ALGORITHMIC EFFICIENCY INDICATOR FOR THE OPTIMIZATION OF ROUTE SIZE

Ciro Rodríguez
National University Mayor de San Marcos. Lima, (Perú).
E-mail: crodriguezro@unmsm.edu.pe ORCID: https://orcid.org/0000-0003-2112-1349

Miguel Sifuentes
National University Mayor de San Marcos. Lima, (Perú).
E-mail: newangel2018@hotmail.com ORCID: https://orcid.org/0000-0002-0178-0059

Freddy Kaseng
National University Federico Villarreal. Lima, (Perú).
E-mail: fkaseng@unfv.edu.pe ORCID: https://orcid.org/0000-0002-2878-9053

Pedro Lezama
National University Federico Villarreal. Lima, (Perú).
E-mail: pedrolezamagonzales@gmail.com ORCID: https://orcid.org/0000-0001-9693-0138

Recepción: 06/02/2020 **Aceptación:** 23/03/2020 **Publicación:** 15/06/2020

Citación sugerida:
Rodríguez, C., Sifuentes, M., Kaseng, F., y Lezama, P. (2020). Algorithmic efficiency indicator for the optimization of route size. *3C Tecnología. Glosas de innovación aplicadas a la pyme, 9*(2), 49-69. http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

ABSTRACT

In software development, sometimes experienced programmers have difficulty determining before performing their tests, which algorithm will work best to solve a particular problem, and the answer to the question will always depend on the type of problem and nature of the data to be used, in this situation, it is necessary to identify which indicator is the most useful for a specific type of problem and especially in route optimization. Currently, there are techniques and algorithms used in Artificial Intelligence, which, however, cannot display their potential without a well-defined data set. The paper seeks to explain and propose an algorithm selection indicator to build a consistent data set, given the lack of availability of data that allows the best route size decision to be made.

KEYWORDS

Software development, Algorithm efficiency, Indicator, Route size.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

1. INTRODUCTION

The type of problem and nature of the data to be used is necessary to identify which indicator is the most useful for a specific kind of problem and especially in route optimization. As Microsoft (n.d.) explain are several ways to make the selection. In Machine Learning, cross-validation is one of the most commonly used. It was finding the best set of parameters like space, cross-validation, metric, accuracy. It is important to formulate, evaluate, and compare the values of the parameters, after the evaluation and comparison, it is possible to choose the best alternative as shown in Figure 1.

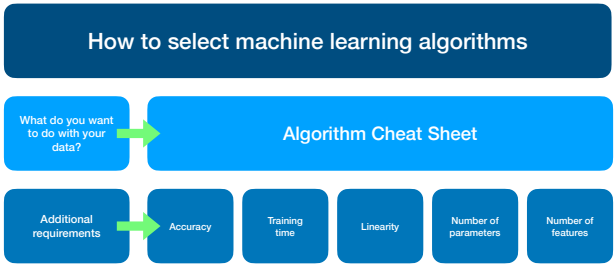

Figure 1. What are the requirements of your data science scenario? **Source:** (Microsoft, n.d.).

In the search for the most appropriate algorithm to implement in the route optimizer we will have three candidates for the best indicator: the algorithmic efficiency, the size (sum of weights of the edges) of the route obtained and the following relation, derived from the previous indicators. The witness search is expensive (Dibbelt, 2016), and therefore the witness search is usually aborted after a certain number of steps. If no witness was found, is assumed that none exists and add a shortcut, as shown in Figure 2.

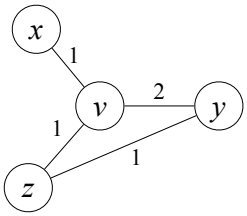

Figure 2. Contraction of v. **Source:** (Dibbelt, 2016).

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

Figure 2 sows pair (x, y); if is considered first, a shortcut {x, y} with weight 3 is inserted. If the pair (x, z) is considered first, an edge {x, z} with weight 2 is inserted. This shortcut is part of a witness $x \rightarrow z \rightarrow y$ for the pair x, y. The shortcut {x, y} is thus not added if the pair x, z is considered first.

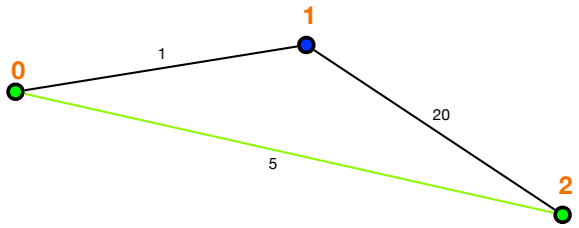

Figure 3. The minimum path from node 0 to node 2.

Figure 3 shows the result of using the Dijkstra algorithm in the graph, which presents the minimum sum of edges to get from node 0 to node 2, however, it is not the most efficient of the algorithms presented in this research paper.

2. CONCEPTUAL FRAMEWORK

2.1. ARTIFICIAL INTELLIGENCE AND DATASETS

Forecasts say that in the next decade, there will be approximately more than 150,000 million sensors connected to the network (more than 20 times the Earth's population). These data help AI devices to think as we think, accelerating their learning curve and automation of data analysis with all the processed information; considering as much data the system receives, more learning and accuracy becomes (PowerData, 2017).

Today, artificial intelligence can learn without human support. New cases are known daily, such as the example of a Google DeepMind algorithm, which recently learned on its own how to win 49 Atari games, without the need for any interaction from anyone.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

In the past, AI growth was minimal for two main reasons: limited data sets that used representative data samples, instead of using real-time data, and inability to analyze massive amounts of data in seconds (PowerData, 2017).

2.2. THE LACK OF DATASETS

The ability to manage large volumes and data sources is enabling the capabilities of AI (Artificial Intelligence) and machine learning . However, some organizations cannot yet exploit AI capabilities for various reasons:

- The lack of data availability.
- Limited sample sizes negatively affect their abilities.
- The lack of appropriate technologies to analyze massive data in milliseconds.

2.3. ALGORITHMICS

Knowing what problem is going to be solved is only half the job. In dealing with problems, they generally do not have a precise and straightforward specification of them. Problems such as creating a gourmet-worthy recipe or preserving world peace may be impossible to formulate in a way that supports a computer solution; Although it is believed that the problem can be solved on a computer, it is usual that the distance between several of its parameters is at least considerable (Aho, Hopcroft, & Jefrey, 1988).

Often only through experimentation, is it possible to find reasonable values for these parameters. If it is possible to express certain aspects of a problem with a formal model, it is generally beneficial to do so, because once the problem is formalized, solutions can be sought based on a precise model and determine if there is already a program that solves that problem. The algorithm has a remarkable importance in the development of software when considering various algorithms for a given problem.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

The algorithm is defined as the study of algorithms, which are a sequence of ordered and finite steps, each of which has a precise meaning and can be executed with a limited amount of effort in a finite time to solve a problem. By steps means the set of actions or operations that are performed on certain objects. For the solution of a problem, a collection of algorithms is taken into account, depending on the particular characteristics of the problem (selected algorithm) ("Complejidad algorítmica", n.d.).

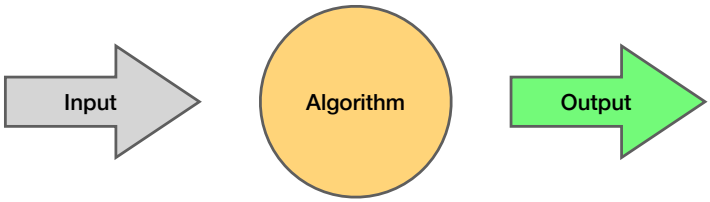

Figure 4. Determinism as a feature of the algorithms.

An algorithm as Figure 4 must have the following characteristics:

- Accuracy: An algorithm must be expressed without ambiguity.
- Determinism: Every algorithm must respond in the same way before the same conditions
- Finite: The description of an algorithm must be limited.

An algorithm must also achieve the following objectives, which are often contradicted:

- Simple: Make it easy to understand, code, and debug.
- Efficient: Efficient use of computer resources in time, space (memory) and processor, so that it runs as quickly as possible.

It is usually difficult to find an algorithm that meets both, so a compromise must be reached that best fits the requirements of the problem ("Complejidad algorítmica", n.d.).

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

The algorithm allows evaluating the results of different external factors on the available algorithms in such a way that it will enable to select the one that best suits the particular conditions. It allows indicating how to design a new algorithm for a specific task.

It has been said that each instruction in an algorithm must have a "precise meaning" and must be executable with a "finite amount of effort"; But what is clear to one person may not be clear to another, and it is often challenging to demonstrate rigorously that an instruction can be done in a finite time. Sometimes, it is difficult to show the sequence of instructions ends with some input, even if the meaning of each instruction is very clear. However, an agreement is usually reached as to whether or not a sequence of instructions constitutes an algorithm (Aho *et al.*, 1988).

2.4. ALGORITHMIC COMPLEXITY

Algorithmic complexity represents the number of resources (temporary) that an algorithm needs to solve a problem and therefore allows to determine the efficiency of the said algorithm (Universidad de Valladolid Campus de Segovia). It is not referred to the difficulty in designing algorithms. The criteria that will be used to assess algorithmic complexity do not provide absolute measures but measures relative to the size of the problem. What is relevant, at first, is the measure of temporal complexity in terms of the input data.

When a problem is resolved, there is often a need to choose between several algorithms. How should you choose? Two objectives are often contradicted:

1. That the algorithm is easy to understand, code, and debug.
2. That the algorithm efficiently uses the computer resources and, in particular, that it be executed as quickly as possible ("Complejidad algorítmica", n.d.).

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

When writing a program to be used once or a few times, the first objective is the most important. In this case, it is very likely that the cost of programming time dramatically exceeds the value of program execution, so that the price to be optimized is that of writing the program (Aho *et al.*, 1988).

On the other hand, when there is a problem whose solution is going to be used many times, the cost of executing the program can significantly exceed that of writing, especially if large entries are given in most executions. Then, it is more advantageous, from the economic point of view, to perform a complex algorithm provided that the execution time of the resulting program is significantly shorter than that of a more obvious program. And even in situations like that, it may be convenient first to implement a simple algorithm to determine the real benefit that would be obtained by writing a more complicated program. In the construction of a complex system, it is often desirable to implement a simple prototype in which simulations and measurements can be made before engaging in the final design. This concludes that a programmer must not only be aware of ways to get a program to run quickly but also know when to apply these techniques and when to ignore them (Aho *et al.*, 1988).

An algorithm will be more efficient compared to another, provided it consumes fewer resources, such as the time and memory space needed to execute it.

The evaluation of the algorithms for efficiency will be taken into account:

- The growth rate in time. Runtime: It has to do with the time it takes for a program to run (processing time).
- The growth rate in space.

Memory space: Study the amount of space that is necessary for operations during program execution (storage and processing space).

The efficiency of an algorithm can be quantified with the following complexity measures:

- Temporary Complexity or Execution Time: Computation time necessary to execute a program.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

- Spatial Complexity: Memory that an application uses for its execution. The memory efficiency of an algorithm indicates the amount of space required to execute the algorithm; that is to say, the area in memory that occupies all the own variables to the algorithm. To calculate the static memory of an algorithm, the memory occupied by the variables declared in the algorithm is added. In the case of dynamic memory, the calculation is not so simple since this depends on each execution of the algorithm ("Complejidad algorítmica", n.d.).

2.5. GRAPHS

A graph G is a set of points in space, some of which are linked by lines (Menendez, 1998), formally, a graph G consists of two finite sets N, and A. N is the set of elements of the graph, also called vertices or nodes. A is the set of arcs, which are the connections that are responsible for relating the nodes to form the graph. Arcs are also called edges or lines. Nodes are often used to represent objects and arcs to represent the relationship between them. For example, nodes can represent cities and arches the existence of roads that communicate them. Each arc is defined by a pair of elements n1, n2 ∈ N to which it connects. Although the parts are usually different, we will allow them to be the same node (n1 = n2).

Trees have been considered as a generalization of the list concept because they allow an element to have more than one successor. Graphs appear as an extension of the tree concept since, in this new type of structure, each element can have, in addition to more than one successor, several predecessor elements. This property makes graphs the most appropriate structures to represent situations where the relationship between the elements is completely arbitrary, such as road maps, telecommunications systems, printed circuits, or computer networks. Although there are more complex structures than graphs, we will not see them in this course. Graphs can be classified into different types depending on how the relationship between the elements is defined: we can find directed or non-directed and labeled or unlabeled graphs. We will use the tags when we work with the Dijkstra and Prim algorithms.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

The graph is not directed if the arcs are formed by pairs of unordered nodes, not pointed; A graph is directed, also called a digraph, if the pairs of nodes that form the arcs are ordered; they are represented with an arrow indicating the direction of the relation u → v, these being a pair of nodes (Joyanes & Zahonero, 2008).

In some instances it is necessary to associate information to the arcs of the graph. This can be achieved through a label containing any useful information related to the arc, such as the name, weight, cost or a value of any given data type. In this case we talk about labeled graphs. This label could mean the time it takes for the flight between two cities or indicates what the input and output parameters are in the call to a subprogram. An unlabeled graph is a graph where the arcs have no labels. In the case of the graph that represents the traffic direction, the arcs can be labeled.

3. METHODOLOGY

3.1. SHORTEST ROUTE

Researchers consider the Shortest Route Problem as a central problem within the area of networks due to the variety of practical applications, the existence of efficient solution methods, and the use of subroutines in the search for the right solution in complex problems. However, in the subjects in which these kinds of problems are addressed, they are usually presented in a simplified and unclear way, that is, the importance of studying these types of problems is often not recognized or stressed. Theoretical aspect and due to the great diversity of applications (Obregon, 2005), so this problem can be solved by, among others, the Dijkstra and Prim algorithms.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

3.2. MINIMUM PATH ALGORITHMS

A. Naive Algorithm

The naive algorithm for directed graphs has reasonably good efficiency; however, it cannot see beyond the node following the current one of the network.

It is an algorithm that starts from a source node and goes exploring the paths to the rest of the nodes. The following pseudocode shows its operation, using the priority queue as an auxiliary data structure (Jungnickel, 1999).

```
Naive (graph G, sourceN s)
  for u ∈ V[G] do
    distance[u] = INFINITY
    parentNode[u] = NULL
    seen[u] = false
  distance[s] = 0
  insert (queue, (s, distancie[s]))
  while ! isEmpty (queue) do
    u = extractMínimum (queue)
    seen[u] = true
    insert (queue, (v, distance[v]))
```

B. Dijkstra's algorithm

The Dijkstra algorithm, also called the minimum path algorithm, is a model that is classified within the search algorithms. Its objective is to determine the shortest route, from the origin node to any node in the network. Its methodology is based on iterations, so that, in practice, its development becomes difficult as the size of the network increases, leaving it at a clear disadvantage, compared to optimization methods based on mathematical programming (Jungnickel, 1999).

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

```
Dijkstra (graph G, sourceNode s)
  for u ∈ V[G] do
    distance[u] = INFINITY
    parent_node[u] = NULL
    seen[u] = false
  distance[s] = 0
  insert (queue, (s, distance[s]))
  while ! istaEmpty (queue) do
    u = extract_mínimum(queue)
    sen[u] = true
    for each v ∈ adyjancency[u] do
        if ¡ seen[v]and distance[v] > distance[u] + length (u, v) do
        distance[v] = distance[u] + length (u, v)
        parent_node[v] = u
        insert (queue, (v, distance[v]))
```

C. Prim Algorithm

Prim's algorithm, given a related graph, not directed and weighted, finds a minimal expansion tree. That is, it can find a subset of the edges that form a tree that includes all the vertices of the initial graph, where the total weight of the edges of the tree is the minimum possible (Jungnickel, 1999).

Given a set of nodes N, a set of edges E, and a cost function p, our network (graph) G is defined. To apply the Prim algorithm and solve the problem of the minimum expansion tree, all costs associated with the edges mustn't be negative. It is an algorithm that continuously increases the size of a tree, starting with an initial node, chosen at random, to which nodes are added whose distance successively to the previous ones is minimal. In each step, we will consider the edges that contain nodes that already belong to the tree. The minimum cost expansion tree is entirely constructed when there are no more nodes left to add (Jungnickel, 1999).

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

```
Prim (graph G (N, E, p))
  for every u in N do
    distance[u] = INFINITY
    parentNode[u] = NULL
    Insert (queue, < u, distance[u] >)
  distance[u] = 0
  while ! isEmpty (queue) do
    u = extractMínimum (queue)
    for each v adjacent to 'u' do
    if ((v ∈ queue) && (distance[v] > p (u, v)) then
      parentNode[v] = u
      distance[v] = p(u, v)
                              Update (queue, < v, distance[v] >) [10]
```

3.3. ROUTE SIZE INDICATOR

The size of the route is defined as the sum of the weights of all the edges of the graph that the algorithm has traveled.

For the graph in above Figure 2:

A. Routes that each algorithm has followed

- Naive Algorithm: $0 \rightarrow 1 \rightarrow 2$
- Dijkstra algorithm: $0 \rightarrow 2$
- Prim Algorithm: $0 \rightarrow 2$

B. Route size obtained with each algorithm

- Naive Algorithm: $1 + 20 = 21$
- Dijkstra Algorithm: 5
- Prim Algorithm: 5

Both the Dijkstra and Prim algorithms have obtained an optimal route size, with a value of 5.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

3.4. ALGORITHMIC EFFICIENCY COMPLEXITY INDICATOR

Big O notation is used to asymptotically limit the growth of a runtime that is within constant factors above and below. Sometimes we want to limit only above. It is convenient to have a form of asymptotic notation that means "the execution time grows at most by this much, but it can grow more slowly." We use the "big O" notation just for these occasions. If execution time is O (f (n)), then for a sufficiently large n, the execution time is at most k * f (n) for some constant k (Magzhan & Jani, 2013).

A. The efficiency of the naive algorithm (approximate)

   $0(log|V|)=(|log|V|)$

B. Dijkstra algorithm efficiency

   $0((|A|+|V|)log|V|)=0(|A|log|V|)$

C. Prim algorithm efficiency

   $0(n^2)$

To observe results quantitatively, the table 1 will show the data obtained from a previous study (Magzhan & Jani, 2013), regarding Dijkstra's algorithm and others; subsequently, the values will be extrapolated to the cases of the naive algorithm, of complexity O (| log | V |), and Prim, of complexity O (n^2), using the theoretical complexity of each of these algorithms.

Table 1. Results of performance time of the algorithms for different graphs (ms).

Nodes	Edges	Maximum edge cost	Dijkstra with d-heap O ($Alog_dN$) (ms)	Prim O (n^2) (ms)	Naive O (|log|V|) (ms)
100	100	10	0.14	0.12	0.0014
500	100	10	3.77	3.23	0.0377
1000	100	10	11.02	10.34	0.0110
3000	100	10	100.00	90.00	0.0100

Note: It is adapted from Magzhan and Jani (2013), the development of a library for minimum roads, applied to a data protection problem.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

Because the Naive algorithm does not consider the comparison to obtain the shortest distance with the adjacent nodes, unlike the others, it is the most efficient with a relative efficiency of O (| log | V |).

Initially, it was enough to analyze one of the indicators to choose the algorithm indicated for the minimum route problem, then there was an alleged contradiction between the results of algorithmic efficiency and route size, however, through the problem of backpack it was discovered that a relationship between these indicators could be established and we can also accommodate it to different scenarios through the parameters p and q.

4. INDICATOR OF ALGORITHMIC EFFICIENCY RATIO IN ROUTE SIZE OPTIMIZATION

When solving the problem of the selection of route optimizing algorithms, if the algorithmic efficiency indicator is the guide, we will obtain that the recommended algorithm is naive, but this algorithm is, in turn, one of the longest routes. Despite being the most efficient, it is not the right algorithm to solve these types of problems. And by following the criterion of the size of the route, we are not certain that we have the optimal algorithmic efficiency as Formula (1).

$$\frac{1}{(algorithmic\ efficiency)^p * (route\ size)^q} \qquad (1)$$

In the Knapsack problem, we saw how the apparent contradiction of indicators was solved; similarly, for the problem of selecting algorithms in route optimizers, it is proposed to establish a relationship between algorithmic efficiency and route size, thus generating a new criterion.

A third indicator is proposed through the relationship of the previous two (1), we have both indicators in the denominator because it seeks to minimize the size of the route and seeks to minimize complexity thus optimizing algorithmic efficiency. The parameters p and q serve us to graduate the level of importance that we will give to each indicator, for example, for a brute force algorithm it is advisable to assign at least

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

a value of 2 to the parameter p and for our route optimizer the value of the parameter q, not It should be less than 3 due to the great importance of finding the minimum route in emergencies.

What this new indicator expresses to us is the relationship between efficiency and size of the route; however, it forces to decide to assign a level of importance to the two previous indicators.

Evaluating which algorithm to choose for the graph with the new indicator:

A. With p = 1 y q = 2, is possible get:

- Naive Algorithm: $1/(0(\log|3|) * 21^{2})=0.00475261513$
- Dijkstra Algorithm: $1/(0(|3|\log|3|) *5^{2})=0.02794537699$
- Prim Algorithm: $1/(0(3^{2}) * 5^{2})=0.00444444444$

The one that gets the highest score in the indicator is the Dijkstra algorithm, which is why it should be implemented in this specific graph and with these parameters.

B. With p = 5 y q = 1, is possible get:

- Naive Algorithm: $1/(0((\log|3|)^{5}) * 21)=1.92591398109$
- Dijkstra Algorithm: $1/(0/(|3|\log|3|)^{5} * 5)=0.03328740214341$
- Prim Algorithm: $1/(0((3^{2})^{5}) * 5)=0.00000338701$

The one that obtains the highest score in the indicator is the naive algorithm, due to the enormous priority given to efficiency (p = 5), recommended in embedded equipment with minimal processing capacity, it should be implemented in this specific graph and with these parameters.

C. With p = 1 y q = 4 (recommended for route optimizer in ambulances because $q \geq 3$), we get:

- Naive Algorithm: $1/(0(\log|3|) * 21^{4})=0.0000107769$
- Dijkstra Algorithm: $1/(0(|3|\log|3|)* 5^{4})=0.00111781507$

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

- Prim Algorithm: $1/(0(3^2) * 5^4)=0.00017777777$

The one that gets the highest score in the indicator is the Dijkstra algorithm, followed by the Prim algorithm, so it should be implemented in this specific graph and with these parameters. These values of parameters p and q are recommended to optimize ambulance routes in an emergency.

It is essential to understand the power of the p and q parameters in the results since they affect the algorithmic efficiency and the size of the route in an exponential way.

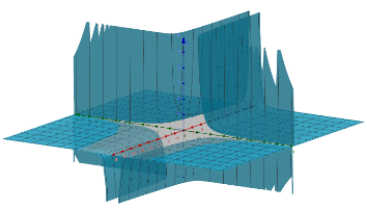

Figure 5. p =1; q = 2.

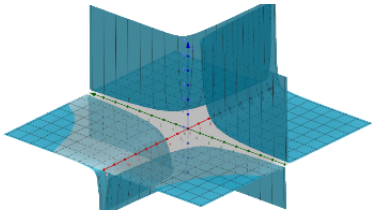

Figure 6. p =5 ; q = 1.

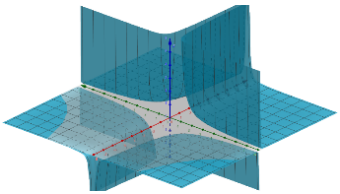

Figure 7. p =1 ; q = 4.

Figures 5, 6, and 7 show the behavior of the indicator with the different parameters assumed by p and q the X-axis, where the red axis represents the algorithmic efficiency and the green Y-axis the size of the route.

5. STORAGE OF RESULTS IN A DATASET

It is important to store the p and q values, preferably in tuples, for later analysis and to relates them to the results of the new indicator obtained and the order of the algorithms obtained to form a consistent dataset and subsequently worked with AI. With the use of the data sets built, various problems can be solved, such as the minimum path and data science through machine learning.

Machine learning is a technique to implement artificial intelligence (Mediaroom Solutions, 2019; Bustamante, Rodriguez, & Esenarro, 2019), without programming millions of rules and decision trees. Its objective is to allow machines to learn automatically and increasingly autonomously. It consists of

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

learning algorithms that can be of 4 types: supervised, unsupervised, semi-supervised, and reinforced. A classic example is music recommendation systems based on preferences and songs that the user likes.

6. DISCUSSION ABOUT INDICATORS AND THE KNAPSACK PROBLEM

Apparent there are a contradiction of the indicators algorithmic efficiency and route size, differents criteria recommend different algorithms, which seems to show an apparent contradiction, a similar situation is presented in the Fractional Knapsack Problem where:

Given n elements e_1, e_2, ..., in with weights p_1, p_2, ..., p_n and benefits b_1, b_2, ..., b_n, , and given a Knapsack capable of holding elements up to a maximum of weight M ((Knapsack capacity), that is, we want to find the proportions of the n elements x_1, x_2, ..., $x_n (0 \leq x_i \leq 1)$ that we have to introduce in the Knapsack so that the sum of the benefits of the chosen elements be maximum. That is, we must find values (x_1, x_2, ..., x_n) so that the benefit is maximized, taking into account that the sum of weights cannot exceed M (the maximum capacity of the Knapsack) (Joyanes & Zahonero, 2008).

At the moment we have two options to achieve our goal, the first one would be to order our descending elements according to the benefit and select them when doing this we would be facing the problem in which we have an item with a great advantage, but with enormous weight.

As a second solution proposal, we have the order of ascending the elements and selecting them; however, the criterion does not work properly when finding an element with low weight, but with a small benefit.

Failing the criteria of maximum benefit and minimum weight gives the appearance that both requirements are in contradiction; however, this is not so because there is the alternative of expressing the third indicator through the relationship of the previous two: b_i/p_i. Here in the numerator, the criteria are to maximize and the denominator to minimize.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

A voracious algorithm that solves the problem orders the elements in a decreasing way concerning their ratio b_i/p_i and adds objects while they fit (Joyanes & Zahonero, 2008). The precondition for this algorithm is the vectors of weights and benefits are sorted in descending order according to the b_i/p_i ratio.

```
Knapsack (real benef, real weight, real cap, real sol)
  For i from 0 to n − 1 do
    sol[i] ← 0.0
  rest ← cap
  i ← 0
  While (i ≤ n − 1) and (weight[i] ≤ resto) do
    sol[i] ← 1
    resto ← resto − weight[i]
    i ← i + 1
  If i < n then
    sol[i] ← resto/weight[i] [12]
```

As we can see, the weight and benefits indicators were not confronted; it was only necessary to express the relationship between them and the connections used as a new indicator.

7. CONCLUSIONS

Initially, it was enough to analyze one of the indicators to choose the algorithm indicated for the minimum route problem, then there was an alleged contradiction between the results of algorithmic efficiency and route size, however, through the of Knapsack problem it was discovered that a relationship between these indicators could be established and we can also accommodate it to different scenarios through the parameters p and q.

In the search to contribute with the fruits of this study to the solution of various graph problems later with the results of the algorithms selected by the indicator (1), we can form consistent data sets that nourish current and future AI techniques.

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69

ACKNOWLEDGMENTS

This paper would not have been possible to realize without the assistance, support, and patience of the research group. We would like to thank Dra. Magnolia Rueda for her invaluable advice and unsurpassed knowledge.

REFERENCES

Aho, A., Hopcroft, J., & Jefrey, U. (1988). Estructura de datos y algoritmos. En G. Levine Gutiérrez (Ed.). Addison-Wesley Iberoamericana: Sistemas Técnicos de Edición. https://www.academia.edu/23710587/Estructura_de_Datos_y_Algoritmos_-_Aho_Hopcroft_Ullman

Bustamante, J. C., Rodriguez, C., & Esenarro, D. (2019). Real Time Facial Expression Recognition System Based on Deep Learning. *International Journal of Recent Technology and Engineering (IJRTE), 8*(2S11), 4047-4051. https://doi.org/10.35940/ijrte.B1591.0982S1119

Dibbelt, J. M. (2016). *Engineering Algorithms for Route Planning in Multimodal Transportation Networks.* Karlsruher Instituts für Technologie (KIT). https://publikationen.bibliothek.kit.edu/1000053050/3808147

Google Site. (n.d.). *Complejidad Algoritmica.* (Algoritmo de Prim). https://sites.google.com/site/complejidadalgoritmicaes/prim

Joyanes, L., & Zahonero, I. (2008). Estructura de Datos en Java. In L. Joyanes, I. Zahonero, & J. L. García (Ed.), *Estructura de Datos en Java* (1ª ed., 161-189). MCGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.

Jungnickel, D. (1999). Graphs networks and algorithms. In D. Jungnicquel, *Algorithms and Computation in Mathematics* (Volume 5, 35-56). Springer. https://doi.org/10.1007/978-3-662-03822-2

Magzhan, K., & Jani, H. M. (2013). A Review And Evaluations Of Shortest Path Algorithms. *International Journal of Scientific & Technology Research, 2*(6), 99-104. http://www.ijstr.org/final-print/june2013/A-Review-And-Evaluations-Of-Shortest-Path-Algorithms.pdf

Mediaroom Solutions. (n.d.). *Tendencias de inteligencia artificial y sus ventajas para la empresa.* https://www.mediaroomsolutions.es/blog/tendencias-en-inteligencia-artificial/

Menendez, A. (1998). Una breve introducción a la teoría de grafos. *Suma*, (28), 11-26. https://revistasuma.es/IMG/pdf/28/011-026.pdf

Microsoft. (n.d.). *How to select algorithms for Azure Machine Learning.* https://docs.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms

Obregon, B. (2005). *Teoría de redes. El problema de la ruta más corta*. Universidad Nacional Autónoma de México. http://webcache.googleusercontent.com/search?q=cache:I5G9zPMqws4J:www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/539/obregonquintana.pdf%3Fsequence%3D12+&cd=1&hl=es&ct=clnk&gl=pe

PowerData. (2017). *El valor de la gestión de datos: Inteligencia Artificial, Machine Learning y Big Data*. https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/ia-inteligencia-artificial-y-machine-learning-se-vinculan-al-big-data

Complejidad algorítmica. (n.d.). Departamento de Informática. Universidad de Valladolid. Campus de Segovia. https://www2.infor.uva.es/~jvalvarez/docencia/tema5.pdf

http://doi.org/10.17993/3ctecno/2020.v9n2e34.49-69