Revista Brasileira de Ciência, Tecnologia e Inovação

DOI: https://doi.org/10.18554/rbcti.v6i1.5403

Research Paper

Inventory Optimization in a soft drink company based in Python

Otimização do estoque de uma fábrica de bebidas não alcoólicas em Python

Breno Prado Franchini Bizerra[1]*, Vinicius Abrão da Silva Marques 2*

* Department of Mechanical Engineering, Federal University of Triângulo Mineiro, Uberaba, MG, Brazil.
1 Orcid: 0000-0002-0078-0401 E-mail: breno.bizerra@gmail.com, 2 Orcid: 0000-0002-5499-0404 E-mail: vinicius.marques@uftm.edu.br

ABSTRACT: The aim of the present article is to present computational mathematical modeling using to optimize a company's inventory based on Python. The study concerned soft drink production organization. Algorithms Scipy.optimize.minimize and Genetic Algorithm were herein used to optimize the quantity of products in stock by taking into consideration different manufactured goods, forecast sales based on sales history, and production maximum capacity. The adopted methods were compared to assess their responses and processing time. Based on the results, both methods are capable of giving optimal responses. The Genetic Algorithm method gives lesser optimized answers to each product, but it accounts for better general production distribution, although it needs more processing time to analyze all possibilities. The Scipy.optimize.minimize method provided more optimized answer to most products, but it was not efficient in production distribution, although it presented shorter processing time.

Keywords: Inventory Optimization, Sales forecast, Python, Scipy, Genetic Algorithm.

RESUMO: Este trabalho tem o intuito de apresentar a utilização de modelagem matemática computacional para otimização em Python do estoque de uma fábrica. O problema é construído em torno da organização da produção de bebidas. A utilização dos algoritmos *Scipy.optimize.minimize* e Algoritmo Genético (*Genetic Algorithm*) empregados neste trabalho tem como objetivo otimizar a quantidade de produtos em estoque, levando em conta os diferentes produtos fabricados, a previsão de venda baseada em histórico e a capacidade máxima de produção. Os métodos utilizados são comparados em suas respostas e tempo de processamento. Os resultados mostram que ambos os métodos têm a capacidade de encontrar respostas otimizadas para o problema. O método *Genetic Algorithm* encontra respostas menos otimizadas para cada produto, mas realiza uma melhor distribuição geral da produção, além de necessitar de maior tempo de processamento para analisar todas as possibilidades. O método *Scipy.optimize.minimize* encontra uma resposta mais otimizada para a maioria dos produtos, mas não realiza uma distribuição eficiente da produção, além de apresentar um menor tempo de processamento.

Palavras-chave: Otimização de Estoque, Previsão de vendas, Python, Scipy, Algoritmo Genético.

INTRODUCTION

The remarkable growth in soft drinks' production in Brazil in the last few years demanded more efficient ways to organize production and more efficient machine operation schedules in order to increase industrial yield (PAGLIARUSSI; MORABITO; SANTOS, 2016). Increase in the number of small- and medium-sized companies, and increased competition and quality in the manufacture category is now a reality. However, how can one be ahead of the market? (SEBRAE, 2021).

Using computational tools to simulate production, its modeling and creating mathematical optimization methods allows companies to decrease machine setup time and to increase production schedule efficiency. Thus, it is possible achieving cost reduction with the manufacturing process and, consequently, reduced final products' cost.

The *Genetic Algorithm* is one of the most used computational mathematical optimization methods, it is an evolutionary process that works with populations of candidates (possible answers to problems) in parallel. Therefore, it is possible seeking different solutions through members' allocation in several response spectra. It works as "generating and testing" method to assess optimum, or almost optimum, responses, without significant limitation, since it is based on biological evolutions (CARVALHO, 2021).

Scipy.optimize.minimize is another computational technique adopted for mathematical optimization. It allows finding the minimum and maximum values, or the zeros of the function, as well as stands out for its simple implementation (SCIPY-LECTURES, 2020).

It is essential knowing the evolution process and the whole set of manufactured products in order to apply the herein addressed methods and optimize a soft drink production factory. After such a knowledge is acquired, it is possible suggesting solutions to efficiently meet the company's demands (PESSANHA; ALVARENGA; ARICA, 2015).

The aim of the present study was to optimize the inventory of a soft drinks company based on data, such as that about its sales history and inventory control, which were provided by the company itself. It was possible implementing an algorithm to analyze sales predictions and inventory management by applying the optimization methods, based on the provided data, without adopting any invasive machine operation methodology in the company.

If one takes into account that products' sales variations are seasonal, it is possible estimating the necessary production based on sales recorded for previous months and, consequently, avoiding product shortage in the inventory or excessive production, since both scenarios are harmful for the company's budget.

Among advantages in applying the herein tested methods, one finds low implementation cost due to Python language adoption, which is a free access software. Furthermore, there was no need of making face-to-face visits to the company, since it just demanded using the company's database.

Based on the present results, costs indirectly generated by better inventory control were reduced, and it has improved production organization, shortened work shifts and machine setups

METHODOLOGICAL PROCEDURES

Production line operation of a real soft drinks production company located in São Paulo State was analyzed in order to find the closest solution to the nominal optimum for the herein proposed problem. Inventory-control mathematical modeling was carried out based on the company's sales history. Other parameters were also mapped, among them: maximum production capability, which is closely linked to the number of bottles the PET-bottle molding machine can produce. This process was pointed out as the most critical in the company; thus, the method was adopted to model the objective function and its constraints, which are used in optimization methods – these constraints are listed below. It is possible finding the optimum and real solution possible for the problem.

Constraints

Production limit, threshold inventory and predicted sales data allowed finding mathematical equations capable of representing the company's real constraints. Constraint's equations are defined by two dependent variables: $rest1$ and $rest2$, which were defined in order to be implemented through each one of the herein assessed methods.

Lists of products' amounts were created for each coding routine. These lists provide predicted sales values (Vp), which are based on sales records from previous years. Subsequently, the other algorithms' input parameters were calculated based on initial inventory (Ei – equation 1), expected inventory (Ed – equation 2), and threshold inventory (L – equation 3). These calculations were proportionally defined based on $Vp,$ according to expectations by the company's managerial team.

$$Ei = Vp \times 0.6 \tag{1}$$
$$Ed = Vp \times 0.7 \tag{2}$$
$$L = Vp \times 0.1 \tag{3}$$

Function inequalities, which must not be negative, were used to limit the *Scipy.optimize.minimize* algorithm's constraints. Based on definition by the methods' library, constraints variables' values must be higher than, or equal to, zero. It is possible calculating "x" of each product "j", which must be produced to meet the demand, based on sales predictions data set for each product found in each Vp.

$$Ef[j] = Ei[j] - Vp[j] + x[j] \tag{4}$$
$$rest1[j] = Ef[j] - L[j] \tag{5}$$
$$rest2 = P\_max - np.sum(x) \tag{6}$$

Wherein, "Ef" (equation 4) is the final production-cycle inventory, which is given by the difference between initial inventory, plus production, minus the sales prediction; P_max is the company's maximum real production; and $np.sum(x)$ (*NumPy* command library) is the sum of production recorded for all products. The difference between final inventory and threshold inventory must always be higher than, or equal to, zero in the first constraints, equation 5. As for the second constraints, equation 6, the sum of the production recorded for all products must be lower than, or equal to, the company's production limit.

Scipy.optimize.minimize algorithm's constraints are based on the same information provided to the *Genetic Algorithm* method. However, while the first constraints is still found through equation 5, the second one is given by equation 7 in order to meet the algorithms' operation demands, in this case:

$$rest2 = np.sum(x) - P\_max \quad (7)$$

Algorithms seek to optimize the difference between final and expected inventory by varying the number of produced goods. The response must be searched within the range defined by inequality constraints, while to converge to the best result possible. A penalty algorithm was developed to make the created constraints force the system to converge to outcomes that are in compliance with reality. Based on this algorithm, the responses found outside the constraint's thresholds match a higher value. In this case, 10^{10} was the chosen penalty for users to get to known that the received response was penalized; in other words, the code discards the solution and generates a new iteration in order to seek an optimized solution.

After defining the constraints system of equations for both algorithms, one can model the objective function.

Objective function

The objective function is calculated based on the same database used for the constraints, i.e., it used the Final Inventory formulation (equation 4) and the Expected inventory (equation 2) to optimize the difference between the two parameters.

$$Var = np.sum(np.abs(Ed[j] - Ef[j])) \quad (8)$$

The problem can be summarized to a single objective function called Var.

Based on equation 8, which is found by summing all produced goods in the modulus of differences between the expected inventory and the final one. The same formulation was adopted to both algorithms; therefore, the algorithm will search for production $x[j]$ to minimize the Var value by respecting the implemented constraints.

In case the system does not converge, it is possible changing the number of iterations input and population size so that it will be more likely for the algorithm to find a converging response. It is the case when the number of iterations is too low, according to which, the code cannot converge to an optimized outcome.

RESULTS AND DISCUSSION

Computational Results

This section addresses the recorded outcomes for the implementation of two optimization methods (*Genetic algorithm* and *Scipy.optimize.minimize*).

The following Python libraries were used:

1- geneticalgorithm [1];
2- scipy.optimize along with the SLSQP method [2].

Values of setup parameters were chosen based on the sensitivity analysis applied to each one of the algorithms. Given that the aim of the present study does not lie on discussing parameters' influence, but the outcomes of the herein assessed case, the case parameters and results accounting for the most optimized outcomes found will be presented. The case recording the lowest objective function value found will be considered the most optimized one (equation 8). It means that parameters' choice was made regardless of processing time. Processing time outcomes are presented just to compare the methods' computational performance based on simulations carried out in an Intel Core i7 computer with 8 Gb of RAM memory.

Code initialization parameters set for the *Genetic Algorithm* were determined as shown in equations 9 to 13.

$$n = 22 \tag{9}$$
$$Ea = np.zeros(n) \tag{10}$$
$$Var = np.zeros(n) \tag{11}$$
$$rest1 = np.zeros(n) \tag{12}$$
$$rest2 = np.zeros(n) \tag{13}$$

The internal parameters chosen in the *Genetic Algorithm* to deliver the most optimized responses about the processing capacity of the used computer were chosen as presented in equations 14 to 21.

$$'max\_num\_iteration': 800 \tag{14}$$
$$'population\_size': 800 \tag{15}$$
$$'mutation\_probability': 0.2 \tag{16}$$
$$'elit\_ratio': 0.01 \tag{17}$$
$$'crossover\_probability': 0.3 \tag{18}$$
$$'parents\_portion': 0.3 \tag{19}$$
$$crossover\_type': 'uniform' \tag{20}$$
$$'max\_iteration\_without\_improv': None \tag{21}$$

With respect to algorithm *Scipy.optimize.minimize*, the same initial parameters set for the *Genetic Algorithm* were adopted for algorithm *Scipy.optimize.minimize*. Internal parameters were defined based on the options provided by the "SLSQP" method, as shown in equations 22 and 23.

$$'type': 'ineq', 'fun': restricao1 \tag{22}$$
$$'type': 'ineq', 'fun': restricao2 \tag{23}$$

The used information was collected in the company's database and from technical information provided by the person in charge of production. The present study took into account all real constraints of the analyzed company, so there would be no divergence in

[1] Available at: https://pypi.org/project/geneticalgorithm/
[2] Available at: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html

the recorded results and in the likelihood of implementing it. Sales recorded in October 2020 were taken into account as initial data for the current study, as well as the total production of 430,000 bottles under normal machine operation.

Table 1 provides the number of sold bottles of each good produced in the company in October 2020. These data made it possible finding the initial, expected and limit inventory values to be applied in code formulations.

Table 1. Number of bottles of each product sold in October 2020.

	PRODUCT	NUMBER OF BOTTLES
1	Guarana 2L	37,674
2	Gipps guarana 2L	16,872
3	Orange 2L	18,702
4	Lime 2L	9,186
5	Citrus 2L	12,474
6	Cola 2L	37,980
7	Sparkling water 2L	9,180
8	Lime sparkling water 1L	4,716
9	Sparkling water 510ml	42,120
10	Lime sparkling water 510ml	97,380
11	Apple sparkling water 510ml	30,876
12	Guarana 250ml	21732
13	Orange 250ml	4620
14	Apple 2L	38586
15	Grape 2L	11406
16	Grape 250ml	4884
17	Cola 250ml	6756
18	Cola 1L	2592
19	Guarana 1L	6660
20	Apple 1L	11784
21	Apple 250ml	17652
22	Energetic drink k10	1800

Based on data in **Table 1**, it was possible finding the ideal production according to the initial inventory and sales prediction, namely: 490,195 bottles. Such an ideal production is indicative of what the production of each item should be in order to make it possible reaching the ideal expected inventory. However, since the company's maximum production is 430,000 bottles a month, it is necessary optimizing production in order to meet the best production outcomes of each product and, therefore, to minimize the difference between the expected inventory and the final one.

Production outcomes from *Scipy.optimize.minimize* and *Genetic Algorithm* necessary to optimize the inventory are shown in **Table 2**.

It is possible observing that the necessary productions found by the two algorithms aim at finding the closest final inventory to the expected inventory through different ways. Although algorithm *Scipy.optimize.minimize* recorded processing time close to 1 second, it is also possible observing that the *Genetic Algorithm* (1:30-minute processing time) was more effective, because it found the most optimized result and the best real applicability results, when it was adapted to the necessary production of each product.

Table 2. Optimized production of each product.

	PRODUCT	SCIPY MINIMIZE	GA
1	Guarana 2L	41,441	29,334
2	Gipps guarana 2L	18,559	15,725
3	Orange 2L	20,572	17,562
4	Lime 2L	10,104	7,238
5	Citrus 2L	13,721	12,816
6	Cola 2L	41,777	35,068
7	Sparkling water 2L	10,097	8,121
8	Lime sparkling water 1L	5,187	3,857
9	Sparkling water 510ml	46,331	36,967
10	Lime sparkling water 510ml	46,922	106,227
11	Apple sparkling water 510ml	33,963	31,629
12	Guarana 250ml	23,905	21,524
13	Orange 250ml	5,081	4,302
14	Apple 2L	42,444	39,519
15	Grape 2L	12,546	12,444
16	Grape 250ml	5,372	3,176
17	Cola 250ml	7,431	5,743
18	Cola 1L	2,851	1,802
19	Guarana 1L	7,325	4,856
20	Apple 1L	12,962	11,401
21	Apple 250ml	19,417	19,403
22	Energetic drink k10	1,979	1,264
	Total	429,987	429,978

Table 3 presents the comparison between inventory values resulting from the application of both methods.

Based on **Table 3**, algorithm *Scipy.optimize.minimize* seeks the best necessary production optimization for each product, even if it ends up lacking production space for one of the products. With respect to *Genetic Algorithm*, one can observe better interpretation about the company’s needs, and it makes production smaller for each product, but it would also always fulfil a production priority, in other words, it seeks results closer to reality.

Yet, according to **Table 2**, total production based on each algorithm is closer to the company’s limit (430,000 bottles); therefore, it would not be possible increasing production. However, the goal is to minimize the difference between expected and final inventory, as shown in **Table 3**. Based on **Table 3**, the total value based on each algorithm is below the expected results, but it is not possible increasing production, as shown in **Table 2**. The higher the company’s maximum yield, the greater the difference in the total of final inventories, since the algorithm aims at optimization, so that this difference would be as small as possible if one takes into consideration the constraints. The number of produced goods added to the initial inventory also meets sales predictions in **Table 1**; in case the opposite happens, the final inventory would be negative, but it was not observed. Maximum production constraints only affect the final inventory; therefore, it does not concern sales losses, but inventory control based not only on the number of produced bottles, but also on the initial inventory and on sales predictions. These data are collected in the sales history records shown in **Table 1**. Finally, final inventory constraints above the limit were also met by both algorithms.

Table 3. Comparison between final optimized inventory and the expected inventory.

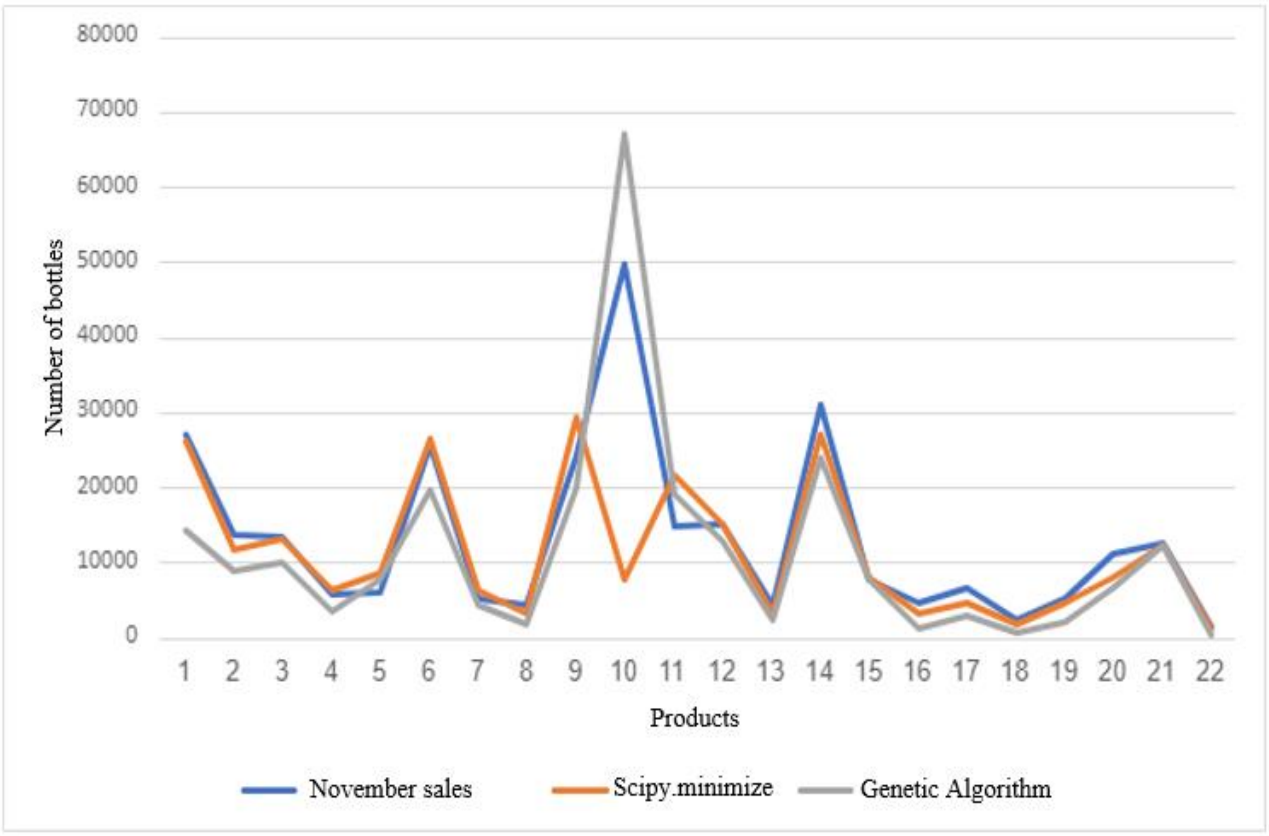
	OCTOBER SALES	EXPECTED INVENTORY	SCIPY MINIMIZE FINAL INVENTORY	GENETIC ALGORITHM FINAL INVENTORY
1	Guarana 2L	26,371	26,371	14,265
2	Gipps guarana 2L	11,810	11,810	8,977
3	Orange 2L	13,091	13,091	10,081
4	Lime 2L	6,430	6,430	3,564
5	Citrus 2L	8,731	8,731	7,828
6	Cola 2L	26,586	26,585	19,876
7	Sparkling water 2L	6,426	6,425	4,449
8	Lime sparkling water 1L	3,301	3,301	1,970
9	Sparkling water 510ml	29,484	29,483	20,119
10	Lime sparkling water 510ml	68,166	7,970	67,275
11	Apple sparkling water 510ml	21,613	21,613	19,279
12	Guarana 250ml	15,212	15,212	12,831
13	Orange 250ml	3,234	3,233	2,454
14	Apple 2L	27,010	27,010	24,085
15	Grape 2L	7,984	7,984	7,882
16	Grape 250ml	3,418	3,418	1,222
17	Cola 250ml	4,729	4,729	3,040
18	Cola 1L	1,814	1,814	765
19	Guarana 1L	4,662	4,661	2,192
20	Apple 1L	8,248	8,248	6,687
21	Apple 250ml	12,356	12,356	12,342
23	Energetic drink k10	1,260	1,259	544
	Total	311,936	251,734	215,727

Comparison between October optimized final inventory results and real November sales

After the optimized outcomes were recorded, it was possible comparing October final inventory data predicted by both methods; in other words, November initial inventory based on real sales values recorded by the company in November 2020 (**Figure 1**). Accordingly, it was possible deeply understanding the behavior of each algorithm and defining whether they would be reliable in managing production based on sales recorded for the following moth. Based on **Figure 1**, the *Genetic Algorithm* accounts for a safer inventory, i.e., it meets the November demand, even in cases when the sales of a given product do not follow the average. The same is not observed through the *Scipy.optimize.minimize* curve, whose graphic is extremely similar to that of November sales in almost every case, despite the case when sales behaved differently from the expected, such as product 10's case.

It was possible concluding that the *Genetic Algorithm* accounts for the best result because it better meets cases presenting lack of sales uniformity between products. Once again, **Figure 1** concerns an initial inventory analysis based on predicted sales, in other words, it does not mean saying that the inventory will not meet November sales. **Figure 1** actually shows that the final October inventory, which was optimized by the *Genetic Algorithm*, would almost meet the November sales' demand by itself, so it was considered to be a safe inventory. The algorithm can be once more used to predict November production, which will meet the initial inventory difference, the predicted sales and minimize the difference, so that November final inventory is as close as possible to the expected one, and yet, respecting the constraints. The process is repeated for the following months.

Figure 1 Comparison between October optimized final inventory and November sales.



It is essential highlighting that the difference in product 10's behavior emerges from the comparison between the *Scipy.optimize.minimize* results and future sales, i.e., at the moment to optimize November sales data were not yet available. Therefore, it does not mean a method divergence, but two different interpretations about a possible production distribution. *Genetic Algorithm* outcomes result in less different behavior concerning product 10, because October optimized production was higher than that by *Scipy.optimize.minimize;* thus, both solutions were valid and met the imposed constraints.

CONCLUSIONS

Two computational methods were herein used to solve an inventory control issue in a soft drinks company. Both implemented algorithms recorded optimized results, but with different applicability.

To the best of the authors' knowledge, there is no case study similar to the present one in the literature. Real data provided by the company were used in the present research and results recorded by the optimization models were discussed.

Comparatively, algorithm *Scipy.optimize.minimize* processing time is much faster than that of the *Genetic Algorithm*. However, its disadvantage lies on the fact that it does not cover the distribution of each product in overall production, and it can make high demand products face impaired production.

The *Genetic Algorithm,* in its turn, was able to find a sufficiently optimized production based on products' demand parameters, since it analyzes several possible responses and seeks to better fit the distribution of each product in overall sales. However, in order to have code efficiency, it is necessary having large iteration number and bigger code population size, a fact that leads to longer processing time.

In conclusion, the G*enetic Algorithm* accounted for the best results, since it better fulfilled the demand of cases that did not present sales uniformity between products, and it ensures safer final inventories for the coming month.

Based on the proposed method, it is possible optimizing different market needs and increase business profitability, without large operational costs.

The present authors suggest the application of the herein adopted methodology throughout the company's yearly production to identify the possible influence of non-mapped parameters in future studies and to improve the proposed model.

REFERENCES

CARVALHO, A. P. L. F. **Algoritmos genéticos**. Available at: https://sites.icmc.usp.br/andre/research/genetic/. Access on: Jan. 10th, 2021.

MALAQUIAS, N G. L. **Uso dos algoritmos genéticos para a otimização de rotas de distribuição**. 2006. 113p. Tese (Mestrado em Engenharia Elétrica) - Universidade Federal de Uberlândia, Faculdade de Engenharia Elétrica, Uberlândia, MG, 2006.

PAGLIARUSSI, M. S.; MORABITO, R; SANTOS, M. O. Otimização da programação da produção de bebidas à base de frutas por meio de modelos de programação inteira mista. **Scientific Electronic Library Online – Scielo,** p. 1, dezembro 2016. DOI: https://doi.org/10.1590/0104-530X2288-15. Available at: https://www.scielo.br/j/gp/a/XM8DcbSWFGDHCTPfHWJWrCB/?lang=pt. Access on: Jan. 9th, 2021.

PESSANHA, L. P. M.; ALVARENGA, R. L.; ARICA, G. G. M. Modelagem e resolução do problema integrado de dimensionamento e sequenciamento da produção: Caso de uma

pequena empresa de produtos de limpeza. *In*: **XXXV Encontro nacional de engenharia de produção**. Fortaleza, CE, 2015.
SCIPY-LECTURES. **Mathematical optimization: finding minima of functions**. Available at: https://scipy-lectures.org/advanced/mathematical_optimization/. Access on: Dec. 23rd, 2020.

SEBRAE. **Micro e pequenas empresas geram 27% do PIB do Brasil.** Available at: https://www.sebrae.com.br/sites/PortalSebrae/ufs/mt/noticias/micro-e-pequenas-empresas-geram-27-do-pib-do-brasil,ad0fc70646467410VgnVCM2000003c74010aRCRD. Access on: Jan. 9th, 2021.

Received on: 2021/04/12
Approved on: 2021/07/04