Complexity of Instances for Combinatorial Optimization Problems

Jorge A. Ruiz-Vanoye[1], Ocotlán Díaz-Parra[1],
Joaquín Pérez-Ortega[2], Rodolfo A. Pazos R.[3]
Gerardo Reyes Salgado[2] and Juan Javier González-Barbosa[3]
[1] *Universidad Autónoma del Estado de Morelos, Mexico*
[2] *Centro Nacional de Investigación y Desarrollo Tecnológico, Mexico*
[3] *Instituto Tecnológico de Ciudad Madero, Mexico*

1. Introduction

The theory of the computational complexity is the part of the theory of the computation that studies the resources required during the calculation to solve a problem (Cook, 1983). The resources commonly studied are the time (execution number of an algorithm to solve a problem) and the space (amount of resources to solve a problem). In this area exist problems classifications that are approached within the theory of the complexity, some definitions of the complexity classes are related to this investigation:

a. P class.-It is the class of recognizable languages by a determinist Turing Machine of one tape in polynomial time (Karp, 1972).
b. NP class.-It is the class of recognizable languages by a Non-determinist Turing Machine of one tape in polynomial time (Karp, 1972).
c. NP-equivalent class.-It is the class of problems that are considered NP-easy and NP-hard (Jonsson & Bäckström, 1995).
d. NP-easy class.-It is the class of problems that are recognizable in polynomial time by a Turing Machine with one Oracle (subroutine). In other words a problem *X* is NP-easy if and only if a *Y* problem exists in NP like *X* is reducible Turing in polynomial (Jonsson & Bäckström, 1995).
e. NP-hard class.-A Q problem is NP-hard if each problem in NP is reducible to Q (Garey & Johnson, 1979; Papadimitriou & Steiglitz, 1982). It is the class of problems classified as problems of combinatorial optimization at least as complex as NP.
f. NP-complete class.-A *L* language is NP-complete if *L* is in NP, and Satisfiability $\leq_p$ *L* (Cormen et al., 2001; Karp, 1972; Cook, 1971). It is the class of problems classified like decision problems.

A combinatorial optimization problem is either a minimization problem or a maximization problem and consists of three parts: a) a set of instances, b) candidate solutions for each instance, c) a solution value (Garey & Johnson, 1979). The combinatorial optimization problems that was used in this paper: the General Asymmetric Traveling Salesman Problem

(ATSP), JobShop Scheduling Problem (JSSP) and Vehicle Routing Problem with Time Windows (VRPTW).
The general Asymmetric Traveling Salesman Problem (ATSP), which can be stated as follows: given a set of nodes and distances for each pair of nodes, find a route of minimal overall length that visits each of the nodes exactly once (Cormen et al., 2001). The distance of node *i* to node *j* and the distance of node *j* to node *i* can be different (equations 1, 2, 3, 4).

$$\min z(x) = \sum_{j=1}^{m}\sum_{i=1}^{m} d_{ij}x_{ij} \tag{1}$$

$$\sum_{j=1}^{m} x_{ij} = 1; \quad i = 1,...,m; \quad d_{ij} \neq d_{ji} \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1; \quad j = 1,...,m\,; \quad 0 \le x_{ij} \le 1 \tag{3}$$

$$x_{ij} = \begin{cases} 0, \text{if tour traverses from } i \text{ to } j \\ 1, \text{otherwise} \end{cases} \quad \forall i,j \tag{4}$$

The JobShop Scheduling Problem (JSSP) contains a number of machines and a set of Jobs each one with precedence restrictions, the problem is to solve the question if exist a scheduling of jobs that help to improve and to efficiency the use of the machines being eliminated the idle times. It is recognized by that it does not have to be able human nor machine sufficiently fast that it can obtain the optimal solution for JSSP due to the solutions space, which cannot be expressed by a polynomial function (deterministic algorithm), the space of solutions for this kind of problem can be only expressed like an exponential function. For the problem of JSSP is necessary to diminish makespan (*cmax*), this can be formulated as it follows (equations 5, 6, 7, 8):

$$\min\ c_{\max} \tag{5}$$

$$c_{jK} - t_{jk} \ge c_{jh},\ j = 1,2,...,n \ \ h,k = 1,2,...,m \tag{6}$$

$$c_{jK} - c_{ik} + M(1 - x_{ijk}) \ge t_{jk},\ \ i,j = 1,2,...n \quad k = 1,2,...,m \tag{7}$$

$$c_{iK}, c_{jK} \ge 0,\ x_{ijk} = 1 \text{ or } 0,\ i = 1,2,...,n \quad k=1,2,...,m \tag{8}$$

The Vehicle Routing Problem with Time Windows (VRPTW) is a combinatorial optimization problem complex (Toth & Vigo, 2001; Ruiz-Vanoye et al., 2008b; Cruz-Chávez et al., 2008). The VRPTW (Toth & Vigo, 2001) consists basically of to minimize the costs of subject transportation to time restrictions of each route and capacity on the cradle of the demand of each client (equations 9 and 10).

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{9}$$

$$a_i \sum_{j \in \Delta^+(i)} x_{ijk} \le w_{ik} \le b_i \sum_{j \in \Delta^+(i)} x_{ijk}, \forall k \in K, i \in N \tag{10}$$

In this paper, we propose specify the complexity of instances for problems of combinatorial optimization (ATSP, VRPTW, and JSSP).

2. Related works

An important measurement of complexity that we can attribute to Shannon (1949), is the complexity of Boolean circuits. For this measurement he is advisable to assume that *f* function at issue transforms finite strings of bits into finite strings of bits, and the complexity of *f* is the size of the smaller Boolean circuit than it calculates *f* for all the inputs of *n* length. But, it does not exist a classification of the complexity of instances for combinatorial optimization problems.

In some combinatorial optimization problems exists diverse types of instances size at the moment, for example:

REINELT (Reinelt, 1991) mentions that in General Asymmetric Traveling Salesman Problem (ATSP) instances exist whose size based on the number of cities or *nc*. See Table 1.

YAMADA (Yamada & Nakano, 1997) mentions that for the problem JobShop Scheduling Problem (JSSP) the size of the problem is the number of jobs or *J* and the number of machines or *M*. See Table 2.

SOLOMON (Solomon, 1987; Toth & Vigo, 2001) mentions that in the problem Vehicle Routing Problem with Time Windows (VRPTW) exists classifications of type C for instances clustered, type RC for instances Random and Clustered, Type R for Random instances and in addition the instance is determining by the number of clients or *CN*. See Table 3.

Instances	*nc*
br17	17
ft53	53
ft70	70
ftv33	33
ftv35	35
ftv38	38
ftv55	55
ftv64	64
ftv70	70
ftv170	170
rbg443	443
ry48p	48

Table 1. Number of cities for ATSP instances

Instances	*J*M*
abz5	10x10
abz6	10x10
abz8	20x15
ft06	6x6
ft10	10x10
ft20	20x5
orb01	10x10
orb02	10x10
yn1	20x20
yn2	20x20
yn3	20x20
yn4	20x20

Table 2. Number of jobs and number of machines for JSSP instances

Instances	*CN*
c101	25
c102	25
c205	25
r101	50
r102	50
r201	100
r205	100
rc101	200
rc102	200
rc201	500
rc202	500
rc203	500
rc204	500
rc205	500

Table 3. Client number for VRPTW instances

3. Complexity of Instances for Combinatorial Optimization Problems

In this section, we propose a basic methodology (Fig. 1) for create the metric that permit to classify or to determine the complexity instances of combinatorial optimization problems.
Step 1. To identify the maximum instance solved of the problem.
Step 2. To identify the instances parameters of the problem.
Step 3. Elaboration of the general metric of instances complexity taking into account measured of descriptive statistics and the parameters of the problems (equation 11).

$$InstancesComplexity = (PS + AM + SD + GD + RA)/100 \quad (11)$$

Where: *PS* is the instance average size at the rate of the instance maximum solved of the problem, *AM* is the sum of the arithmetic mean of each instance parameters (that have several elements), *SD* is the sum of the standard deviations of each type of parameter (that have several elements), *GD* is the sum of the genetic distances which quantifies the similarity or differentiates between the populations from the frequencies of the elements from each parameters (that have several elements) of the problem, *RA* is the sum of the existing reasons between the greatest value and the value smaller of each parameter (that has several elements) of the problem.

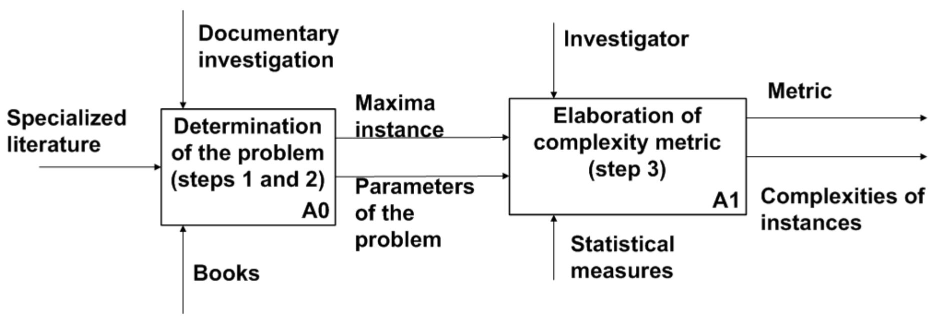


Fig. 1. Methodology of complexity instances for computational combinatorial problems

4. Experimentation

The experimentation was performed on a computer with Celeron processor 1.5GHz, 512 MB of memory, 80 GB of hard disk.
We use: the ATSP instances obtained of TSPLIB (Reinelt, 1991) benchmark, the JSSP instances of JSSPLIB (Yamada & Nakano, 1997) benchmark, the VRPTW instances of Solomon (Solomon, 1987) benchmark, and a genetic algorithm from Heuristics Lab (Wagner & Affenzeller, 2004) software.
A genetic algorithm (GA) is one of the heuristic methods used to find approximate solutions to NP-complete problems, GA is inspired by the Darwinian principles of the evolution of the species, and use own techniques of the genetics, such as: inheritance, mutation, natural selection and recombination (or crossover). The simplest form of genetic algorithm involves three types of operators: selection, crossover (single point), and mutation (Holland, 1975; Mitchell, 1998). Selection, this operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce. Crossover, this operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. The crossover operator roughly mimics biological recombination between two single-chromosome organisms. Mutation, this operator randomly flips some of the bits in a chromosome. Mutation can occur at each bit position in a string with some probability. The genetic algorithm (Wagner & Affenzeller, 2004) was used to verify the time and the quality of instances solution with the purpose of determining if the metric generated classify in complexity terms. The input parameters were: selection operator = Roulette, crossover operator = OX, mutation operator = Simple Inversion, generations = 1000, population size = 100, mutation rate = 0.05, replacement strategy = Elitism, crossover rate = 1, n-Elitism = 1, tournament group size = 2.

4.1 Experimentation in ATSP

Using the process contained in the general methodology to ATSP, the steps they would be of the following way:
Step 1. The maximum instance solved for ATSP is of 1.904.711 cities or World TSP (Applegate et al., 2006).
Step 2. The instances parameters of general ATSP can be codified in a hypothetical language: L = "nc, d_1, d_2, …, d_z". In other words two general parameters nc = number of cities and d_z =distances between the cities.
Step 3. Elaboration of the instances complexity metric for ATSP. In order to create the complexity general metric, for which it is necessary to determine *PS* (equation 12), *AM* (equation 13), *SD* (equation 14), *GD* (equation 15), *RA* (equation 16) indicators.

$$PS = \frac{nc}{nc\max} \qquad (12)$$

Where: *PS* is the average size of the instance at the rate of the instance maximum solved of ATSP, *nc* is the value of the problem to solve and *ncmax* is the size of greater instance solved.

$$AM = \sum_{i=1}^{n} \frac{d_i}{nc} \tag{13}$$

Where: *AM* is the sum of the arithmetic mean of each parameter (that has several elements) of the instance, d_i are the elements of the parameter that has several elements called matrix of cities and *nc* is the number of cities.

$$SD = \sqrt{\left(\sum_{i=1}^{n} \left(AM - \left(\frac{d_i}{nc} \right) \right) \right)^2} \tag{14}$$

Where: *SD* is the sum of the standard deviations of each type of parameter (that has several elements), d_i = Value of distances between cities *i*, *nc* = number of cities, *AM* = index of the arithmetic mean of distances.

$$GD = \begin{cases} \dfrac{\sum Frequency(d_i) > 1, \text{ if } Frequency(d_i) > 1 \text{ for some } i}{0, \text{ otherwise}} \end{cases} \tag{15}$$

Where: *GD* is the sum of the genetic distances which quantifies the similarity or differentiates between the populations from the frequencies of the elements from each parameter of the problem.

$$RA = \frac{\min d_i}{\max d_i} \tag{16}$$

Where: *RA* is the sum of the existing reasons between the greatest value and the smallest value of each parameter (that has several elements) of the NP-hard problem, min d_i = minimal distance between cities, max d_i = value of maximum distances between cities.
In Table 4 are the results of applying the instances complexity metric (IC) on ATSP instances, in addition to the summary of the obtained result to execute algorithm GA with the ATSP instances.

Instances	*nc*	*GA*		*PS*	*AM*	*SD*	*RA*	*GD*	*IC*
		d	*t*						
br17	17	0	0.79	0.000008	12.12	18.45	0.16	0.53	0.31
ft53	53	45.2	1.72	0.000027	283.15	377.02	0.15	0.32	6.60
ft70	70	28.8	3.68	0.000036	721.74	434.82	0.27	0.24	11.57
ftv33	33	42	3.21	0.000017	88.47	61.93	0.26	0.22	1.50
ftv35	35	26.6	3.22	0.000018	139.07	64.33	0.41	0.36	2.04
ftv38	38	38.1	3.09	0.000019	68.14	62.52	0.20	0.26	1.31
ftv44	44	56.3	3.17	0.000023	70.18	63.31	0.21	0.30	1.34
ftv47	47	43.8	3.09	0.000024	145.40	65.45	0.41	0.48	2.11
ftv55	55	94.0	3.16	0.000028	88.99	61.52	0.28	0.56	1.51
ftv64	64	75.9	2.91	0.000033	21.83	68.26	0.06	0.65	0.90
ftv70	70	88.1	3.61	0.000036	22.39	69.10	0.07	0.71	0.92
ftv170	170	325	3.91	0.000089	77.18	69.30	0.21	1.70	1.48
rbg443	443	124	6.33	0.000232	1.77	8.27	0.03	894.00	9.04
ry48p	48	1067	3.04	0.000025	264.11	578.73	0.96	0.12	8.43

Table 4. Results obtained from descriptive statistical measures with the ATSP instances

Where: *d* is the difference between best know and the obtained solution, *t* is the solution time for the instance on the GA algorithm. In the instance complexity (*IC*) between ft70 and ftv70 instances exist different values for instances with equal number of cities (*nc*), indicates that the ft70 instance is more complex than ftv70 instance and is verified with the obtained solution to apply the GA algorithm to the instances. In addition to being able to compare between the ftv70 and ftv64 instances, the ftv70 instance is more complex than ftv64 instance.

4.2 Experimentation in JSSP

Using the methodology with problem JSSP, are the following steps:

Step 1. The maximum instance resolved is of 20 by 20 symmetrical (Yamada & Nakano, 1997).

Step 2. The parameters of the instances of the JSSP can be codified in a hypothetical language: L = "nj, nm, $J_1(m_0, pt_0, \ldots m_{nm\text{-}1}, pt_{\text{nm-1}}), \ldots, J_{nj}$ $(m_0, pt_0, \ldots, m_{nm\text{-}1}, pt_{nm\text{-}1})$. Where nj = Number of jobs, nm = number of machines, J = Jobs, m = machine and pt = processing time.

Step 3. Elaboration of the complexity metric for JSSP. In order to create the complexity general metric is used the equation 1, for which it is necessary to determine *PS* (equation 17), *AM* (equation 18), *SD* (equation 19), *DG* (equation 20), *RA* (equation 21) indicators.

$$PS = \left(\frac{nj * nm}{MAXJSSP} \right) \tag{17}$$

Where: PS is the so large average of the instance at the rate of the maximum instance from problem JSSP. And $MAXJSSP$ = the maximum instance solved in literature or MAX($nj * nm$).

$$AM = \sum_{i=1}^{z} \frac{m_i}{m_{(nj*nm)}} + \sum_{i=1}^{z} \frac{pt_i}{pt_{(nj*nm)}} \tag{18}$$

Where the *AM* metric is the sum of the arithmetic mean of the *m*, *pt* parameters.

$$SD = \sqrt{\left(\sum_{i=1}^{n} \left(AMm - \left(\frac{m_i}{m_{(nj*nm)}} \right) \right) \right)^2} + \sqrt{\left(\sum_{i=1}^{n} \left(AMpt - \left(\frac{pt_i}{pt_{(nj*nm)}} \right) \right) \right)^2} \tag{19}$$

Where the *SD* metric contains the sum of the standard deviations of the *m*, *pt*, *J* parameters. And *MA* is the arithmetic mean of *m* parameter; *MApt* is the arithmetic mean of *pt* parameter.

$$GD = [Frequency(m) + Frequency(pt)] / 100 \tag{20}$$

Where the *GD* metric contains the genetic distances between the populations by the frequencies of the parameters (*m*, *pt*).

$$RA = \frac{MIN\ m}{MAX\ m} + \frac{MIN\ pt}{MAX\ pt} \tag{21}$$

Where the *RA* metric contains the sum of the reasons between the minimal and maximuml values of the *m* and *pt* parameters.

The Table 5 contains the results of applied the complexity metric for JSSP, and the results of the GA on JSSP instances. This can be observed that in the Instance Complexity (*IC*) between la01 and la05 instances exist different values with equal Number of Jobs and Number of Machines, indicates that the la01 instance is more complex than the la05 instance is verified with the quality of the obtained solution. Also sample that in between the yn1 and yn4 instances, the yn4 instance is more complex that yn1 instance. Where: *d* is the difference between best know and the obtained solution, *t* is the solution time for the instance on the GA algorithm.

Instances	*J*M*	*GA*		*TP*	*MA*	*DA*	*RA*	*DG*	*IC*
		d	*t*						
abz5	10x10	4.29	0:55.3	0.25	73.64	2302.69	0.505	0.10	23.77
abz6	10x10	3.39	1:09.3	0.25	58.51	1830.02	0.204	0.10	18.89
abz8	20x15	16.6	2:00.8	0.75	29.12	2247.37	0.275	0.20	22.77
abz9	20x15	14.3	1:46.5	0.75	29.28	1862.31	0.275	0.02	18.92
ft06	6x6	0	0:08.5	0.09	7.25	156.57	0.100	0.06	1.64
ft10	10x10	8.06	0:28.4	0.25	51.64	1615.42	0.020	0.10	16.67
ft20	20x5	3.43	0:35.3	0.25	51.16	2280.33	0.020	0.20	23.31
la01	10x5	0	0:13.7	0.12	53.82	1177.07	0.122	0.10	12.31
la02	10x5	0.15	0:15.2	0.12	50.24	1098.87	0.121	0.10	11.49
la03	10x5	4.35	0:04.3	0.12	44.22	966.58	0.076	0.10	10.11
la04	10x5	2.03	0:14.5	0.12	46.22	1010.19	0.051	0.10	10.56
la05	10x5	0	0:13.5	0.12	40.06	874.45	0.051	0.10	9.14
la06	15x5	0	0:22.3	0.18	51.78	1730.97	0.071	0.10	17.83
la07	15x5	0	0:25.4	0.18	48.13	1608.84	0.082	0.10	16.57
la08	15x5	0	0:25.9	0.18	49.05	1639.64	0.051	0.10	16.88
la09	15x5	0	0:24.6	0.18	54.62	1825.91	0.070	0.10	18.80
la10	15x5	0	0:23.1	0.18	53.28	1781.12	0.051	0.10	18.34
orb01	10x10	5.00	0:28.7	0.25	53.36	1668.86	0.050	0.10	17.22
orb02	10x10	5.96	0:23.4	0.25	50.91	1591.87	0.060	0.10	16.43
orb03	10x10	9.65	0:24.4	0.25	52.79	1651.21	0.050	0.10	17.04
yn1	20x20	0.78	1:15.8	1.00	36.12	3217.55	0.204	0.25	32.55
yn2	20x20	2.78	1:18.1	1.00	36.23	3227.61	0.204	0.26	32.65
yn3	20x20	2.07	1:19.0	1.00	32.07	3169.99	0.204	0.27	32.07
yn4	20x20	0.80	1:25.8	1.00	36.84	3281.54	0.204	0.31	33.19

Table 5. Results obtained by the descriptive statistics with the JSSP instances

4.3 Experimentation in VRPTW

Using the methodology with VRPTW, the following steps are obtained:

Step 1. The maximum instance solved of problem VRP is F-n135-k7, with VN = 12, C = 2210, CN = 135 (Toth & Vigo, 2001).

Step 2. The parameters of the instances of problem VRPTW can be codified in a hypothetical language: L = "VN, C, (CN_1, XCO_1, YCO_1, D_1, RT_1, DT_1, ST_1, …, CN_z, XCO_z, YCO_z, D_z, RT_z, DT_z, ST_z)". Where VN = Vehicle Number, C = Capacity, CN = Customer Number, XCO = X Coord., YCO = Y Coord., D = Demand, RT = Ready Time, DT = Due date, ST = Service Time.

Step 3: Elaboration of the complexity metric for problem VRPTW. In order to create the complexity general metric the equation 1 is used, for which it is necessary to determine *PS* (equation 22), *AM* (equation 23), *SD* (equation 24), *GD* (equation 25), *RA* (equation 26) indicators.

$$PS = \frac{VN * C * CN_n}{VRPTWMAX},\ VRPTWMAX = MAX(VN * C * CN_n) \tag{22}$$

Where: PS = is the problem size of the instance at the rate of the maximum instance solved from VRPTW, max($VN * C * CN_n$) = the value of the maximum instance solved by literature.

$$AM = \sum_{i=1}^{n} \frac{XCO_i}{CN_z} + \sum_{i=1}^{n} \frac{YCO_i}{CN_z} + \sum_{i=1}^{n} \frac{D_i}{CN_z} + \sum_{i=1}^{n} \frac{RT_i}{CN_z} + \sum_{i=1}^{n} \frac{DT_i}{CN_z} + \sum_{i=1}^{n} \frac{ST_i}{CN_z} \tag{23}$$

Where: In AM contains the sum of the arithmetic mean of the XCO, YCO, D, RT, DT y ST parameters.

$$\begin{aligned} SD = & \sqrt{\left(\sum_{i=1}^{n}\left(AMXCO - \left(\frac{XCO_i}{CN_z}\right)\right)\right)^2} + \sqrt{\left(\sum_{i=1}^{n}\left(AMYCO - \left(\frac{YCO_i}{CN_z}\right)\right)\right)^2} + \\ & \sqrt{\left(\sum_{i=1}^{n}\left(AMD - \left(\frac{D_i}{CN_z}\right)\right)\right)^2} + \sqrt{\left(\sum_{i=1}^{n}\left(AMRT - \left(\frac{RT_i}{CN_z}\right)\right)\right)^2} + \\ & \sqrt{\left(\sum_{i=1}^{n}\left(AMDT - \left(\frac{DT_i}{CN_z}\right)\right)\right)^2} + \sqrt{\left(\sum_{i=1}^{n}\left(AMST - \left(\frac{ST_i}{CN_z}\right)\right)\right)^2} \end{aligned} \tag{24}$$

Where: SD contains the sum of the standard deviations of the XCO, YCO, D, RT, DT y ST parameters. $AMXCO$ is the arithmetic mean of XCO parameter, $AMYCO$ is the arithmetic mean of YCO parameter, AMD is the arithmetic mean of D parameter, $AMRT$ is the arithmetic mean of RT parameter, $AMDT$ is the arithmetic mean of DT parameter, and $AMST$ is the arithmetic mean of ST parameter.

$$\begin{aligned} GD = [& Frequency(XCO) + Frequency(YCO) + \\ & Frequency(D) + Frequency(RT) + \\ & Frequency(DT) + Frequency(ST)]/100 \end{aligned} \tag{25}$$

Where: GD contains the genetic distances between populations from the frequencies of each parameter (XCO, YCO, D, RT, DT y ST).

$$\begin{aligned} RA = & \left(\frac{MIN\ XCO}{MAX\ XCO}\right) + \left(\frac{MIN\ YCO}{MAX\ YCO}\right) + \\ & \left(\frac{MIN\ D}{MAX\ D}\right) + \left(\frac{MIN\ RT}{MAX\ RT}\right) + \\ & \left(\frac{MIN\ DT}{MAX\ DT}\right) + \left(\frac{MIN\ ST}{MAX\ ST}\right) \end{aligned} \tag{26}$$

Where RA contains the sum of the reasons between the minimal and maximuml values of the XCO, YCO, D, RT, DT y ST parameters.

In Table 6 are the results of applying the complexity metric on VRPTW instances, in addition to the summary of the obtained result to execute algorithm GA with the VRPTW instances. Where: d is the difference between best know and the obtained solution, t is the solution time for the instance on the GA algorithm, VN = Vehicle Number, C = Capacity, CN = Customer Number. It can be observer that in the Instance Complexity (IC) between

c204 and c201 different values for instances with equal Vehicle Number Capacity and Customer Number exist, this indicates that the instance c204 is more complex than c201 and is verified with the obtained solution to apply GA to the instances.
It is necessary to mention that the values of complexity of the instances of single VRPTW are comparable between instances of the same problem.

Instances	*VN*	*C*	*CN*	*GA*		*TP*	*MA*	*DA*	*RA*	*DG*	*IC*
				d	*t*						
c101	25	200	25	0	0:58	0.034	1143.92	28598	1.017	0.10	297.43
c102	25	200	25	0.05	1:03	0.034	1158.68	28967	1.022	0.17	301.26
c103	25	200	25	0	1:00	0.034	1284.92	32123	1.037	0.23	334.09
c104	25	200	25	0	1:01	0.034	1303.68	32592	1.037	0.27	338.97
c105	25	200	25	0	1:26	0.034	1151.12	28778	1.060	0.10	299.30
c201	25	1000	25	0	1:35	0.122	3657.96	91449	0.894	0.08	951.08
c202	25	700	25	0	1:34	0.122	3898.96	97474	0.894	0.15	1013.74
c203	25	700	25	0	2:03	0.122	4032.08	100802	0.897	0.21	1048.35
c204	25	700	25	0.1	1:48	0.122	4081.20	102030	1.263	0.25	1061.12
c205	25	700	25	0	1:44	0.122	3675.56	91889	0.941	0.08	955.65
r101	25	200	25	0.06	1:40	0.034	310.36	7759	0.431	0.09	80.69
r102	25	200	25	1.37	1:11	0.034	318.44	7961	0.431	0.16	82.80
r103	25	200	25	1.12	1:19	0.034	317.96	7949	0.558	0.22	82.67
r104	25	200	25	0.16	1:54	0.034	309.56	7739	0.558	0.26	80.49
r105	25	200	25	2.15	1:20	0.034	310.68	7767	0.484	0.09	80.78
r201	25	700	25	1.19	1:49	0.174	1058.32	26458	0.458	0.09	275.17
r202	25	700	25	0.39	2:09	0.174	1122.44	28061	0.492	0.16	291.84
r203	25	700	25	0.83	2:05	0.174	1146.04	28651	0.549	0.22	297.97
r204	25	700	25	2.66	2:04	0.174	1114.84	27871	0.549	0.26	289.86
r205	25	700	25	0	2:00	0.174	1057.80	26445	0.509	0.09	275.03
rc101	25	700	25	0	1:22	0.034	326.72	8168	0.329	0.08	84.95
rc102	25	700	25	0	1:17	0.034	327.72	8193	0.329	0.15	85.21
rc103	25	700	25	0	2:09	0.034	323.36	8084	0.425	0.21	84.08
rc104	25	700	25	0.11	1:27	0.034	313.28	7832	0.425	0.25	81.45
rc105	25	700	25	0.44	1:15	0.034	335.04	8376	0.383	0.08	87.11
rc201	25	700	25	0	1:42	0.174	1024.00	25600	0.220	0.08	266.24
rc202	25	700	25	0	2:03	0.174	1077.24	26931	0.220	0.15	280.08
rc203	25	700	25	0.3	2:38	0.174	1099.20	27480	0.365	0.21	285.79
rc204	25	700	25	0	2:07	0.174	1067.52	26688	0.365	0.25	277.56
rc205	25	700	25	0	2:14	0.174	1044.88	26122	0.333	0.08	271.67

Table 6. Results obtained by metrics descriptive statistics with VRPTW instances

5. Conclusions

We can conclude that the creation of a mathematical expression based on the descriptive statistics able is possible to measure the complexity the instances of combinatorial optimization problems, in this paper was demonstrated for ATSP, VRPTW and JSSP. It is necessary to mention that, the values of instances complexity of ATSP, VRPTW and JSSP problems are only comparable between instances of the same problem.
The intention of this paper was to classify the complexity of the instances and not therefore the complexity between problems NP, as future works, we propose to validate the methodology for other NPs problems.

6. References

Applegate, D.L.; Bixby, R.E.; Chvátal Y. & Cook, W.J. (2006). *The Traveling Salesman Problem: A Computational Study,* Princeton University Press, ISBN:0691129932, New Jersey

Blanton, J.L. & Wainwright, R.L. (1993). Multiple Vehicle Routing with Time and Capacity Constrainst using Genetic Algorithms, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 452-459, ISBN:1-55860-299-2, San Mateo, CA, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Chakrabortya, U.K. (2008). Including Special Section: Genetic and Evolutionary Computing, *Information Sciences*, Vol. 178, N. 23, 4419-4420, ISSN: 0020-0255

Chen, C.L. & Bulfin R.L. (1990). Scheduling unit processing time jobs on a single machine with multiple criteria. *Computers and Operations Research*, Vol. 17, N. 1, 1–7, ISSN: 0305-0548.

Cook, S.A. (1971). The Complexity of Theorem Proving Procedures, *Proceedings of 3rd ACM Symposium on Theory of Computing*, pp. 151-158, Shaker Heights, Ohio, United States, May 03–05, 1971, ACM, New York, NY, USA

Cook, S.A. (1983). An Overview of Computational Complexity. *Communications of the ACM,* Vol. 26, No.6, (June 1983) 400-408, ISSN: 0001-0782

Cormen, H.T.; Leiserson,C.E.; Rivest, R.L. & Stein, C. (2001). *Introduction to Algorithms,* MIT Press, ISBN:0262032937, Cambridge , Massachusetts London, England

Cruz-Chávez, M.A.; Díaz-Parra, O.; Juárez-Romero, D. & Martínez-Rangel, M.G. (2008). Memetic Algorithm Based on a Constraint Satisfaction Technique for VRPTW, In: *Artificial Intelligence and Soft Computing – ICAISC 2008, Vol. 5097,* Rutkowski, N L. et al. (Eds.), 376-387, Springer-Verlag, ISBN: 978-3-540-69572-1

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability, a Guide to the Theory of NP-completeness,* W. H. Freeman and Company, ISBN:0-7167-1044-7, New York

Gładysz. B. (2007). Fuzzy robust courses scheduling problem, *Fuzzy Optimization and Decision Making*, Vol. 6, N. 2, 155-161, ISSN: 1568-4539

Dahal, K.P.; Hossain,A.; Varghese,B.;Abraham,A.; Xhafa,F. & Daradoumis, A. (2008). Scheduling in Multiprocessor System Using Genetic Algorithms, *7th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'08)*, pp. 277-282, ISBN 978-0-7695-3184-7, 2008, IEEE Computer Society press, USA.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems,* University of Michigan Press, Ann Arbor

Jain, L.C. ; Tan, S.C. & Lim, C.P. (2008). An Introduction to Computational Intelligence Paradigms. *In: Computational Intelligence Paradigms, Vol. 137*, Studies in Computational Intelligence. Lakhmi C. Jain et al. (Eds.), 1-23, Springer-Verlag. ISBN: 978-3-540-79473-8

Jonsson, P. & Bäckström, C. (1995). Complexity Results for State-Variable Planning under Mixed Syntactical and Structural Restrictions, *Proceedings of the sixth international conference on Artificial intelligence: methodology, systems, applications*, pp. 205-213, ISBN:981-02-1877-X, Smolenice Castle, Slovakia, 1994, World Scientific Publishing Co., Inc. River Edge, NJ, USA

Karp, R.M. (1972). Reducibility Among Combinatorial Problems, In: *Complexity of Computer Computations,* R.E. Miller and J.W. Thatcher, (Ed.), 85-104, Springer, ISBN: 0306307073

Lim, C.P.; Jain, L.C.; Nguyen N.T. & Balas, V.E. (2008). Guest Editorial: Special issue on advances in computational intelligence paradigms and applications, *Fuzzy Optimization and Decision Making*. Vol. 7, N. 3, 215-217, ISSN 1568-4539

McCabe, T.J. (1976). A complexity measure, *Transactions on Software Engineering,* Vol.2, N.4, (July 1976) 308-320, ISSN:0098-5589

Mitchell, M. (1998). *An Introduction to genetic algorithms,* MIT Press, ISBN 0−262−13316−4, London, England

Papadimitriou, C. & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity,* Prentice-Hall, ISBN: 0-201-53082-1, New Jersey

Pérez, J.; Pazos, R.A.; Frausto, J.; Rodríguez, G.; Romero, D. & Cruz, L. (2004). A Statistical Approach for Algorithm Selection, *Proceedings of III International Workshop on Experimental an Efficient Algorithms*, pp. 417-431, ISBN 3-540-22067-4, Angra dos Reis, Brazil, May 25-28, 2004, Springer-Verlag, Berlin

Reinelt, G. (1991). TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing,* Vol. 3, N. 4, 376-384, ISSN: 0899-1499

Ruiz-Vanoye, J.A.; Díaz-Parra, O. & Landero, V. (2008). A Metric to Discriminate the Selection of Algorithms for General Problem ATSP, In: *Knowledge-Based Intelligent Information and Engineering Systems – KES 2008*, Vol. 5177, I. Lovrek, R.J. Howlett, and L.C. Jain (Eds.), 106-113, Springer-Verlag, ISBN: 978-3-540-85562-0, Berlin

Ruiz-Vanoye, J.A.; Zárate, J.A.; Díaz-Parra, O. & Landero, V. (2008). Applied Statistical Indicators to the Vehicle Routing Problem with Time Windows for Discriminate Appropriately the Best Algorithm, In: *Computational Science and Its Applications – ICCSA 2008*, Vol. 5073, Part II, O. Gervasi et al. (Eds.), 1131-1141, Springer-Verlag, ISBN: 978-3-540-69840-1, Berlin

Shannon, C.E. & Weaver, W. (1949). *The Mathematical Theory of Communication,* University of Illinois Press, ASIN: B000UFWID2, Urbana, Illinois

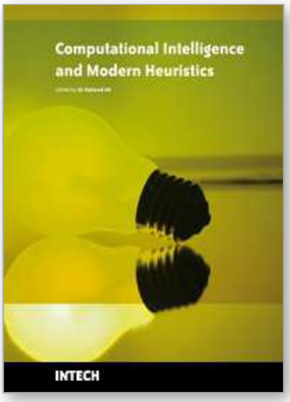
Solomon, M.M. (1987). Algorithms for Vehicle Routing and Scheduling Problems with Time Window Constrains. *Operations Research,* Vol. 35, No. 2 (March-April 1987) 254-265, ISSN:0030-364X

Toth, P. & Vigo, D. (2001). *The Vehicle Routing Problem,* Society for Industrial and Applied Mathematic (SIAM), ISBN: 0898714982, Philadelphia

Thangiah, S.R. (1998). Genetic Algorithms, Tabu Search and Simulated Annealing Methods for Vehicle Routing Problems with Time Windows, In: *Practical Handbook of Genetic Algorithms: Complex Structures*, Vol. III: Complex structures, L. D. Chambers (Eds.), 347-381, CRC Press , ISBN: 0-8493-2539-0, Florida

Wagner, S. & Affenzeller, M. (2004). The HeuristicLab Optimization Environment. *Technical Report,* Institute of Formal Models and Verification. Johannes Kepler University Linz. Austria

Yamada, T. & Nakano, R. (1997). Genetic Algorithms for Job-Shop Scheduling Problems, *Proceedings of Modern Heuristic for Decision Support*, pp. 67-81, London, 18-19 March 1997, UNICOM seminar


Computational Intelligence
and Modern Heuristics
INTECH