APPLICATION OF GENETIC ALGORITHM IN A LOGISTIC ENGINEERING PROBLEM

Aplicação do Algoritmo Genético em um Problema de Engenharia Logística

Aplicación del algoritmo genético en un problema de ingeniería logística

Revista Desafios

Artigo Original
Original Article
Artículo Original

Luiz Felipe Rocha Moreira[1], Jadiel Caparrós da Silva[*1]

[1] Curso de Engenharia Elétrica, Universidade Federal do Tocantins - UFT, Palmas/TO, Brasil.

**Correspondência: Laboratório de Modelagem em Sistemas Elétricos, Av. NS 15, 109 Norte, Plano Diretor Norte, Universidade Federal do Tocantins - UFT, Palmas/TO, Brasil. CEP:77001090. E-mail : jadiel@uft.edu.br.*

Artigo recebido em 24/11/2020 aprovado em 12/01/2022 publicado em 02/05/2022.

ABSTRACT

The research work described in this paper main aim to investigate the effectiveness of the Genetic Algorithm applied in solving a problem of Logistics Engineering. In addition, the paper presents a dense theory on the subject in order to contribute to researchers in this area. With this clear objectives, the composition of this paper will initially clarify the technical concepts used. Subsequently, the problem that will be solved as well as its modeling is presented, so that at the end an algorithm is presented, focusing on the construction of the logic of the algorithm, as well as the data obtained that prove the effectiveness tool as a way of solving the defined problem. The concepts behind the algorithm used here, are derived from the most recent studies on Artificial Intelligence and are based on biological studies of the theory of evolution and genetics.

Keywords: Genetic Algorithm, Logistics Engineering, Evolutionary Algorithms.

RESUMO

O trabalho de pesquisa descrito neste artigo tem como objetivo principal investigar a eficácia do Algoritmo Genético aplicado na solução de um problema de Engenharia Logística. Além disso, o artigo apresenta uma densa teoria sobre o assunto a fim de contribuir com pesquisadores da área. Com estes objetivos claros, a composição deste artigo irá inicialmente esclarecer os conceitos técnicos utilizados. Posteriormente, é apresentado o problema que será resolvido bem como sua modelagem, de forma que ao final seja apresentado um algoritmo, com foco na construção da lógica do algoritmo, bem como os dados obtidos que comprovam a eficácia da ferramenta como uma forma de resolver o problema definido. Os conceitos por trás do algoritmo aqui utilizado são derivados dos estudos mais recentes em Inteligência Artificial e são baseados em estudos biológicos da teoria da evolução e da genética.

Palavras-chave: *Algoritmo Genético, Engenharia Logística, Algoritmos Evolutivos.*

RESUMEN

El trabajo de investigación descrito en este artículo tiene como objetivo principal investigar la efectividad del Algoritmo Genético aplicado en la resolución de un problema de Ingeniería Logística. Además, el artículo presenta una teoría densa sobre el tema con el fin de contribuir a los investigadores en esta área. Con estos objetivos claros, la composición de este trabajo permitirá aclarar inicialmente los conceptos técnicos utilizados. Posteriormente se presenta el problema que se resolverá así como su modelado, de manera que al final se presenta un algoritmo, enfocándose en la construcción de la lógica del algoritmo, así como los datos obtenidos que comprueban la efectividad de la herramienta como herramienta. forma de resolver el problema definido. Los conceptos detrás del algoritmo utilizado aquí, se derivan de los estudios más recientes sobre Inteligencia Artificial y se basan en estudios biológicos de la teoría de la evolución y la genética.

Descriptores*: Algoritmo genético, Ingeniería logística, Algoritmos evolutivos.*

INTRODUCTION

The development of science has accompanied mankind since its first rupestrian records and the advent of fire and, since then, it has never ceased motivated by the interest in new discoveries and the desire to be able to control the world around it. So many ages have passed and man's scientific thinking has gained more space and recognition from society, and his willingness to modify and understand the environment in which he lives has made him achieve very important advances in several areas.

After many years of study, science has enabled the creation and use of tools that help humanity to solve specific problems, such as: performing mathematical calculations (calculators), observing the stars in the sky (telescopes), perceiving organisms not visible to the naked eye (microscopes), and so on. These equipments were created in order to fulfill, a task that the human body and intellect would not be able to achieve, at least not in a short time. With the passing of the ages, the instrument relationship between man and machine gained new concepts, the objective of such tools became more complex becoming an object of study for the development of something that, now, no longer operates simply by developing some skill human, but acting like a human being in solving a problem.

In this context, any and all algorithms created to date can be conceptualized, however, the ones that come closest to this objectification are the evolutionary algorithms, more specifically the genetic ones. According to Linden, R. (2012), Genetic Algorithms (GAs) are a branch of evolutionary algorithms, and as such, they can be defined as a search technique based on a metaphor of the biological process of natural evolution. Through this tool, the solution of intractable problems gained a new form of resolution. Intractable problems are thus named due to the time needed to solve them, such problems comprise fundamental issues of some technologies currently used, such as search sites, logistics system managers, geolocation software, among others. Such tools have become important pieces in the execution and functioning, of several areas of the society, as the researchers who use the GA of the search sites or the food distribution companies that use GA to optimize their logistic network.

Thus, the purpose of this paper is based on the need to optimize processes related to logistics, using heuristic techniques that have a potential for global optimization and with the potential to find a solution considered optimal for the problem. In this context, the simple process of cargo transportation can be thoroughly analyzed so that an optimization solution can be designed and implemented in order to save time and resources. In addition, this paper also seeks to provide a broad theoretical foundation on GAs in order to provide conceptualization to researchers who perhaps consider GA as a solution tool for defined problems.

Therefore, this paper proposes the use of GAs as a tool to solve the problems faced by the logistics sector, which are apparently simple, however, a great effort is needed to determine, for example, the best location of a distribution center that has a number of locations to be served, with specific routes configurations. The perception of the complexity that the problem takes when working with a large number of locations, if an attempt is made to use an exact solution, makes GA considered one of the best solutions to the problem, bringing an answer in acceptable, efficient and optimized time to the problem. Thus, this paper presents the development and use of a computational tool, through the Python programming language, using GA applied in the solution of the Knapsack Problem (KP), with the necessary adaptations in order to become a problem involving the loading of a truck. This application aims

to optimize the logistics processes, in order to save time and financial resources in a situation that simulates a real case. So this article aims to provide an important contribution to the scientific community and to companies through the following specific objectives:

- Demonstration of the functioning of a basic GA, in addition to its application to the NP-Hard problem of the KP where more refined knowledge will be used.
- Present the composition of the tool in a current programming language.
- Generate a solution that will be useful when applied in real situations, and
- Contribute to the development of process optimization in the engineering areas.

In order to achieve the specific purposes, this paper is structured as follows. In this section, the context that justifies the work was presented, theoretically presenting the problem and the tool used as a solution. In the next section will present the NP-Hard problem of the KP which is analogous to the problem that will be modeled. The elucidation of all concepts regarding GA's is presented in section titled as: Genetic Algorithm. Posteriorly, will discuss the order of the processes that will lead to the making of the algorithm. Then, the tests and results are presented, and in the finally section are presents the conclusion and suggestions for future work.

THE KNAPSACK PROBLEM

Initial Considerations

Intractable or too complex problems, due to its enormous amount of data, are common in nature and areas of society. Problems of this nature can be classified into two main types defined as (LODI and MONACI, 1990):

- Easy Problems (EP): when the solution to problem can be found using polynomial algorithms.
- Hard Problems (HP): these are problems where the only known solutions to solve them are exponential algorithms.

This class of denomination indicates the complexity of the time required to find the solution to the problem. A function can be defined that determines the complexity for the two types of problems, being respectively (CROWDER et al., 1983):

$$\textit{Complexity function EP} = S(p(n)), \quad (1)$$

in which $p(n)$ is a polynomial .

For this function, it is remarkable how complexibility will vary in magnitude in relation to EPs. For any problem of this nature, the increase or decrease in complexity will be polynomial.

$$\textit{Complexity function HP} = S(m^n), \quad (2)$$

in which $n > 1$.

As for the HP complexibility function, the degree of complexity of the problem will vary exponentially and cumulatively. The most common example of a problem of this nature can be cited as the Traveling Salesman Problem (TSP) (SHAYANFAR and SCHONFELD, 2019), which has a function of complexity equal to $(S(n!))$. The factorial growth of the complexity of this function demonstrates that the problem is not feasible to be solved by common methods (BURIOL, 2000).

Within these classifications, made in order to segregate some problems of statistical nature in different groups, there are the classes of problems P (Problems), NP (Non-deterministic Problems) and NP-Complete. The problems classified as P are those whose solution can be found in polynomial time and are considered treatable. The NP problems are the problems in which the determination of the most adequate solution to the problem is not viable, being, therefore, only verifiable in polynomial time. The problems classified as NP-Complete have a characteristic that, if one of the instances of the

problem can be solved, then the whole problem can be solved in polynomial time (RAJKANTH et al., 2017).

Problem Definition

This subsection will present the concepts that define the problem that will be used throughout the work as a basis for modeling the algorithm and its elements. Within these concepts are its theoretical and mathematical definition that represent the character of the problem for bases that can be modeled and applied to real cases (SUMETTHAPIWA et al., 2020).

Theoretical definition

According to Marques (2004), KP can be defined through the assumption that among several items available, a climber should carry his Knapsack with some of these items, taking into account the maximum bearable capacity. In addition, each item has a utility value and the climber must select them seeking to maximize the total utility value. Modeling the KP mathematically, the following data can be considered (GOTTLIEB and RAID, 1999):

- n = number of items available;
- v_i = utility value of the item, where i=1, 2, ..., n;
- p_i = weight of the item, where i = 1, 2, ..., n;
- l = Capacity of the Knapsack.

Based on this definition, several other denominations and variations arose for this same problem, such as the Knapsack Problem 0-1 (KP 0-1)and the Entire Knapsack Problem (EKP), which are the cases that will be used as a basis for modeling the algorithm in this paper. According to Marques (2004), KP 0-1 is one of the most common cases of KP, but with the restriction that it is possible to select only one item from those available, thus, it is one of the most studied discrete optimization problems, due to some factors such as:

1. Can be viewed as the simplest integer optimization problem;
2. It appears frequently in other more complex problems;
3. Can represent a wide range of real situations;
4. Any entire linear optimization problem can be transformed into a KP 0-1;
5. Application in combinatorial optimization problems.

Mathematically this problem is composed of a decision variable, defined as x_i, where if the product is selected for the Knapsack the variable will receive a value of 1, otherwise it will receive a value of 0. Resulting in the following notation (SHAYANFAR and SCHONFELD, 2019):

$$x_i = \begin{cases} 1, if\ item\ i\ is\ selected; \\ 0, otherwise. \end{cases} \quad (3)$$

in which i=1, 2, …, n.

So that, through the selection of each of the items, the climber can maximize the value assigned to the Knapsack load, he must follow the criterion of maximum Knapsack capacity, resulting in the following expressions (TORRES-ESCOBAR et al., 2018):

$$Function\ to\ be\ maximized: \theta = \sum_{i=1}^{n} v_i, x_i \quad (4)$$

$$Under\ the\ conditions\ of: \theta = \sum_{i=1}^{n} l_i . x_i \leq L \quad (5)$$

$$\text{in which } x_i = 0\ or\ 1\ and\ i = 1, 2, \dots, n \quad (6)$$

The mathematical modeling of the problem turns it into a summation of the product of the item's utility values by the choice variable, returning the total utility value assigned to the Knapsack. This sum, however, must comply with the conditions of not exceeding the maximum capacity of the Knapsack, represented by equation (5) as well as following the conditions of limitation in which the element x_i can only assume value 0 or 1, and that the variable i will

from 1 to the number of items available (TENG and TZENG, 1996).

The second variation of the problem, resembles the example that will be used as a problem in this paper. Its definition is the one that best represents the problem in general, because there are no limitations on the number of items that is selected. This feature makes it possible to select more than the same element that is available, making it closer to a real case of positioning a certain cargo in a truck of available space *L*. Mathematically defining the problem, there will be, as in the previous case, a variable considered as the decision variable (SUMETTHAPIWAT et al., 2020):

$$x_i = \textit{number of items in the sample space i selected} \quad (7)$$

in which i = 1, 2 ,..., n

Using the expression of the variable in (7), the function that defines the functioning of the problem and the search for its solution can be modeled, considering equations (4), (5) and (8):

$$\textit{in which } x_i \geq 0 \textit{ and entire, } i = 1, 2, \ldots, n \quad (8)$$

Equations 4, 5 and 8 demonstrate the complete expression for EKP. The sum in 4 represents the utility value of the whole Knapsack at the end of the choice of items, following the conditions imposed in 5 and 8, which, unlike the previous case, now allow the loading of more than a single item of the same nature in the Knapsack, the amount of which is now represented by x_i which can take on any positive integer value (PINO et al., 2011). Therefore, using the concepts of NP, the KP fits as an NP-Complete problem and the problem, cannot be solved using just a polynomial algorithm, requiring the use of another non-deterministic methodology, such as the heuristic methods of evolutionary algorithms, that will be used to solve the problem covered in the Tests and Results section.

Genetic Algorithm

The purpose of this section is to provide the reader with a basis for the importance of studying GA, as well as the origin of the technique used in this work, which will be Evolutionary Algorithms. Furthermore, this section will provide the reader with support for studies related to the Computational Definition of GA, the Characteristics of GA, the Basic Elements of GA, the Processing of GA and finally, the Genetic Convergence and Mutation Operator (AZAD and HASIN, 2019; NAZIF and LEE, 2012).

Initial Considerations

The reproduction of concepts found in nature, through scientific studies for the creation of technologies that help in solving problems, is an old and common practice in society. When analyzing the phenomena related to genetics, through a mathematical and logical look, it is possible to absorb from the processes of crossing and genetic mutation a powerful scanning tool for solving problems, which have a basis in combinatorial analysis. According to Linden, R. (2012) GAs are a branch of evolutionary algorithms and as such can be defined as a search tool that uses a metaphor for the biological process of natural evolution. This use therefore generates tools that can be applied to the most diverse stochastic problems.

Evolutionary Algorithms

The analogy to the evolution process, arising from the theory of evolution, at the level of genetics is present in the production of the logic of the algorithms, since it acts on elements called individuals or chromosomes, which represent the elements of the problem to be addressed. These, in turn, are contained within the sample space of the problem, called population. To these structures are applied genetic operators, such as recombination and mutation, which will be responsible for making the necessary changes

on the individuals that are distributed in the sample space creating new individuals. An evaluation is applied to each individual that will quantify the quality of the individual as a solution to the problem. The process is then repeated with the action of the genetic operators until the solution found is satisfactory (PRINS, 2004).

As it is a system that works in a loop feature, each new repetition of the algorithm is given the name of generation. The association again is analogous to its equivalent in real life, since it is up to the system the analogy of two “Parents” elements, that will combine and generate “Sons” elements, as well as the elements "Sons" will combine with other elements to continue the process (ARDJMAND, 2020).

Computational Definition of GA

Within the concepts of evolutionary algorithm, another branch was born, inspired by Neo-Darwinian theories, which is the combination of three natural phenomena: species evolution, natural selection and genetic inheritance. Within this theory, the concept of evolution is inherent in the way life continues, so that it undergoes the action of some processes at the moment of the combination of two individuals of a population, which cause variations, namely: reproduction, mutation, competition and selection (FOGEL, 1995). Based on these foundations, studies on the computational modeling of these phenomena were formally initiated in the 1970s. The definition absolved through these studies, classifies GAs as a search method based on the fundamentals of natural biological evolution of species and as heuristic methods (MATEO and ALBERTO, 2018).

Computationally, GAs have as their search method the best solution to a specific problem, based on a logical computational cycle, composed of a population (sample universe), on which genetic operators (mutation and crossover) act, who perform combinations among individuals on whom an assessment will be imposed based on what would be the best solution to the target problem until the moment when a solution is evaluated as good (YAZDI et al., 2020).

Characteristics of GA

GAs have a specific set of characteristics, the probabilistic character being one of the main characteristics, in contrast to other deterministic methods, which have fixed solutions to a given problem. Unlike these methods, given an initial population and the same set of parameters, GAs return a different solution each time they are executed, bringing great advantages over conventional deterministic methods, factor makes them extremely applicable to real problems (AZAD and HASI, 2019).

Another important feature of GAs is that they work with a large number of points, further reinforcing their applicability in real problems. This characteristic also differentiates them from deterministic methods, as their research region will concentrate throughout the process in a small part of the sample space, sweeping only a subset of the universe of available solutions (ALI et al., 2013).

Basic Elements of GA

Some elements are common to all GA’s, regardless of their application, which are (AZAD and HASIN, 2019; HAUPT and HAUPT, 1998):

- Individuals: They are compositions generated by the code that carry information about the problem with it and that represent to the algorithm the possible solutions to the problem. These individuals are composed of chromosomes that represent the characters that make up the solution. In addition to the information brought by the chromosomes, there are attributes that define more fully the possible solutions to the problem such as: identifications, values, indicators, among others.

For example, with the TSP case, chromosomes would indicate which cities and in which order the salesman should go.

- Rating (fitness): The evaluation element, known as fitness, carries an important function of classifying possible solutions (individuals) according to some criteria, using data provided by these individuals and his evaluation method imposed, this element will therefore be the selectivity of the entire algorithm, returning at the end of the processing what he evaluated as the best answer to the problem.
- Crossover operators: Crossover operators are responsible for crossing the chromosomes of two individuals selected in order to generate sons and, through successive combinations, generate increasingly better solutions.

GA processing

The first stage of the GA process is performed on a specific group of individuals called of population, where information will be coded for the universe of binary data. The individuals represent the elements that make up the basis of the problem, (as is the case of this work, the NP-complete problem). These elements are very numerous, but they have certain characteristics shared among themselves, and when they are efficiently encoded, result in an increase in the efficiency of the solution, saving time and processing.

The next step in coding the population is the action of the operators, who took charge of combining possible solutions to the problem, guided by the character of the evaluation function. Within this process, the concept of genetic convergence or loss of diversity is of paramount importance for the construction of operators as an algorithm and has a great impact on the quality of the solution found (WANG and WANG, 2019).

Genetic Convergence and Mutation Operator

According to Linden, R. (2012) genetic convergence is the character of a population with low genetic diversity, which, due to having similar genes, cannot evolve, except for the occurrence of positive mutations and, however, the loss of diversity can be defined as being the number of individuals who are never chosen by the parent selection method. These elements are from the sample space that receives a low evaluation and, therefore, tend not to be selected by genetic operators for the creation of a new individual (BEASLEY, 1996).

To overcome this factor, mutation operators are used. These operators are responsible for modulating mutations that occasionally occur in nature and for providing characteristics that make certain individuals more capable of surviving than others (AZAD and HASIN, 2019).

METHODOLOGY

In this section, all the processes that will lead to the solution of the proposed problem will be presented. The research begins with the elaboration of the proposed problem, as well as the presentation of the concepts necessary for the understanding of the problem and its solution. After that, necessary concepts are presented for the structuring and foundation of the specialized research of GA, and finally, the methodology used to solve the proposed problem.

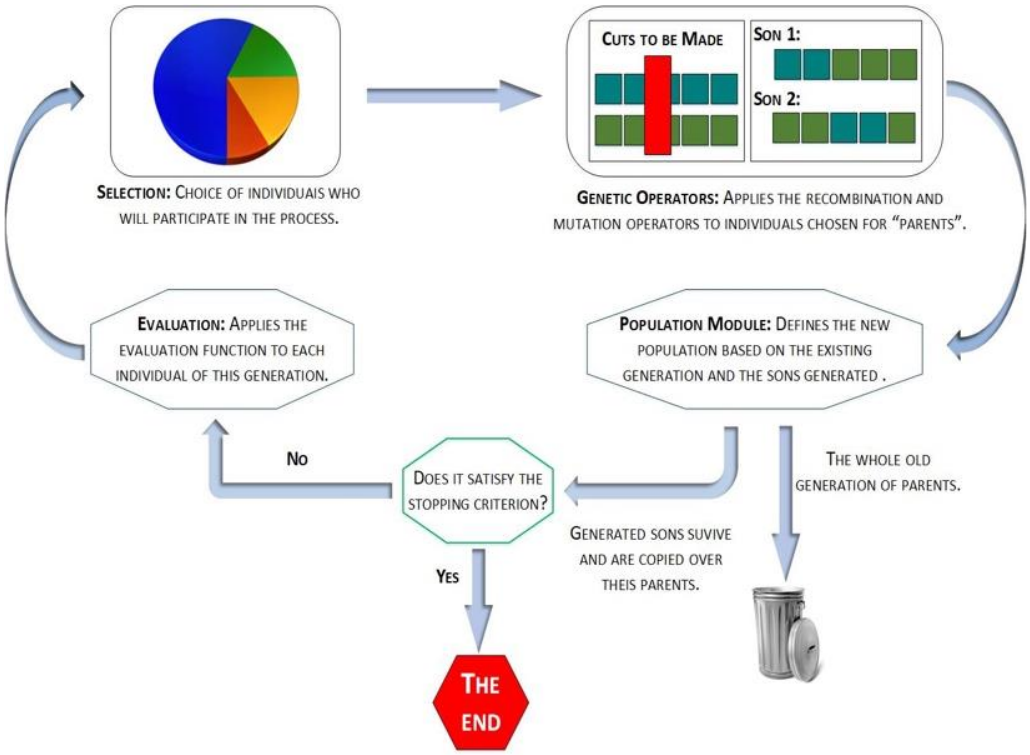
Introduction to the problem and the tool

The presentation of the concepts that involve the NP-complete problems and the KP were necessary because the problem to be treated, using the GA method, is analogous to those topics that were referenced in theory, and some parameters and conditions will be changed in the real case to be evaluated during this work.

In order to clarify the process and the structure that will compose the body of the created algorithm, a

basic scheme of the GA tool, is shown in Figure 1 (LINDEN, 2012). This structure will be replicated for the construction of the GA applied in this paper, and the concepts already presented will be translated into certain operations within the programming language chosen for the creation of the tool that was Python.

Figure 1. Scheme of a genetic algorithm.



Source: Adapted from Linden (2012).

The choice of using the Python tool was motivated by the facilities that this programming language has, for being a free and open source software, and has been gaining a lot space in the specialized academic community (SHEPPARD, 2016; POLI et al., 2008).

The problem to be treated

It is important to clarify the nature and conceptualization of the problem to be addressed. And in the case of this work is to solve a logistics problem, which has the following description: a driver wants to allocate a certain load of products, where each product has a certain value in dollars ($) and volume in meters cubic meters (m^3), in a truck with total volume of L cubic meters. The driver's objective is to allocate a load of the highest possible value inside the truck, using as little space as possible, thus maximizing its loading efficiency.

The problem requires that product data, such as values and volumes, be reported by the operator, as well as the total capacity of the truck. These data will be crucial for the algorithm since these are the criteria that it will use to define the best solution to the problem.

In order to make the system more dynamic and close to reality, in this paper two different variations will be made, where in one of the tests the load available to be located inside the truck will contain only one unit of each available product, and in the second test, it will be made available different quantities of each product to be worked. Thus, the objective of GA will be to determine the best possible solution for loading the L-volume truck, with the

products that are informed by the operator. Finally, GA will generate information on what will be the best solution to the problem, as well as its generation, products that have been selected, price, occupied space and a graphical mapping of the development of the best solutions over the generations covered by the algorithm.

Modeling

Modeling any situation or problem in the real world for the language used by the machines requires the modeler to think strategically about the problem, so that the whole problem will have a structure where its parts are interconnected from beginning to end. The basis used for assembling this structure is the schematic shown in Figure 1, and the resulting algorithm passes through a satisfaction criterion that will evaluate each individual of the generated population, select them for the crossing, apply the genetic operators of crossing and mutation and then will discard the old population replacing it with the descendants of the previous one. This entire logical process is repeated as long as the stopping criterion is not met, and these stopping parameters are defined by the user. Figure 2 shows this GA scheme, called abstract (SUMETTHAPIWA et al., 2020).

Figure 2. Abstract Genetic Algorithm.

```
# Abstract Genetic Algorithm

1 Receives stopping criteria
2       Initializes Population
3       Evaluates Population
4       while stop criteria not met
5               Select Parents
6               Applies the crossover operator
7               Applies the mutation operator
8               Evaluates the population
9               Discard the old population
10              Defines the new population
11
12      # Once the stop criterion is satisfied
13      List the best individuals
14
15      End of Algorithm
```

Source: Authors.

All of these ordering of actions come from the logic of programming abstracted from the evolutionary processes arising from biology. Each of the steps represents a step analogous to those found in nature, and all of them are integrated to form a system that, based on the assessment of individuals present in the population, will perform manipulations in a continuous way, in order to evolve more and more the "species" until the stopping criteria are met (BURIOL, 2000; BEASLEY, 1996; MATEO and ALBERTO, 2018).

Individual modeling

The main basis for the elaboration of the algorithm described in this article is called Object Oriented Programming (OOP) (MILLIKEN, 2020), which aims to model real-world tasks for computer language. The use of OOP to perform these tasks consists of assigning to the given class (which represents the object to be modeled) the necessary attributes so that the algorithm can understand what this class can store, execute and assign, as shown in Figure 3. In it, the class is represented with its proper attributes, which will be used and manipulated during the execution of GA by other classes or functions.

As it represents the first element of the problem, the modeling started by representing a truck and its possible loads. To represent this set, it is necessary to first model the items for the algorithm, represented by the Product class shown in the abstract algorithm of Figure 3.

Figure 3. Defining the Product class (Abstraction).

```
1  class Product ( ):
2      receives the attributes (name, space, value):
3              stores in x = name
4              stores in y = space
5              stores at z = value
```

Source: Authors.

Therefore, the class created in the algorithm that represents the items must receive the assignments of each item that can be allocated inside the truck, and receive characteristics of the value and volume, which are factors that later on the algorithm will use to evaluate each of the products and their inclusion or not in the truck.

After defining the objects that make up the problem data (Products), it is necessary to model the individual, which will be the exact representation of the solution possibilities for the problem, applied to each of the situations addressed. For the case, from the allocation of several products in a limited space, the modeling of individuals (solutions) is summarized in a binary distribution, which has as many elements as available products, with each product being assigned a number in that distribution, and that represents, literally, if a certain product will be taken or not. For this binary distribution, the products to which the number "0" is assigned will not be taken in the truck and those that receive the number "1" will be taken (TENG and TZENG, 1996).

The archetype of each individual or solution will have this binary distribution format, since, given a certain number of items, the pattern of the solutions will maintain the pattern of binary sequences, which make an analogy to the individual's "chromosome" modeled for the problem. This association helps in modeling the individual within the algorithm as well as its manipulation by the crossing and mutation operators (SHAYANFAR and SCHONFELD, 2019).

Furthermore, as shown in Figure 4, the class created for each individual must receive certain parameters that allow the algorithm to recognize and operate the values of each product in order to find the best answer.

Figure 4. Defining the Product class (Abstraction)

```
1 class Individual ():
2     receives the attributes (space, values, limit_spaces, rating, used_space, generation):
3             stores in a1 = space
4             stores in a2 = values
5             stores at a3 = limit_spaces
6             stores at a4 = generation
7             rating = attributes to the individual
8             used_space = attributes to the individual
9             chromosome = row_matrix
```

Source: Authors.

The class defined in Figure 4 brings with it all the information that the GA will need from the individual who is trained through it. Each of the characteristics shown are defined in order to fully model the load that will be carried in a truck of capacity to be defined by the user. This class is composed of some information that is necessary for the GA to carry out the evaluation of the individual generated, the verification of the possibility of that individual being a viable answer, in addition to being able to perform operations such as crossing and mutating this individual with others generated in order to find the best possible answer.

Finally, the individual class has as its attribute its primordial composition, which is the chromosome, represented by a single row matrix and several columns, with zero and one elements. The term chromosome derived from biology translates in an analogous way what this element represents for GA, which would be the expressed characteristics themselves, inherent only to that individual, and which can be combined with others, or mutate, generating descendants.

Evaluation modeling (Fitness)

Through the assessment of the individual GA will be able to select the best individuals, in order to make the crossing operators act on them, in an attempt to find even better results. The GA evaluation can be done through several methods, and in this work it was

made based on the Elitist Method, which defines the best solutions based on the "rating" parameter, ranking the solutions with the highest rating for the that has the smallest.

Figure 5 shows the abstraction of the evaluation function, which is responsible for the evaluation within the algorithm and performs the sum of the utility values of the Individual's items and at the end it will verify if it corresponds to a viable solution for the problem through the condition imposed by the space limit of the truck. So that non-viable solutions have no value as good solutions within the sample space, whenever an Individual exceeds the space limit defined by the truck's available limit, this solution receives a low grade, defined for this work as being of unit value, that individual therefore, it will not be excluded from the population, but it will have minimal chances of being selected to generate descendants (AZAD, 2019).

Figure 5. Function Evaluation (Abstraction).

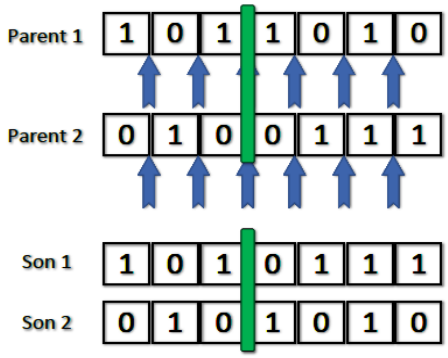
```
1  function Evaluation ( ):
2      if  gene on Individual's chromosome = 1
3          rating = sum of the utility values of the items
4          used_space = sum of the spaces occupied by the items
5
6      individual's grades = assessment grade
7      individual's space = space used
```

Source: Authors.

Crossing operator modeling

The crossing operator will be the GA method responsible for ordering the crossing of the chromosomes of the two Individuals selected as parents. The crossing of two "parents" tends to generate descendants characterized as more apt solutions and, with the advancement of generations, the population tends to evolve in quality of solution (SHI et al., 2020). For this work, the use of the one-point crossing method was defined, where, considering the chromosome of each individual, a random point is selected so that the crossing occurs in an alternate symmetrical way, as shown in Figure 6.

Figure 6. One-point crossing.



Source: Adapted from Granatyr (2018).

Therefore, after selecting a randomly defined point in the chromosome chain, the operator performs the genetic exchange between the Parents' genes to generate the descendants. As shown in Figure 6, Son 1 receives the first three genes from Parent 1 and the remaining genes from Parent 2. In the formation of Son 2, it receives the first three genes from Parent 2 and the genes remaining from Parent 1 (SHI et al., 2020).

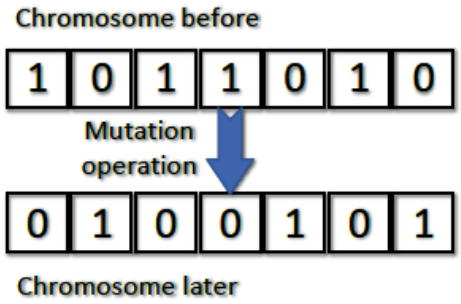
Mutation operator modeling

Modeling what would be the mutation process that occurs in nature is quite different in the field of algorithms, since this method is performed on the individual's chromosomes and any alterations in genes, greatly affect the solutions represented by the individuals.

The process shown in Figure 7, literally illustrates the process that occurs within the mutation operator used to compose the GA of this work. The operator acts on the genes of the chromosome by inverting them, therefore, for the genes of value "0", their value will be changed to "1" and vice versa. This operator is linked to an occurrence rate defined by the user, which is usually pre-determined between 0.5% and 1%, influenced by the good results obtained in previous works (SHI et al., 2020; SHI et al., 2018).

This rate will define the frequency of action of this operator on individuals in the population, more specifically on the children of a given generation, which will be when the mutation operators will come into action.

Figure 7. Mutation Operator.



Source: Authors.

It is worth mentioning that for greater efficiency, the ideal is that tests be carried out to find the best mutation rate applied to each problem.

Construction of the algorithm

Having elucidated all the elements, the GA needs to unite them so that, through a continuous process and within the generated sample space, it is able to seek the best solution to the problem.

Thus, in this work, GA was organized in a certain order of action as in the scheme shown in Figure 1, where GA was formed by an algorithm that uses all the elements previously modeled, placed in a conditional loop that will have a stop condition the number of generations determined by the user. Essentially, the algorithm must generate a population of solutions and, after that, evaluate these solutions and then apply the operators on that population in order to direct it to the best possible optimization, repeating the cycle until the stop condition.

Select parents method

This method is the function within the algorithm responsible for the selection of individuals within the population, which will be those whose descendants will replace the old one, on the pretension of being better solutions to the problem. Although the purpose of the algorithm is always to search for better solutions, as there are randomized processes within the algorithm, the descendants of these individuals will not always be better solutions to the problem, however, GA must be able to use these individuals to generate even better solutions. to the problem as it passes through processing. The basis for building the method that selects parents will be that of addicted roulette, which consists of selecting individuals based on their rating. This methodology aims to reproduce the method of natural selection that acts on species in nature, where more able parents generate descendants more often than less able parents, making the characteristic of the best individuals predominate in the new population formed by the descendants (MUSTAFI and SAHOO, 2019).

GA abstraction

The code was assembled in Python language, using the OOP methodology for its construction. Then a class called “Genetic Algorithm” was created that will carry with it all the necessary functions for the execution of the logic shown in Figure 8 and with the intention of performing the following steps:

1° Initialize the population.

2° Sort the population by rating (in descending order).

3° Define the best Individual in the current generation.

4° Perform the sum of the evaluation scores of individuals in the population.

5° Select the parents using the addicted roulette method.

6° Performs the crossing process through the crossing operators.

7° If determined, perform the mutation process through the operator.

8° Replaces the old population with the new one.

9° Evaluates the new individuals of the population.

10° Orders the population based on the elitist method.

11° Allocates the best individual in the first position of the population.

12° Displays the current generation stating: the best individual, his total utility value and his total space occupied.

The algorithm follows in its execution this logical sequence, where with each new action of the crossing and mutation operators, it is possible to visualize the best solution found by them. The results will be displayed through graphs according to the progress of the population count that will serve as an evaluation of the performance of the algorithm.

In Figure 8, it is possible to visualize the union of all methods, organized in the process chain, which results in the main set of the problem solving tool in this paper. The abstraction of the GA shown is intended to clarify the cycle of steps covered by the algorithm in search of the solution to the problem.

Figure 8. Scope of the Genetic Algorithm used (Abstraction).

```
1   class GeneticAlgorith ( ):
2           receives the attribute (population size):
3           stores at x = population size;
4
5           executes the function Initializes the population;
6
7           executes the function Sorts the population;
8
9           performs the sum of ratings;
10
11          executes the function Select parents;
12
13          run the Crossover method;
14
15  if condition matched:
16          Performs the mutation method on the generated descendants;
17
18  Executes the function Sorts the population;
19
20  It allocates the best individual in the first position of the new population;
21
22  Displays the current generation, the best Individual its total utility value
23  and used space;
24
25  if the stop criterion is matched
26          End of execution
```

Source: Authors.

The user defines the “population size” data received by the GA, which stores it for use within the function that generates the initial population. This variable will define the number of individuals that the populations will have throughout the processing of the GA. With the initial population generated, the ranking of the best individuals is made, even if they do not correspond to viable solutions, so that the sum of the evaluation of each individual can then be made. The next step of the algorithm consists of applying the methodologies, where the addicted roulette method will be applied to select the parents, in addition to the action of the crossing operators to perform the combination of these parents, and, if determined by the mutation rate, the action will occur of the mutation operator on the descendants generated. This process of generating a new population is followed by its ordering and evaluation based on elitism followed by its display on the user's screen, thus ending a generation of the algorithm. Bearing in mind that the execution cycle lasts as long as the algorithm does not match the stop condition, which is represented by the number of generations determined by the user.

TESTS AND RESULTS

The use of GA as a solution tool for the proposed problem, due to its stochastic nature derived from the random combinations made by the algorithm in order to find the best answer, presents different results in repeated runs that may or may not turn out to be considered optimal solutions. to the problem. Due to this characteristic, the results were evaluated according to their utility values and occupied space, and the purpose of the experiments carried out with the algorithm was to bring these values closer and closer to the best possible conditions. Therefore, the objective is that the utility values are maximized and the space occupied by the best solution found is as optimized as possible.

The machine on which the GA experiments were carried out has a Processor: Intel® Core™ i3-

2100 CPU@3.10GHz 3.10GHz, Memory (RAM): 4.00 GB, System type: 64-bit Operating System and Python Version: 3.7 64-bit.

Tests Preparation

The conditions imposed on the algorithm, such as population size, space limit supported by the truck, number of generations, mutation rate, and the characteristics of the items with which the algorithm will have to work were defined based on the nature of the experiments carried out. The aim is to adapt the characteristics of the algorithm in the best possible way according to the nature of the problem being treated.

As proof of the method's effectiveness, applications were initially made for GA implemented over data sets available online and which are regulated by the GNU LGPL license (open access databases). These applications range from problems related to combinatorial analyzes, to those involving NP-difficult and NP-complete problems. The purpose of using these databases is to assess the efficiency of the developed algorithm while acting on one of the chosen cases, so that, later on, the algorithm can be applied in situations created and adapted to real cases of the logistics of transport companies.

Problem P01

In the first experimentation with the algorithm presented, the database chosen was P01, present in the Data for the 01 Knapsack Problem, that brings with it the characteristics of Table 1.

The sampling of data shown, simulates a certain sampling of items, which need to be arranged in a space limit of value 165 (Truck Limit). The nature of the measures involved in the problem does not necessarily need to be defined, that is, it is understood that the Limit value will have the same unit of measurement as the weight or space occupied by the items, plus the utility values of each item indicate only the assignment that each one adds to being taken in the "truck".

Table 1. Problem Description P01

PROBLEM P01		
	Truck Limit	**165**
Item	**Weight/Space occupied**	**Utility Value**
1	23	92
2	32	57
3	29	49
4	44	68
5	53	60
6	38	43
7	63	67
8	85	84
9	89	87
10	82	72

Source: Authors.

The best ideal solution defined by the database for this problem has the following form:

Table 2. Parameters used for P01.

IDEAL SOLUTION PROBLEM P01	
Chromosome	[1 1 1 1 0 1 0 0 0 0]
Total Weight/Occupied Space	165
Utility value	309

Source: Authors.

It is important to note that the database used here informs the best solution to the problem P01, which is the allocation of items 1, 2, 3, 4 and 6 in the truck. This solution provides the highest utility value for the load in the most optimized weight or space used. The intention is that the constructed algorithm is, therefore, capable of reaching this solution or approaching with little variation of the described values.

Test 1

The parameters used initially to run the algorithm on the problem were:

TEST 1	**Population Size** = 20 **Mutation rate** = 0.01 **Number of Generations** = 100

Using these parameters of problem P01 for processing the algorithm, after repeating the same execution for a low number of attempts, GA returns the best solution to the problem. Figure 10 shows the results of the best solution found after its execution, informing the number of the generation in which it was found, the total utility value of that individual, the space it occupies, the representation of its chromosome, the description of the chosen items for that solution and the time it takes for GA to process this answer.

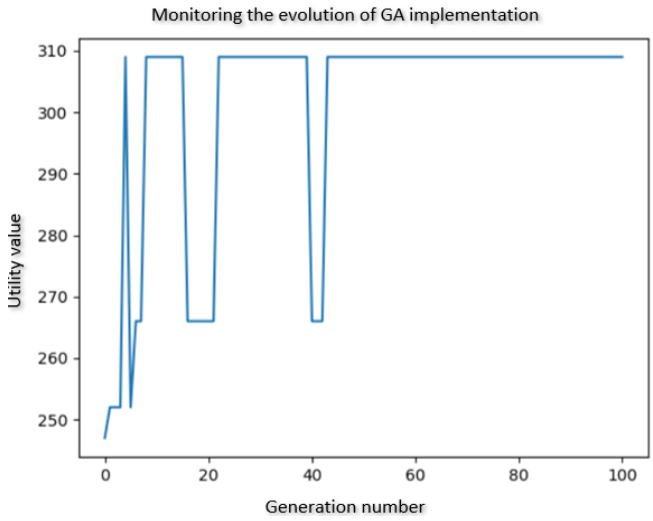
Figure 10. Solution for case P01 found by GA.

```
Best solution -> Generation: 11  Value: 309  Space: 165
Chromosome: ['1', '1', '1', '1', '0', '1', '0', '0', '0', '0']
Name: Item 1 $ 92
Name: Item 2 $ 57
Name: Item 3 $ 49
Name: Item 4 $ 68
Name: Item 6 $ 43
Operation duration: 0.056658
```

Source: Authors.

The graph represented in Figure 11 shows the development of the solutions, using the utility values of the best solution found in each generation over the generations. Through it, it is possible to observe the behavior of the algorithm throughout its processing in its search for the best possible solution to the problem.

Figure 11. Graph of the solution for the case P01 found by the GA.



Source: Authors.

As can be seen in Figure 11, the test parameters presented here resulted in the ideal response, that is, the same response informed by the database itself (P01 Knapsack Problem) as being the ideal response.

Test 2

So in order to bring more dynamism to the tests, the parameters of the algorithm were changed to:

TEST 2	**Population Size** = 20 **Mutation rate** = 0.02 **Number of Generations** = 100

Through this change in the parameters, it was possible to observe that by varying the number of generations to be processed by GA, the ideal response is still obtained at a higher frequency than in the previous experiment, thus reducing the number of repetitions necessary for an ideal response to be found for the problem. However, this change causes, as expected, an increase in processing time, however GA continues to work in less than 1 second. The results obtained by the algorithm are shown in Figures 12 and 13.

Figure 12. Solution for case P01 found by GA.

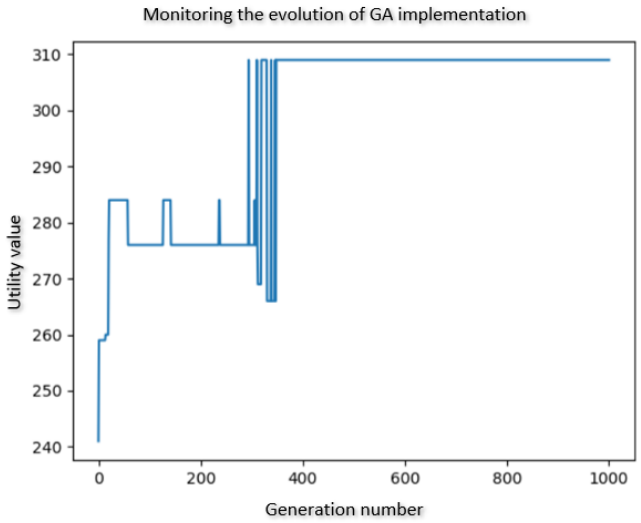
```
Best solution -> Generation: 294  Value: 309  Space: 165
Chromosome: ['1', '1', '1', '1', '0', '1', '0', '0', '0', '0']
Name: Item 1 $ 92
Name: Item 2 $ 57
Name: Item 3 $ 49
Name: Item 4 $ 68
Name: Item 6 $ 43
Operation duration: 0.524429
```

Source: Authors.

As can be seen in Figure 12, after changing the parameters, GA also returned the result considered ideal. As well as the graph shown in Figure 11, the graph represented in Figure 13 shows the development of the solutions, making it possible to observe the behavior of the algorithm throughout its processing, in

search of the best possible solution to the problem. The heuristic behavior of the algorithm in these two solutions is remarkable, where, for both cases, the best solution was found in a low order generation as a function of the whole. For the first case, the best individual was found in generation 11 of 100 and for the second experiment, it was found in generation 294 of 1000.

Figure 13. Graph of the solution for the case P01 found by the GA.



Source: Authors.

Truck Loading

Once you have gauged the effectiveness of the algorithm built with an example provided by Data for the 01 Knapsack Problem, it is possible to apply GA in a real situation, to analyze the behavior and results obtained by the tool under certain conditions. The case to be studied is the loading of cargo in a truck of available volume *L*.

The situation has as its only definition of parameters, a list of products shown in Table 3, with different utility value and occupied space characteristics, and the total available space of the truck that will be defined within the algorithm.

In this Table 3, we list all the products that are available to be taken in the space/weight limit truck equal to 100. Some of the products are repeated to also simulate the case of the Knapsack problem, where the quantity of each of the available items it also varies, which brings a representation to the actual loading cases of the transport companies. Several tests were carried out on the situation set up in order to obtain the best possible result for the problem, therefore, the parameters of population size, mutation rate and number of generations were changed several times.

Table 3. Truck Loading.

SITUATION - TRUCK LOADING		
Truck limit		**100**
Item	**Weight/Space occupied**	**Utility Value**
43" LED TV	9.1	1799
Air conditioning	31	1038
Coffee machine	3.6	260
Computer	12	1646
Computer	12	1646
Cooker	32	999
Microwave	16	648
Washing machine	34	1254
Ventilator	4.1	169
Vacuum Cleaner	5.9	389
Blender	1.8	139
Blender	1.8	139
Fridge	58	2399
Fridge	58	2399
Video game	4.35	1535
Video game	4.35	1535
Video game	4.35	1535
Notebook	1.95	2008
Smartphone	0.2	2649
Smartphone	0.2	2649

Source: Authors.

Test 3

The parameters used to execute the problem algorithm proposed for the first case were:

TEST 3	**Population Size** = 20 **Mutation rate** = 0.01 **Number of Generations** = 100

The results obtained through the tests as well as the simulation graph are shown in Figures 14 and 15.

Figure 14. Solution for the Truck Loading case.

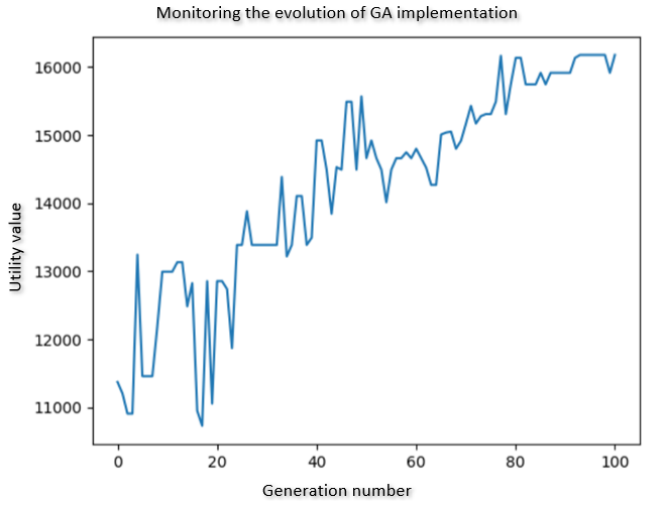
```
Best solution -> Generation: 93  Value: 16178  Space: 97.0999999998
Chromosome: ['1', '1', '0', '1', '1', '0', '1', '0', '0', '0', '1', '0', '0', '0', '1', '1', '1', '1', '1', '0']
Name: 43" LED TV $1799
Name: Air Conditioning $ 1038
Name: Computer $ 1646
Name: Computer $ 1646
Name: Microwave $ 648
Name: Blender $ 139
Name: Video game $ 1535
Name: Video game $ 1535
Name: Video game $ 1535
Name: Notebook $ 2008
Name: Smartphone $ 2649
Operation duration: 0.044264
```

Source: Authors.

Using the parameters for test 3 (Truck Loading) after executing the algorithm (similarly as it was done for the execution of tests 1 and 2), GA returns the best solution to the problem. Figure 15 shows the results of the best solution found after its execution, informing the number of the generation in which it was found, the total utility value of that individual, the space it occupies, the representation of its chromosome, the description of the items chosen for this solution as well as its value and the time required for GA to process that response.

Through the graph shown in Figure 15, it is possible to observe the development of the solutions, using the utility values of the best solution found in each generation over the generations.

Figure 15. Graph of the solution for the Truck Loading case.



Source: Authors.

The tests done with these parameters (test 3) showed that GA has a high capacity to solve problems inherent to transportation companies. In this first test it can be observed that the algorithm was executed with extreme speed (time of 0.044264 seconds). It is also observed that the value of the cargo allocated is 16178 dollars. Despite these relevant results for a first test, there was still a 2.9 space left on the truck. In the next subsection, the parameters will be changed in order to obtain an optimization of space and value.

Test 4

The parameters used to execute the algorithm in this test on the problem, in a second attempt were:

TEST 4	**Population Size** = 20 **Mutation rate** = 0.02 **Number of Generations** = 1000

Figure 16 shows the results obtained through the tests and Figure 17 shows the graph of the respective simulation.

Figure 16. Solution for the Truck Loading case

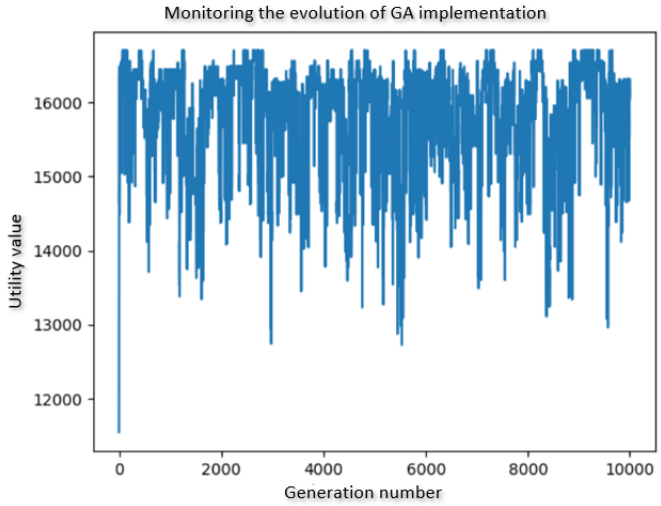
```
Best solution -> Generation: 68  Value: 16703  Space: 99.499999999
Chromosome: ['1', '0', '1', '1', '1', '0', '0', '1', '1', '1', '1', '1', '1', '0', '0', '1', '1', '1', '1', '1', '0']
Name: 43" LED TV $1799
Name: Coffee machine $ 260
Name: Computer $ 1646
Name: Computer $ 1646
Name:  Washing machine $ 1254
Name:  Ventilator $ 169
Name: Vacuum Cleaner $ 389
Name: Blender $ 139
Name: Blender $ 139
Name:  Video game $ 1535
Name:  Video game $ 1535
Name:  Video game $ 1535
Name:  Notebook $ 2008
Name: Smartphone $ 2649
Operation duration: 5.194111
```

Source: Authors.

The results shown in Figures 16 and 17 are quite explicit in relation to the proximity of the solutions, which indicates that the algorithm has a certain tendency to always obtain the most optimized result possible, however, there are other details present in the simulations that are notable for the objectives of

this work. When analyzing some repetitions of the algorithm execution in the configurations previously exposed, where there was a difference in the population size from 100 (test 3) to 10000 (test 4), and it can be noticed that when increasing this value of the algorithm, being provided with a space of repetitions and greater search attempts, excels in finding the best answer to the problem.

Figure 17. Graph of the solution for the Truck Loading case



Source: Authors.

For this case, the best response after tests was shown in Figures 16 and 17, where it was observed that it was possible to load 3 more products in the truck, resulting in an individual's total utility value of 16703, so there was an increase of $525. Now, the occupied space/weight is 99.4999 showing an optimization of space in 82.76%. This solution can be obtained in tests with a smaller population size, however, it appears after a few executions of the algorithm, but the same does not occur when increasing the value of the population size. Although the greatness of this number caused the algorithm to finish the process in a considerably longer time compared to the population size experiment equal to 100, the time in which it was performed (Duration 5.1941 seconds) is considerably fast, once the process ends up returning what would be the best possible solution using this technique, in a much larger range of assertiveness as evidenced by the results.

Comments

In this chapter, the tests and results obtained through the application of the proposal were presented and shown in section The Knapsack Problem together with the technological strategy of the Genetic Algorithm and the methodology presented. The objective of these tests was to evaluate which is the best KP solution, with the necessary adaptations in order to become a problem involving the truck Loading.

Through the analysis of the behavior of the graphs in Figures 11, 13, 15 and especially in Figure 17, it is possible to notice that with the processing of the algorithm, the system reproduces in a similar way the process of natural evolution, that is, the variation shown has character increasing and denotes a search for continuous evolution of the algorithm until the end of processing. It is also possible to notice through the graphs shown in Figures 13 and 15, that due to the smoothness of the data, when the GAs faced the individuals corresponding to the ideal solution to the problem, there were some unsuccessful attempts to search for even better solutions, as shown by the variations by the alternating peaks. This characteristic is beneficial for the algorithm because it denotes its statistical capacity to continuously search for better solutions through the use of crossing and mutation operators.

CONCLUSION

The Genetic Algorithm proved to be a powerful tool in solving combinatorial problems impossible by common methods, which further elevates its value as a methodology, since large parts of the logistical, transport and industrial processes problems have permeated in their nature combinatorial

problems. The quick solution of the problem described here, also allows to classify the modeled tool as effective not only for the case set up for study, but also for similar application examples. Through the results shown, it was possible to achieve one of the main objectives of this paper, which was to investigate the effectiveness of the genetic algorithm applied in solving a problem in logistics engineering.

Future works

The text of this paper brought a theoretical foundation to the researchers in order to enable the implementation of GA in other problems or even to improve the problem exposed here. As a suggestion for future work, using the tool and its application demonstrated in this work, we can mention the application of the algorithm in the solution of a complete logistics problem, where the entire process of cargo transportation, goods flow, delivery control, among other similar processes, GA would be responsible for deciding the most efficient path to be taken among the various possibilities present in these processes. The tool shown here can also be increased with the advent of several technical elements, since, if provided and modeled correctly the data, GA will find an optimal solution for any combinatorial problem. Another suggestion would be to develop a mobile app of the algorithm in order to make it a more intuitive and illustrative tool, adding to its operation a clear and direct interface so that any user can operate as well as manipulate the display of results.

As noted, the application of GAs is not limited to just one Engineering or one type of problem. Thus, it is essential to expand your study in all directions, allowing great contributions to society, a fundamental role of the scientific community.

REFERÊNCIAS

ALI, D., DEVIKA, K. and KALIYAN, M. 'Davor Svetinovic. An optimization model for product returns using genetic algorithms and artificial immune system', **Resources, Conservation and Recycling**, Vol. 74, No. 5, pp.156–169, 2013.

ARDJMAND, E., YOUSSEF, E. M., MOYER, A., YOUNG II, W. A., WECKMAN, G. R., SHAKERI, H. 'A multi-objective model for minimising makespan and total travel time in put wall-based picking systems', **International Journal of Logistics Systems and Management**, Vol. 36, No. 1, pp.138-176, 2020.

AZAD, T. and HASIN, M. A. A. 'Capacitated vehicle routing problem using genetic algorithm: a case of cement distribution', **International Journal Logistics Systems and Management**, Vol. 32, No. 1, pp.132–146, 2019.

BEASLEY, J. E. 'A genetic algorithm for the set covering problem', **European Journal of Operational Research**}, Vol. 94, pp.392-404, 1996.

BURIOL, L. S. **Algoritmo memético para o problema do caixeiro viajante assimétrico como parte de um framework para algoritmos evolutivos**, Master's dissertation, Campinas State University - UNICAMP, Campinas, São Paulo, Brazil, 2000.

CROWDER, H. P., JOHNSON, E. L. and PADBERG, M. W. 'Solving Large-Scale Zero-One Linear Programming Problems', **Operations Research**, Vol. 31, pp.803–834, 1983.

FOGEL, D. B. and FOGEL, L. J. 'An introduction to evolutionary programming', in: Alliot J. M., Lutton E., Ronald E., Schoenauer M. and Snyers, D. (Eds): **Artificial Evolution, Lecture Notes in Computer Scienc**e, Vol. 1063, Springer, Berlin, Heidelberg, 1996.

GRANATYR, J. 'Modelo Afetivo de Reputação utilizando Personalidade e Emoção'}, 1st. ed., Beau Bassin: **New Academic Editions**, Vol. 1. 157p, 2018.

HAUPT, R. L. and HAUPT, S. E. Practical Genetic Algorithms, **John Wiley and Sons**, Inc., Hoboken, New Jersey, USA, 1998.

GOTTLIEB, J. and RAIDL, G. R. 'Characterizing locality in decoder-based EAs for the multidimensional knapsack problem', **In Proc. of Artificial Evolution**, pp.38–52, 1999.

LINDEN, R. **'Algoritmos Genéticos'**, 3rd. ed., Editora Ciência Moderna Ltda, Rio de Janeiro, Brazil, 496p, 2012.

LODI, A. and MONACI, M. 'Integer linear programming models for 2-staged two-dimensional knapsack problems', **Mathematical Programming**, Vol. 94, Nos. 2/3, pp.257–278, 2003.

MARQUES, F. P. '**O problema da mochila compartimentada e aplicacoes**', PhD, University of São Paulo - USP, Institute of Science and Mathematics and Computing - ICMC-USP, São Carlos, São Paulo, Brazil, 2004.

MARTELLO, S., TOTH and P. 'Knapsack Problems: Algorithms and Computer Implementations', **John Wiley and Sons**}, Chichester, England, 1990.

MATEO, P. M. and ALBERTO, I. 'Graph-based solution batch management for multi-objective evolutionary algorithms', **Applied Soft Computing**, Vol. 62, pp. 619–635, 2018.

MILLIKEN C. P. 'Object-Oriented Programming. In: Python Projects for Beginners',**Apress**, Berkeley, California, USA, 2020.

MUSTAFI, D. and SAHOO, G. 'A hybrid approach using genetic algorithm and the differential evolution heuristic for enhanced initialization of the k-means algorithm with applications in text clustering', **Soft Computing** vol. 23, no. 15, pp. 6361-6378, 2019.

NAZIF, H. and LEE, L. S. 'Optimized', **Applied Mathematical Modeling**, Vol. 36, No. 5, pp.2110–2117, 2012.

PINO, R., FERNÁNDEZ, I., FUENTE, D. and GARCÍA, N. 'A genetic algorithm approach to a 3D container-loading problem', **International Journal of Logistics Systems and Management**, Vol. 10, No. 2, pp.192–207, 2011.

POLI, R., LANGDON, W. B. and MCPHEE, N. F. '**A Field Guide to Genetic Programming**', 1st ed. [S.l.]: Lulu Enterprises, 252p, 2008.

PRINS, C. 'A simple and effective evolutionary algorithm for the vehicle routing problem', **Computers & Operations Research**, Vol. 31, No. 12, pp.1985–2002, 2004.

RAJKANTH, R., SRINIVASAN, G. and GOPALAKRISHNAN, M. 'Material flow optimisation in a multi-echelon and multi-product supply chain', **International Journal of Logistics Systems and Management**, Vol. 26, No. 1, pp.105–124, 2017.

SHAYANFAR, E. and SCHONFELD, P. 'Selecting and scheduling interrelated road projects with uncertain demand', **Transportmetrica A: Transport Science**, Vol. 15, No. 2, pp.1712–1733, 2019.

SHEPPARD, C. '**Genetic Algorithms with Python**' 1st ed. [S.l.]: CreateSpace Independent Publishing Platform, 532p, 2016.

SHI, X., WEI L., YANYAN, L., DINGSHAN, D. and YONGLAI, W. 'Research on the performance of multi-population genetic algorithms with different complex network structures', **Soft Computing: Methodologies and Application**, Springer, 2020.

SHI, X., WEI, L., LI, Y. Y., WEI, Y. L. and DENG, D. S. 'Different performances of different intelligent algorithms for solving FJSP: a perspective of structure', **Computational Intelligence and Neuroscience**, Vol. 2018:4617816, 2018.

SUMETTHAPIWAT, S., INTIYOT, B. and JEENANUNTA, C. 'A column generation on two-dimensional cutting stock problem with fixed-size usable leftover and multiple stock sizes', **International Journal of Logistics Systems and Management**, Vol. 35, No. 2, pp.273–288, 2020.

TENG, J. and TZENG, G. 'A Multi-objective Programming Approach for Selecting Non-Independent Transportation Investment Alternatives', **Transportation Research Part B-Methodological**, Vol. 30, No. 4, pp.291–307, 1996.

TORRES-ESCOBAR, R., MARMOLEJO-SAUCEDO, J. A. and LITVINCHEV, I. 'Binary monkey algorithm for approximate packing non-congruent circles in a rectangular container', **Wireless Networks**, Springer, 2018.

WANG, S. and WANG, G. 'Reasoning Method of Situation Information System Based on Multi-agent Network', **4th International Conference on Information Systems Engineering**, ICISE 2019, pp.16-20, Shanghai, China, 2019.

YAZDI, A.K., WANG, Y.J. and KOMIJAN, A.R. 'Green supply chain management in an emerging economy: prioritising critical success factors using grey-permutation and genetic algorithm', **International Journal of Logistics Systems and Management**, Vol. 36, No. 2, pp.199–223, 2020.