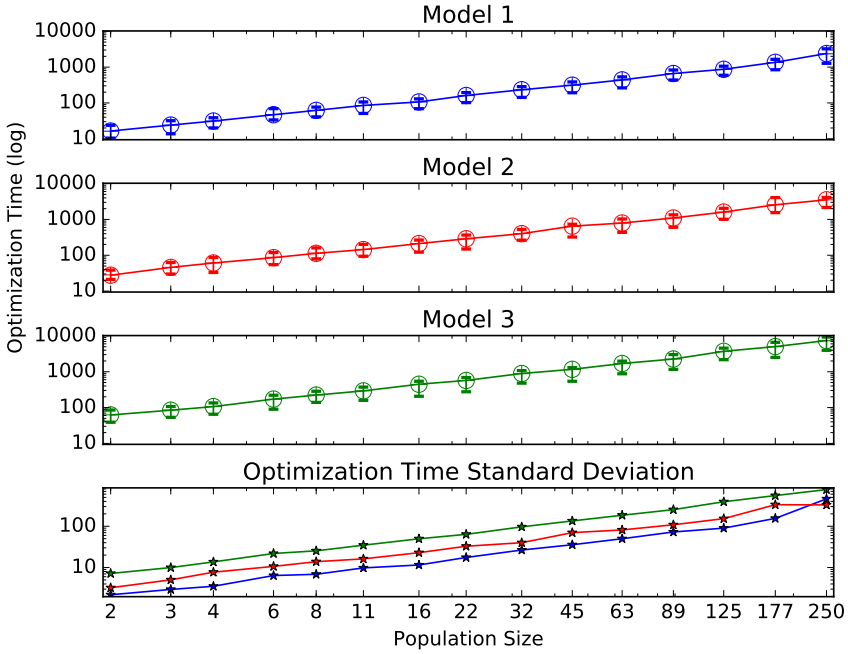

Figure 8: Computational time vs population size. Linear R^2 are 0.9689, 0.9892 and 0.9898 for models 1, 2 and 3 respectively. Linear R^2 for standard deviation is 0.9211, 0.9749, and 0.9995 for models 1, 2 and 3 respectively. Log-log axis used to make it possible to show full time range for all models.

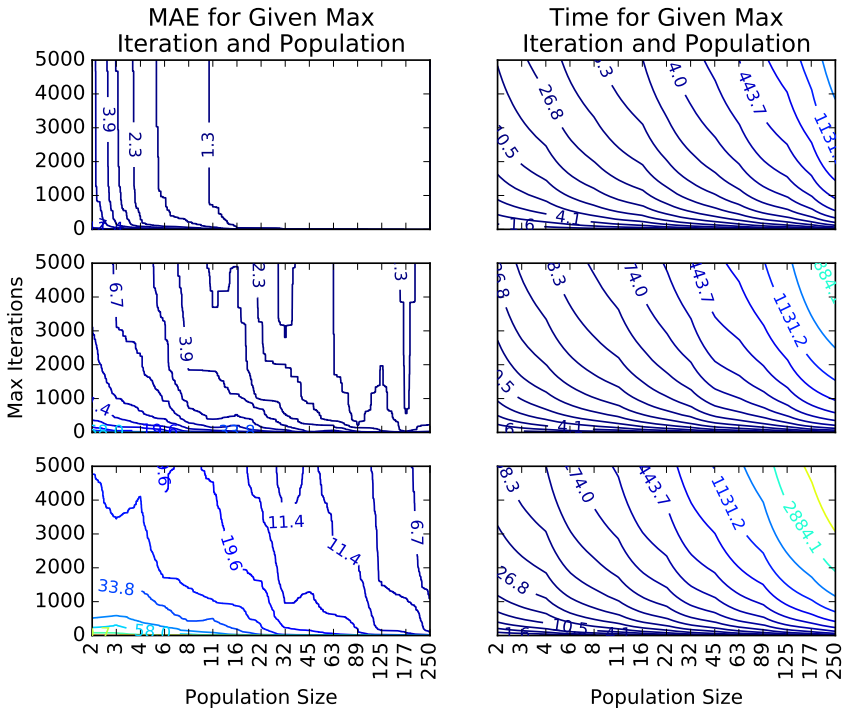

Figure 9: Relationship of maximum iteration and population size with mean absolute error and simulation time.

The use of an optimization algorithm to solve troubleshooting problems is justified only to

models with dozens of fault nodes, i.e. bigger than model 2, since smaller models can be solved in a feasible time through extensive search. It became clear how the troubleshooting problem becomes harder to solve as the model comes closer to real-life scales, to the point in which its accuracy cannot be guaranteed.

An extension of this work will focus on increasing the ranges up to large scales, while comparing the results with well-known evolutionary algorithms in order to investigate whether the limitations are in the TLBO algorithm or in the complexity of the problem under consideration. Other approaches such as hybrid evolutionary algorithms will also be considered to ensure the quality of the solution.

The influence of the population size on the Estimated Cost of Repair seems to diminish as population increases, as shown in section 4.3. This fact suggests that there might be a point where the population size only affects the simulation time but leaves the mean absolute error unmoved. Exploring the limits of the population size might lighten this matter.

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.

Crawford, B., Soto, R., Leiva, F. A., Johnson, F., and Paredes, F. (2015). Problema del conjunto de cobertura resuelto mediante el algoritmo binario de optimización basado en enseñanza-aprendizaje. In *Conferência Ibérica de Sistemas e Tecnologias de Informação*, p. 106–109, Águeda. IEEE.

Eiben, A. E. and Smit, S. K. (2012). *Autonomous Search*, chapter Evolutionary Algorithm Parameters and Methods to Tune Them, p. 15–36. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-21434-9. URL http://dx.doi.org/10.1007/978-3-642-21434-9_2.

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

LÃn, V. (2014). Decision-theoretic troubleshooting: Hardness of approximation. *International Journal of Approximate Reasoning*, 55(4):977 – 988. ISSN 0888-613X. URL http://www.sciencedirect.com/science/article/pii/S0888613X1300162X. Special issue on the sixth European Workshop on Probabilistic Graphical Models.

Michalewicz, Z. and Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Springer, enlarged 2nd edition. ISBN 9783540224945. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/3540224947.

Nannen, V. and Eiben, A. (2006). A method for parameter calibration and relevance estimation in evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, p. 183–190, New York, NY, USA. ACM. ISBN 1-59593-186-4. URL http://doi.acm.org/10.1145/1143997.1144029.

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

Pinel, F., Danoy, G., and Bouvry, P. (2012). *Security and Intelligent Information Systems: International Joint Conferences, SIIS 2011, Warsaw, Poland, June 13-14, 2011, Revised Selected Papers*, chapter Evolutionary Algorithm Parameter Tuning with Sensitivity Analysis, p. 204–216. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-25261-7. URL http://dx.doi.org/10.1007/978-3-642-25261-7_16.

Rao, R. V. and Patel, V. (2012). An elitist teaching-learning based optimization algorithm for solving complex constrained optimization problems. *International Journal of Industrial Engineering Computations*, 3:535–560.

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.

Simon, D. (2013). *Evolutionary Optimization Algorithms*. Wiley. ISBN 9781118659502. URL `https://books.google.com.br/books?id=gwUwIEPqk30C`.

Srinivas, C., Reddy, B. R., Ramji, K., and Naveen, R. (2014). Sensitivity analysis to determine the parameters of genetic algorithm for machine layout. *Procedia Materials Science*, 6:866 – 876. ISSN 2211-8128. URL `http://www.sciencedirect.com/science/article/pii/S2211812814004696`. 3rd International Conference on Materials Processing and Characterisation (ICMPC 2014).

Vianna, W. O. L., Rodrigues, L. R., Yoneyama, T., and Mattos, D. I. (2016). Troubleshooting optimization using multi-start simulated annealing. In *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.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *Trans. Evol. Comp*, 1(1):67–82. ISSN 1089-778X. URL `http://dx.doi.org/10.1109/4235.585893`.

Xia, K., Gao, L., Wang, L., Li, W., and Chao, K. M. (2013). A simplified teaching-learning-based optimization algorithm for disassembly sequence planning. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, p. 393–398.