2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Scientific Algorithms for the Car Renter Salesman Problem

Denis Felipe, Elizabeth Ferreira Gouvêa Goldbarg, Marco Cesar Goldbarg
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, Brazil
denisfelipe.natal@gmail.com, beth@dimap.ufrn.br, gold@dimap.ufrn.br

***Abstract*—This paper presents the Scientific Algorithms, a new metaheuristics inspired in the scientific research process. The new method introduces the idea of *theme* to search the solution space of hard problems. The inspiration for this class of algorithms comes from the act of researching that comprises thinking, knowledge sharing and disclosing new ideas. The ideas of the new method are illustrated in the Traveling Salesman Problem. A computational experiment applies the proposed approach to a new variant of the Traveling Salesman Problem named Car Renter Salesman Problem. The results are compared to state-of-the-art algorithms for the latter problem.**

Keywords—car renter salesman problem; metaheuristics; scientific algorithms; scientific research

I. INTRODUCTION

Research is one of the essential processes for the construction of human insight, responsible for creating new knowledge and to develop understanding in different areas. The act of researching consists of seeking the truth by means of a formal procedure, a method of reflective thinking, allowing the discovery of new facts or data, relationships or laws in any field of knowledge, conducting a scientific process. The scientific method is a systematic procedure based on logic, rationality, efficiency and effectiveness, in order to assist decision-making of researchers in the task of producing scientific knowledge [1].

The scientific research processes inspired a new population based metaheuristics, named Scientific Algorithms. Unlike classical evolutionary algorithms in which individuals compete to reproduce and perpetuate their genetic, characteristics, research suggests cooperation between individuals for intellectual development, perpetuating their ideas. In this context, the intellectual evolution is just as natural and important to human beings as the genetic evolution, contributing significantly to survival. These algorithms bring the new idea of a *theme* to be researched, concentrating the search effort in different sets of variables during execution.

This paper presents the Scientific Algorithms and illustrates its main steps with a didactic example on the Traveling Salesman Problem (TSP). To investigate the potential of the proposed class of algorithms, the approach is applied to a variant of the TSP named Car Renter Salesman Problem (CaRS) [2]. Several metaheuristics were applied to CaRS such as GRASP, VND, Ant Colony Optimization, Memetic Algorithms [2] and Transgenetic Algorithms [3].

This paper is organized in four sections, besides this one. Section II introduces the fundamentals of the proposed approach and section III its application to CaRS. Section IV reports the results of computational experiments on twenty instances with size ranging from 14 to 300 vertices. Finally, section V presents conclusions and remarks about future works.

II. SCIENTIFIC ALGORITHMS

This section presents the basic concepts of the Scientific Algorithms. The TSP is utilized as a didactic example to illustrate the main concepts of the algorithm. Given a weighted graph $G = (V, E)$, where $V = (1, 2, \ldots, n)$ is the set of vertices, $E = \{(i, j) : i, j \in V, i \neq j\}$ is the set of edges and $C = [c_{ij}]$ is the cost of the edge linking vertex i to vertex j, the TSP consists in finding the minimum cost Hamiltonian cycle in G [4]. The TSP is NP-Hard [5] and also one of the most intensely researched problems in Combinatorial Optimization. A review of the TSP is presented in [6].

In the scientific algorithms, the individuals of a scientific community are represented as a population of candidate solutions. In this paper, they are referred as *researchers*. The research topic corresponds to a set of variables of the investigated problem. The variables of the research topic are used to delimit the scope of the search, avoiding irrelevant regions to be explored in the solution space. In this paper, it is referred as *search theme* (or just *theme*, for short). The literature is thought as memory, or significant data about the search stored in a repository of information. This memory is used to bias the search, improving diversification or intensification. In this paper, it is referred as *literature*. The interactions of these three contexts result in the search procedure of the scientific algorithms.

Fig. 1 presents the general framework of a scientific algorithm. The initial population of researchers, referred as *Scientific_community*, is created in step 1 and the literature in step 2. Steps 3 through 11 are the main loop of the scientific algorithm. Step 5 generates a theme. The theme is a set of variables of the problem that will be the focus of the search. It

limits the scope to a restricted area of the space of solutions. Step 6 creates a new solution based on the current solution (researcher), the literature and the theme. The new solution must be as similar as possible to the researcher, except for the variables related to the theme, referred as *thematic variables*. The method used to set new values to thematic variables can be improved with additional information coming from literature. This procedure simulates the formulation of hypotheses in the proposed algorithmic metaphor. A *hypothesis* is a tentative solution. It is simulated using the researcher knowledge and the literature to build a provisory solution to the problem. Step 7 searches for improvements on the new generated solution perturbing the thematic variables in the hypothesis. The variables in the solution that are not related to the theme should not be directly affected by this procedure. This step simulates the verification of hypotheses, colleting, classifying, analyzing and interpreting the research data to support or not the formulated hypothesis. Step 8 updates the researcher if some improvement was found, replacing it by the new solution. This step simulates the personal learning of the researcher from the investigation. Step 9 updates the literature, storing useful data regarding the theme. This process simulates the publication of scientific works, feeding the literature to increase the shared knowledge of the scientific community. Step 12 returns the best solution found during the execution of the scientific algorithm.

Scientific Algorithm

1. **Let** *Scientific_community* be the population of solutions
2. **Let** *Literature* be the repository of information
3. **Do**
4. **For** each *Researcher* in *Scientific_community* **do**
5. *Theme* ← *Select_theme* (*Researcher*, *Literature*)
6. *Hypothesis* ← *Raise_hypothesis* (*Researcher*, *Theme*, *Literature*)
7. *Verify_hypothesis* (*Hypothesis*, *Theme*)
8. *Update* (*Researcher*, *Hypothesis*)
9. *Publish* (*Researcher*, *Hypothesis*, *Theme*, *Literature*)
10. **End for**
11. **While** stopping criterion is not met
12. **Return** the best solution found

Fig. 1. Pseudo-code of the scientific algorithm.

The development of solutions in a scientific algorithm is continuous, seeking to improve the set of candidate solutions iteratively. The iteration of the algorithm consists of selecting sets of variables of the problem and optimizing them in the solutions of the population. This procedure can be performed in parallel. The optimization process is a stochastic search with bias. This set of characteristics associates the scientific algorithms to the category of evolutionary algorithms [7].

One of the most obvious differences between scientific algorithms and genetic algorithms [8], memetic algorithms [9] and cultural algorithms [10] is the absence of the genetic context. Another fundamental difference between scientific algorithms and cultural algorithms is the representation of the concept of knowledge. Cultural algorithms regard knowledge as a mechanism to guide the search process, essentially supported on a genetic algorithm. In the scientific algorithms, knowledge is the very solution of the problem. While cultural algorithms treat the development of knowledge as a secondary task, the scientific algorithms treat the evolution of knowledge as an existential goal. Finally, a feature that makes the scientific algorithms unique is the concept of theme. Although it can be artificially simulated by other approaches, the theme is more than a simple additional structure in the memory. In the scientific algorithms, the theme manages the entire search, controlling the changes in researchers and in literature and the interactions of these contexts.

To elucidate the process of the scientific algorithms, the steps of the general algorithm are illustrated on the TSP. Fig. 2 shows the weighted graph that represents the TSP instance used in this example.

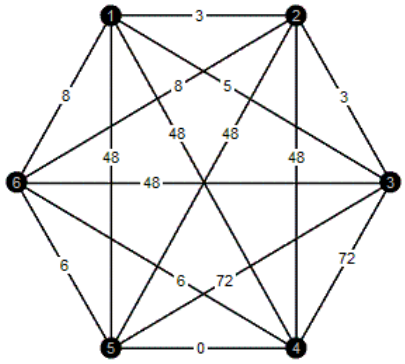

Fig. 2. TSP graph of the didactic example.

A solution to the TSP can be represented by a sequence of vertices visited in the graph. Any construction method can be used to generate the initial solutions, for instance, random generation. The literature should contain significant data to assist the generation of good solutions. In this example, the literature is used to achieve better diversification. It consists in a square matrix, $L = [l_{ij}]$, of order m, $m = |E|$, where l_{ij} is the number of times the edge (i, j) is currently used in the population divided by the population size.

Fig. 3 illustrates a researcher, S, randomly generated, for the graph in Fig. 2. S is represented as {1 - 4 - 3 - 5 - 2 - 6}. The cost of S is 256, the sum of the weights of the edges in the Hamiltonian cycle.

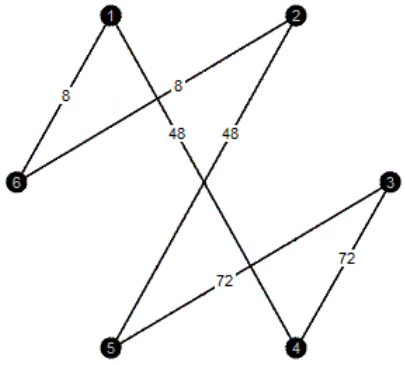

Fig. 3. TSP researcher of the didactic example.

The optimization of S starts by choosing the theme. This decision involves the size of the theme and the assigned values to its variables, chosen at random or by a defined method. In this example, the theme is defined as a set of 4 sequential vertices in S (a path of length 3), randomly chosen in procedure

Select_theme( ) as t = {3, 5, 2, 6}. The thematic variables are the edges connected to those vertices, i.e., *theme* = {(4,3), (3,5), (5,2), (2,6), (6,1)}.

The construction of the hypothesis h, implemented in procedure *Raise_hypothesis*( ), is shown in Fig. 4. In this example, a copy of the researcher is created. Then, all vertices in the theme are removed from the initial solution, creating an intermediary solution containing vertices 1 and 4. The intermediary solution is initialized as the incomplete route h = {1 - 4}. A constructive method is used to create a path containing the vertices of the theme. The constructive method used to complete the solution is based on the *nearest neighbor* algorithm for the TSP [6]. Starting from an arbitrary vertex a in *theme*, the algorithm consists in visiting this vertex and finding the closest non-visited vertex in *theme*, b. If edge (a, b) has a frequency over 50%, i.e., $l_{ab} > 0.5$, a random vertex in *theme*, including b, is chosen to replace b. The procedure is restarted from the last vertex and repeats until all vertices in *theme* are visited, when a path containing the vertices of *theme* is built. Finally, the path is inserted in h at the original position of the removed vertices, in the direction that leads to the lowest cost of the solution. In this example, the initial vertex was 2. To simplify the process, it is assumed that no edge has frequency over 50%. At the end of this operation, the solution h = {1 - 4 - 5 - 6 - 3 - 2} with cost 108 is set as the hypothesis.

Theme	Initial solution	Intermediate Solution
{3, 5, 2, 6}	{1 – 4 – 3 – 5 – 2 – 6}	{1 – 4}
Current Vertex	**Intermediate route**	**Nearest Neighbor**
2	{2}	3
3	{2 – 3}	6
6	{2 – 3 – 6}	5
5	{2 – 3 – 6 – 5}	–
Direction	**Resulting solution**	**Cost**
Forward	{1 – 4 – 2 – 3 – 6 – 5}	201
Backward	{1 – 4 – 5 – 6 – 3 – 2}	108

Fig. 4. Construction of the solution in the didactic example.

After constructing the hypothesis h, the scientific algorithm proceeds to the perturbation in the procedure *Verify_hypothesis*( ), as illustrated in Fig. 5. In the example, the solution h is submitted to local search with the 2-*shift* neighborhood. A solution h' is a neighbor of h if there is a switching between two vertices in t that transforms h in h'. After the perturbation, the solution h = {1 - 4 - 5 - 6 - 2 - 3} is found with cost 70. Once the hypothesis h has a cost better than S, the former replaces the latter in the current population. This step is executed in procedure *Update*( ). Once the population is updated, the matrix representing the literature has to be updated as well, in procedure *Publish*( ).

Theme	Initial solution	Cost
{3, 5, 2, 6}	{1 – 4 – 5 – 6 – 3 – 2}	108
Switching vertices	**Resulting solution**	**Cost**
(3, 5)	{1 – 4 – 3 – 6 – 5 – 2}	225
(2, 3)	{1 – 4 – 5 – 6 – 2 – 3}	70
(3, 6)	{1 – 4 – 5 – 3 – 6 – 2}	179
(5, 2)	{1 – 4 – 2 – 6 – 3 – 5}	272
(5, 6)	{1 – 4 – 6 – 5 – 3 – 2}	138
(2, 6)	{1 – 4 – 5 – 2 – 3 – 6}	155

Fig. 5. Perturbation of the solution in the didactic example.

Those steps are repeated for all researchers until the stopping criterion is met, returning the best solution found by the scientific algorithm.

III. Application to the CaRS

CaRS is a generalization of the TSP where one must visit a given set of cities, starting and finishing in the same city, using rental vehicles for transportation. The goal is to perform the route with the lowest possible cost. Various vehicle types are available for rent, each with its own characteristics and operating costs. These costs include fuel consumption, toll fees and the amount paid for the rental. In addition to these costs, there is an additional fee to be paid to return a vehicle to the city where it was rented, if it is delivered in a different city. CaRS is defined on a complete graph $G = (V, E)$, where V = {1, 2, …, n} is the set of vertices and $E = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. In addition, a set H = {1, 2, …, q} of cars is defined. A cost matrix $C^k = [c^k_{ij}]$ indicates the total rental cost per distance traveled, fuel and possible toll rates between any two cities i and j with vehicle k. Matrix $D^k = [d^k_{ij}]$ indicates the cost to return vehicle k rented in vertex i and delivered in vertex j, $i \neq j$, $d^k_{ij} = 0$ if $i = j$. The objective function is to minimize the total cost of the route plus the return cost of vehicles.

In [2] vertex 1 is considered the starting (and ending) point of the tour, thus, a vehicle must be rented at vertex 1. In this paper, it is considered the version of CaRS where each vehicle can be rented only once.

A. Researchers

Solutions are represented in 2-dimensional arrays, as illustrated in Fig. 6, where the tour is represented in one dimension and the vehicles in the other. The gray array represents the tour and the white represents the vehicles. In the solution represented in Fig. 6 vehicle 1 is rented in vertex 1 and delivered in vertex 5, vehicle 2 is rented in vertex 5 and delivered in vertex 4 and vehicle 3 is rented in vertex 4 and delivered in vertex 1.

Fig. 6. Representation of a solution.

The solutions are generated by a random procedure, using a random number of vehicles. The initial population of candidate solutions has 100 researchers. This value was used in [3] as the population size.

B. Literature

The literature consists in a list of 10 distinct vertices for each vertex in G, where L_i is the list corresponding to the i-th vertex. This list is used to store the source vertex of edges in good solutions. Computational experiments have shown that increasing the size of the list does not contribute significantly to finding better solutions. Initially, these vertices are set randomly. During the execution of the scientific algorithm,

they are replaced by the previous vertices of i in the best solutions found. In addition, the literature stores a pointer to the population of solutions.

C. Theme

The theme is a set of vertices, as represented in Fig. 7, built at random, with probability p of containing each vertex in G. More precisely, $p = 0.3$ when $n \leq 50$ and $p = 0.8$ when $n > 50$. This probability was defined empirically and a study of this parameter is shown in the computational experiments. The thematic variables are the arcs connected to the vertices in the theme and the rental of vehicles at those vertices. In the solution represented in Fig. 6, the thematic variables are the arcs (6, 1), (1, 2), (7, 3), (3,4), (5, 7) and vehicles 1 and 2 assigned to vertices 1, 3 and 7.

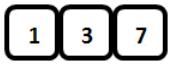

Fig. 7. Representation of a theme.

D. Hypotheses

The method used to generate hypotheses selects at random one of four constructive operators, γ_1, γ_2, γ_3 and γ_4, and applies it to the solution.

The operator γ_1 creates a copy from the researcher. In practice, this operator allows a solution to be directly optimized by the hypothesis verification method.

The operators γ_2, γ_3 and γ_4 creates a copy of the researcher then removes all vertices in the theme from the solution, one by one, removing their respective arcs and adding a new arc between the previous and next vertices, forming an initial route. The result of this procedure in the solution represented in Fig. 6 using the theme represented in Fig. 7 is the incomplete solution represented in Fig. 8. The operators γ_2, γ_3 and γ_4 use three different methods to reinsert the removed vertices in this solution.

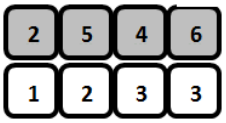

Fig. 8. Representation of an incomplete solution after the first step of operators γ_2, γ_3 and γ_4.

The method used to reinsert the removed vertices in γ_2 is based on the random insertion algorithm for the TSP [6]. One of the removed vertices is randomly chosen and inserted in the path between the two vertices that cause the smallest increase in cost using the same vehicle as the new previous vertex. This procedure is repeated for the remaining removed vertices until all vertices in G are part of the route, forming a feasible solution. Fig. 9 shows an example of the insertion of vertex 7 after vertex 4 in the incomplete solution represented in Fig. 8. The vehicle used in vertex 7 is the same used in vertex 4.

Fig. 9. Example of vertex insertion in an incomplete solution.

The method used to reinsert the removed vertices in γ_3 is inspired on the literature review in the metaphor of scientific research. For each vertex i in the theme, this procedure try to insert i in the path after the vertex in L_i that causes the smallest increase in cost using the same vehicle as the new previous vertex. Vertices that cannot be inserted in this way are added to the route by the random insertion method presented in the operator γ_2.

The method used to reinsert the removed vertices in γ_4 is inspired on the collaboration between two researchers in the metaphor of scientific research. This operator uses the pointer to the population of solutions in the literature. Let S_2 be a random solution from the population. For each vertex i in the theme, this procedure try to insert i in the path after the previous vertex of i in S_2 using the same vehicle as the new previous vertex. Vertices that cannot be inserted in this way are added to the route by the random insertion method, presented in the operator γ_2.

E. Hypotheses Verification

The method used to verify hypotheses applies five improvement operators to the solution, λ_1, λ_2, λ_3, λ_4 and λ_5, in a random sequence.

The goal of operator λ_1 is to insert a new vehicle in the hypothesis. For each vertex i in the theme, this procedure checks the cost variation of renting each unrented vehicle at i and delivering at some vertex j. If the cost of the perturbed hypothesis is improved, the new vehicle is rented at i and the solution is updated. Fig. 10 shows an example of this operator renting vehicle 4 at vertex 7 in the solution represented in Fig. 6. The new vehicle is delivered at vertex 4.

Fig. 10. Example of the operator λ_1.

The operator λ_2 aims at extending the route of a vehicle in the hypothesis. For each vertex i in the theme, this procedure checks the cost variation of anticipating the rental of the next vehicle to i. If no improvement is found, this procedure checks the cost variation of delaying the rental of the current vehicle to i. If some improvement in cost is found, the vehicle is rented at i and the solution is updated. Fig. 11 shows an example of this operator behaving at vertex 7 in the solution represented in Fig. 6. The first solution anticipates vehicle 3 to be rented at vertex 7 and the second solution delays vehicle 2 to be rented at vertex 7.

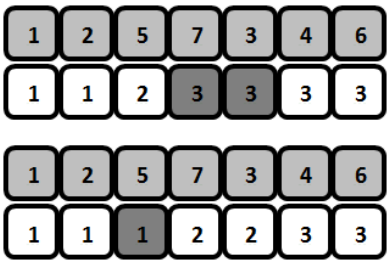

Fig. 11. Example of the operator λ_2.

The operator λ_3 aims at changing the position of a vertex in the hypothesis. For each vertex i in the theme, this procedure removes i from the route, removing its respective arcs and adding a new arc between the previous and next vertices. Then, i is reinserted after the vertex in the tour that causes the smallest increase in cost using the same vehicle as the new previous vertex. This operator does not affect vertices where vehicles are rented. Fig. 12 shows an example of this operator changing the position of vertex 7 in the solution represented in Fig. 6. The vehicle used to traverse vertex 7 is the same used at vertex 1, now previous vertex of 7.

Fig. 12. Example of the operator λ_3.

The operator λ_4 reverses an interval of the hypothesis. This procedure takes each pair of vertices (a, b) in the theme such that the path between a and b is traversed by a single vehicle and checks the cost variation of reversing the direction of this path. If some improvement in cost is found, the direction of the path is reversed and the solution is updated. Fig. 13 shows an example of this operator reversing the path between vertices 3 and 5 in the solution represented in Fig. 6.

Fig. 13. Example of the operator λ_4.

The operator λ_5 exchanges the position of two different intervals of the hypothesis. This procedure takes each two pairs of vertices (a, b) and (c, d) in the theme such that the path between a and b and the path between c and d are traversed by a single vehicle and checks the cost variation of exchanging the position of those paths. If some improvement in cost is found, the positions of the paths are exchanged and the solution is updated. Fig. 14 shows an example of this operator. The first solution is the hypothesis and the second solution is the result of exchanging the path between vertices 1 and 2 and the path between 7 and 3.

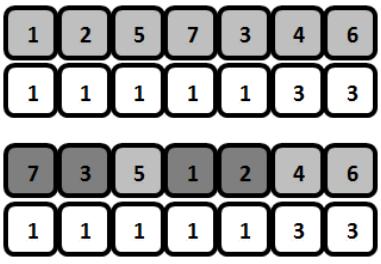

Fig. 14. Example of the operator λ_5.

F. Literature Update

The procedure to update literature is executed as follows. For each vertex i in the theme, this procedure checks if the previous vertex of i in the hypothesis, a, is already in L_i. If it is not, the procedure inserts a in L_i and removes the vertex in L_i that was updated by the highest cost solution. An auxiliary data structure is used to store the costs of the solutions that updated the vertices.

G. Stopping Criterion

The algorithm is terminated when the best solution of the population is not improved for 30 consecutive iterations, indicating convergence. To avoid loss of diversity, only 30% of the solutions are allowed to be identical, preventing new similar solutions to be added to the population in the update step.

IV. Computational Experiments

The scientific algorithm was implemented in C++ using GCC compiler. The machine used in the experiments is equipped with processor Intel Core i5 2.6 GHz, 4 Gb of RAM and Microsoft Windows operating system. The test was performed on twenty non-Euclidian instances proposed in Asconavieta [2] (available in http://www.dimap.ufrn.br/lae/en/projects/CaRS.php). Thirty independent runs of the algorithm were performed for each test case. The scientific algorithm was implemented in C++ using GCC compiler. The machine used in the experiments is equipped with processor Intel Core i5 2.6 GHz, 4 Gb of RAM and Microsoft Windows operating system. The test was performed on twenty non-Euclidian instances proposed in Asconavieta [2] (available in http://www.dimap.ufrn.br/lae/ en/projects/CaRS.php). Thirty independent runs of the algorithm were performed for each test case.

The parameters of this scientific algorithm were set on a preliminary experiment and are: the population size = 100, the size of the vertex lists in the literature = 10 the number of consecutive iterations without improving the best solution in the stopping criterion = 30 and the number of vertices in the theme as defined in item C of Section III.

Fig. 15 and Fig. 16 illustrate the impact of the size of the theme in solution quality and runtime, respectively, of the scientific algorithm in the BrasilNE50n instance, considering mean values on thirty runs. A similar behavior was observed on the other instances. A theme containing n vertices (size 100%) is equivalent to not limiting the algorithmic search and

may be interpreted as the absence of the concept of theme in the scientific algorithm.

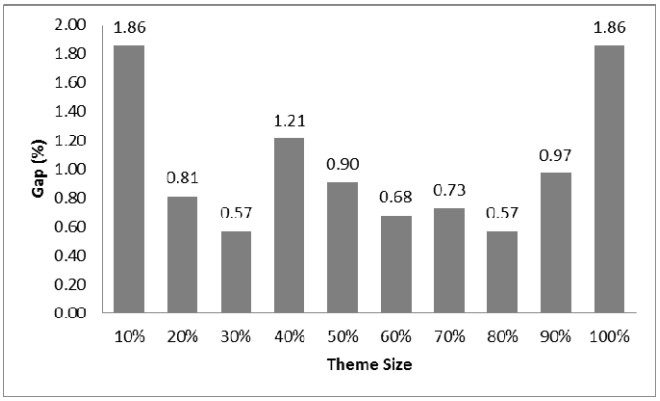

Fig. 15. Variation of the gap relative to the size of the theme in the instance BrasilNE50n.

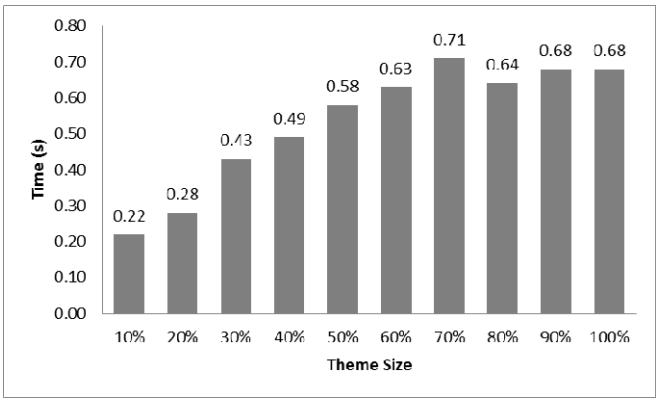

Fig. 16. Variation of the time relative to the size of the theme in the instance BrasilNE50n.

The study reveals that limiting the search of the scientific algorithm to a subset of the vertices of the problem allows finding better solutions when compared to the search without theme restraint. In Fig. 15, it is noted that the worst solutions are found with themes containing 10% or 100% of the vertices of the problem. In the former, the search is limited to a very small region in the solution space, failing to achieve enough diversification. In the latter, the search can explore the entire solution space, failing to achieve enough intensification. The best solutions are found with themes containing 30% or 80% of the vertices of the problem. These values represent the near optimal behavior of the scientific algorithm regarding intensification and diversification in the set of instances studied. A theme of size 30% leads to better solutions when the instance has up to fifty vertices and a theme of size 80% is more efficient when the instance has more than fifty vertices.

Fig. 16 shows that, surprisingly, the highest runtime was found with theme containing 70% of the vertices of the problem. This behavior is easily explained by the stopping criterion defined for this scientific algorithm: convergence of solutions. The execution time of each operator of the scientific algorithm is proportional to the size of the subject. However, certain levels of intensification and diversification of the search can make the solutions converge more or less rapidly. This can affect the number of times each operator will be executed by the algorithm.

Table I shows the results of the scientific algorithm. The columns show: the instance name, in which the number represents the number of vertices in the graph, the best known solution in the literature, the number of times the scientific algorithm found its best solution, the best, worst and average solution found by the scientific algorithm.

TABLE I. RESULTS OF THE SCIENTIFIC ALGORITHM TO SOLVE THE CARS.

Instance	Best	Hits	Min	Max	Av
BrasilRJ14n	167	30	167	167	167.00
BrasilRN16n	188	30	188	188	188.00
BrasilPR25n	226	30	226	226	226.00
BrasilAM26n	202	6	202	203	202.80
BrasilMG30n	271	9	271	276	272.50
BrasilSP32n	254	5	254	263	257.10
BrasilRS32n	269	21	269	270	269.30
BrasilCO40n	576	1	**574**	582	576.87
BrasilNO45n	551	1	**543**	556	**550.97**
BrasilNE50n	619	1	**609**	622	**614.73**
Londrina100n	1189	1	**1166**	**1184**	**1176.13**
Osasco100n	993	1	**975**	**989**	**982.93**
Aracaju200n	1966	1	**1910**	**1927**	**1918.40**
Teresina200n	1423	1	**1381**	**1395**	**1389.40**
Curitiba300n	2240	1	**2150**	**2165**	**2159.63**
berlin52nA	1328	1	**1308**	1330	**1322.20**
ch130n	1706	1	**1671**	**1693**	**1682.83**
d198n	3207	1	**3140**	**3171**	**3159.57**
kroB150n	2983	1	**2931**	**2958**	**2948.37**
rd100nB	1421	1	**1377**	**1398**	**1390.83**

The results show that the proposed algorithm found thirteen new best results for the investigated set of instances. Columns *Max* and *Av* in Table I show that the proposed algorithm found, respectively, nine and twelve worst and average results better than the previous best results.

TABLE II. COMPARISON: STATE OF THE ART AND THE SCIENTIFIC ALGORITHM.

Instance	Transgenetic Algorithms		Scientific Algorithms	
	Gap (%)	Time (s)	Gap (%)	Time (s)
BrasilRJ14n	0.00	1.70	0.00	0.17
BrasilRN16n	0.00	3.40	0.00	0.21
BrasilPR25n	0.88	13.60	0.00	0.55
BrasilAM26n	0.49	13.60	0.40	0.52
BrasilMG30n	2.58	27.20	0.55	0.98
BrasilSP32n	1.96	33.15	1.22	1.12
BrasilRS32n	0.74	34.00	0.11	1.16
BrasilCO40n	0.86	71.40	0.15	2.26
BrasilNO45n	1.27	88.40	-0.01	2.98
BrasilNE50n	1.61	124.95	-0.69	3.13
Londrina100n	1.09	940.95	-1.08	20.26
Osasco100n	1.30	849.15	-1.01	20.64
Aracaju200n	1.22	6246.65	-2.42	68.13
Teresina200n	3.09	7551.40	-2.36	88.44
Curitiba300n	1.83	31782.35	-3.59	346.70
berlin52nA	1.73	153.85	-0.44	6.94
ch130n	1.87	2406.35	-1.36	42.02
d198n	1.43	10194.05	-1.48	91.97
kroB150n	1.24	3801.20	-1.16	50.61
rd100nB	1.47	1063.35	-2.12	21.09

Table II presents a comparison with the transgenetic algorithm proposed in [3]. The transgenetic algorithm outperformed the GRASP, VND and memetic algorithms presented in [2]. Column *Gap* shows the percent deviation from the best results presented in [3] of the average of the solutions found by both algorithms in their independent executions. Column *Time* presents the average processing time, in seconds, spent by each algorithm.

Positive values in column *Gap* indicate that the average is over the previous best result, while a negative value indicates an average below that value. The computational times of the transgenetic algorithm are converted to the equivalent time in the platform of the scientific algorithm. This conversion is performed finding the processing factors of both platforms and multiplying the ratio of those values to convert the processing times.

The results show clear superiority of the scientific algorithm to the state of the art. The scientific algorithm has a processing time from ten to one hundred times faster than the transgenetic algorithm, finding equal or better solutions on all instances.

V. Conclusion

This paper proposed a new approach to solve optimization problems. The proposed algorithm is inspired in the process of knowledge evolution by means of scientific research, characterized as an evolutionary algorithm. The viability of the scientific algorithm is verified by computational experiments on CaRS, comparing the results with the state of the art.

The scientific algorithms allow focusing the computational effort on a small set of variables of the problem, reducing the computational cost. This enables the use of more robust techniques for solving the problem. This feature, added to the good results of computational experiments, supports the hypothesis that the metaphor of scientific research can be used to develop algorithms to solve optimization problems efficiently.

In future works, this approach will be extended to different problems including multi-objective, to investigate the robustness of the scientific algorithms.

References

[1] M. D. Marconi and E. M. Lakatos. *Fundamentos da Metodologia Científica*. Atlas, 2010.

[2] M. C. Goldbarg, P. H. S. Asconavieta and E. F. G. Goldbarg. "A memetic algorithm for the car renter problem," in *Evolutionary Algorithms*. Intech, 2011, pp. 309-326.

[3] P. H. S. Asconavieta, M. C. Goldbarg and E. F. G. Goldbarg. "Evolutionary algorithm for the car renter salesman," in *Proceedings of the IEEE CEC Congress on Evolutionary Computation*, vol. 1, 2011, pp. 593-600.

[4] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys. *The Traveling Salesman Problem: A guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[5] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

[6] G. Gutin and A. P. Punnen. *Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[7] Z. Michalewicz and D. B. Fogel. *How to Solve it: Modern Heuristics*. Springer, 2000.

[8] J. H. Holland. "Genetic algorithms and the optimal allocations of trial," in *SIAM Journal on Computing*, vol. 2, 1973, pp. 99-105.

[9] N. J. Radcliffe and P. D. Surry. "Formal memetic algorithms," in *Springer-Verlag LNCS*, vol. 865. 1994, pp. 1-16.

[10] R. G. Reynolds. "An introduction to cultural algorithms," in *Proceedings of the Third Annual Conference on Evolutionary Programming*, 1994, pp. 131-139.