

Study Cases on Initial Solution and Searching for Tabu Search and Threshold Accepting Algorithms on Bin-Packing Problem

Vanesa Landero Nájera[1(✉)], Ortega Joaquín Pérez[2], Laura Cruz Reyes[3], Carlos Rodríguez Orta[1], and Carlos Collazos Morales[4]

[1] Universidad Politécnica de Apodaca, Nuevo León, México
vanesa.landero@upapnl.edu.mx
[2] Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Departamento de Ciencias Computacionales, AP 5-164, Cuernavaca 62490, México
jperez@cenidet.edu.mx
[3] División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero (ITCM), Cd. Madero, México
laura.cr@cdmadero.tecnm.mx
[4] Universidad Manuela Beltrán, Bogotá, Colombia

Abstract. The scientific community has identified four principal parts on approximation algorithm's logical structure (tuning parameter, generating initial solution, searching and generating neighbor solutions, stopping algorithm execution) and has performed improvements for solving problems such as sorting, forecasting, classification, clustering, constraint satisfaction, decision, optimization. A reviewing of state of art for algorithms Tabu Search (TS) and Threshold Accepting (TA) indicates that in most cases it has performed improvement on logical part for tuning parameter. A reflection about it shows that up to now there has not been an analysis about the other logical parts: initial solution and searching. Study cases for TS and TA over one dimension Bin Packing problem were configured for discovering knowledge of these parts on algorithm performance in terms of quality and time. The results for all configurations indicated that generate a deterministic initial solution and neighbor solutions from this using one method for searching permit to algorithms TA and TS perform better on very short time (reaching solutions with best, same quality and worst with a very minimal difference) than intensify the searching with several methods. It would be of interest if this found knowledge is like other algorithms for other problems, which could contribute to building self-adaptive algorithms that do not waste effort to give the best solution to complex problems.

1 Introduction and General Review

The scientific community of several areas of knowledge (computational complexity theory, graph theory, data mining, artificial intelligence, machine learning, combinatorial optimization, operations research) for solving problems such as decision, optimization, forecasting, classification, clustering, sorting, has identified four principal parts from the

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15648, pp. 71–87, 2025.
https://doi.org/10.1007/978-3-031-97000-9_5

logical internal structure of the solution algorithms, for developing improvements into them and giving the best solution. These logical parts are related to the methodology for: parameter tuning (PT) as temperature, size of tabu list, size of population, etc.; initial solution generating (IS); neighbor solutions searching (SM); algorithm execution stopping (SC). Table 1 presents a revision of literature consulted for Threshold Accepting (TA) and Tabu Search (TS) algorithms [1–11], which have become applied for solving some problems from some mentioned knowledge areas. No work was found that improved the logical part (SC) for these algorithms, but another work, for the k-means clustering algorithm is quoted for exemplifying this logical part [12].

Table 1. Reviewed Specialized Literature

Work	Algorithm Internal Logical Structure			
	PT	IS	SM	SC
[1]	✓			
[2]	✓			
[3]	✓		✓	
[4]			✓	
[5]	✓			
[6]	✓		✓	
[7]	✓	✓		
[8]	✓		✓	
[9]			✓	
[10]	✓			
[11]	✓		✓	
This paper		✓	✓	

As it can be observed in Table 1, the majority of the related works consulted have worked for tuning control parameters [1–3, 5–8, 10, 11]; some other also it is included work for searching in some way into the problem space, generating neighbor solutions and improving the algorithms performance [3, 4, 6, 8, 9, 11]. However, these related works develop improvements in some of these logical parts without analyzing the effects of different methodologies applied to logical parts and understand better the behavior and performance of algorithms. Only a few works perform such study in parameter tuning [2, 5]. In the case of TA algorithm for satisfiability problem (SAT) [2], the proposed geometric cooling function has the best execution time than exponential and logarithmic cooling proposed functions with a minimal difference (smaller than one percent) and similar results were in the case of best solution quality. In the case of TS algorithm for problems quadratic assignment (QAP) and maximum clique problem (MCP) [5], the TS performance is strongly dependent on the characteristics of the instances tackled for

setting the best parameter configuration. It could be interpreted that the performance of both algorithms for those problems does not have a great improvement in the management of one or another proposed methodology for parameter tuning or there is a dependency on the features of problem instance in hand.

A reflection on the reviewed specialized literature shows that up to now there is no work that performs a thorough experimental analysis on the other logical parts, for example: generating of initial solution and performing a searching into the problem space. The analysis of these logical parts could allow the discovery of guidelines to direct the algorithm to not spend so much time in finding the best solution.

The algorithms have been seen as a partial black-box and therefore, it is necessary and important to open completely the box and identify logical parts relevant to the connection of the algorithm and problem during the solution process [13]. As well as identifying the algorithm domination region and delving into its performance, analyzing not only solution quality but also the execution time [14]. The above is due to scenarios existing with a minimal significant difference between solution qualities from two algorithms a_1 and a_2 in comparison but with a very big significant difference between solution times. It would be wrong to affirm that one algorithm outperforms to another only because solution quality is lightly better than another is.

Therefore, it would be interesting to analyze algorithm performance, considering these other logical parts and domination region in terms of solution quality and time, to discover relevant knowledge which could permit the designing of self-adaptive algorithms that do not waste effort to give the best solution to complex problems.

To be able to give small, but relevant steps according to all mentioned above, firstly, starting for a specific domain; in this paper, the focus and main contributions are next. In Sect. 2, a framework is proposed to perform an analysis on the algorithm logical parts in different scenarios of methodologies for: generating the initial solution; generating neighbor solutions and searching into the problem space. Therefore, scenarios and cases of study were configured on: the one-dimension Bin Packing problem, considering important repositories of problem instances [15, 16], which are known by scientific community; the Threshold Accepting and Tabu Search algorithms, which have become successful for solving hard combinatorial optimization problems [6, 7, 17]. In Sect. 3, an analysis of algorithms performance results is performed in terms of domination regions, considering solution quality and execution time for studies cases from configured scenarios. Graphical and statistical analyses were applied. In Sect. 4, a comparison and discussion of results is performed. Finally, in Sect. 5, conclusions about performed work are presented, as well as research future works.

2 Framework

2.1 Formal Nomenclature

$\boldsymbol{P} = \{x_1, x_2, \ldots, x_m\}$ a set instances of problem domain **P**, the space for analysis.
$\boldsymbol{A} = \{a_1, a_2, \ldots, a_n\}$ a set of n algorithms.
$\boldsymbol{Y} =$ the performance space, it represents the mapping of each algorithm to a set of performance metrics (quality and time).

2.2 Problem (Set *P*) and Algorithms

The framework consists of study cases for analyzing the algorithm logical parts in different scenarios of methodologies for: generating the initial solution; generating neighbor solutions and searching into the problem space. The set $\boldsymbol{P}$ refers to the One Dimension Bin-Packing (BPP) problem and it contains 324 problem instances, which were selected randomly from repositories: Beasley's OR-Library [15], the Operational Research Library [16]. The algorithms for solving these problem instances are Tabu Search (TS) and Threshold Accepting (TA) algorithms. The size of Tabu list was fixed as 7 [18] for TS algorithm and the initial temperature was fixed 1 [19] for TA algorithm. The stop criterion happens after 4000 iterations (divergence) in both algorithms.

2.3 Algorithms Variants and Scenarios (Set *A*)

For specific purposes in this paper, the internal logical structure of algorithms Tabu Search algorithm and Threshold Accepting algorithm will be analyzed focusing on methodology for generating the initial solution, which can be generated randomly (R), commonly used by scientific community, or by means deterministic procedure (D), see Algorithms 1, 2.

Algorithm 1. Pseudocode for Generating a Random Initial Solution

	GenerateInitialSolution_Random
1	**Begin**
2	While objects exist to accommodate
3	Insert randomly an object into a new container
4	Do
5	Obtain a list of candidate objects that can enter to
6	container
7	If there are candidates, Then
8	Select randomly an object
9	Insert to container
10	While objects exist that can be entered into container
11	**End**

These two ways for generating the initial solution were analysing in two different scenarios. In the first scenario (see Table 2), the methodology for generating the neighborhoods, from the initial solution and after from generated solutions and so on, consisted in use one method (O). Unlike the second scenario (see Table 3), where it consisted in use several methods (M). These methods O and M were proposed in [20] and described in Algorithms 3, 4. The set $\boldsymbol{A}$ of algorithms for each study case of scenarios 1 and 2 is shown in Tables 2 and 3 respectively. More specifically, $\boldsymbol{A_1} = \{a_1, a_2\}, \boldsymbol{A_2} = \{a_3, a_4\}$ (scenario 1) and $\boldsymbol{A_1} = \{a_1, a_2\}, \boldsymbol{A_2} = \{a_3, a_4\}$ (scenario 2).

Algorithm 2. Pseudocode for Generating a Deterministic Initial Solution

```
GenerateInitialSolution_Deterministic
Begin
While objects exist to accommodate
    Insert randomly an object into a new container
    Do
        Obtain a list of candidate objects that can enter to
        container
        If there are candidates, Then
            Calculate the probability distribution of candidates
            Choice randomly a number r between (0, 1)
            For each candidate object
                If r ≤ to the object probability, Then
                        Choice the object
                        break
            Insert selected object to container
    While exists objects can be enter into container
End
```

Table 2. Scenario 1

set $\boldsymbol{A}$	variants	Initial Solution Methodology		Neighborhood Methodology		Algorithm
		R	D	O	M	
$\boldsymbol{A1}$	a_1	✓		✓		Threshold Accepting
	a_2		✓	✓		ThresholdAccepting
$\boldsymbol{A_2}$	a_3	✓		✓		Tabu Search
	a_4		✓	✓		Tabu Search

Table 3. Scenario 2

set $\boldsymbol{A}$	variants	Initial Solution Methodology		Neighborhood Methodology		Algorithm
		R	D	O	M	
$\boldsymbol{A_1}$	a_1	✓			✓	Threshold Accepting
	a_2		✓		✓	Threshold Accepting
$\boldsymbol{A_2}$	a_3	✓			✓	Tabu Search
	a_4		✓		✓	Tabu Search

Algorithm 3. Pseudocode for Generating a Neighbour Solution with One Method

```
GenerateNeighbourSolution_OneMethod
Begin
Choice randomly a source container and a destination container
Intent generates a neighbor solution with operator (1, 0)
If Neighbor solution could not be generated, Then
    Exchange source and destination
    Intent generates a neighbor solution with operator (1, 0)
    If Neighbor solution could not be generated, Then
        An infeasible solution is accounted
End
```

Algorithm 4. Pseudocode for Generating a Neighbour Solution with Several Methods

```
GenerateNeighbourSolution_SeveralMethods
Begin
Choice randomly a movement of: operator (1, 0), operator (1, 1), operator
(1, 2) and Operator (2, 2)
Choice randomly a source container and a destination container
For four kinds of movements
    //It begins with a selected movement
    Generate a neighbor solution
    If Neighbor solution could be generated, Then
        Break
If Neighbor solution could not be generated, Then
    Exchange source and destination
    Intent generates a neighbor solution with operator (1, 0)
    If Neighbor solution could not be generated, Then
        Intent generates a neighbor solution with operator (1, 2)
        If Neighbor solution could not be generated, Then
            An infeasible solution is accounted
End
```

2.4 Characterizing Algorithm Performance (Set *Y*) and Studies Cases

The algorithms in sets $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ for scenario 1 and scenario 2 were executed to solve the problem instances of set $\boldsymbol{P}$. After execution of each problem instance (1, 2,…,*m*), the algorithm performance is measured by performance metrics *time* and *quality*, described in expressions 1 and 2 respectively. The evaluations number of fitness function for feasible and infeasible solutions is considered for *time*. The *quality* is the ratio between the best solution found by algorithm Q_f (final number of containers) and theoretical solution Q_t (sum of object sizes divided by container capacity *c*), the lower the value, the better the quality.

$$time = feasibles + infeasibles \tag{1}$$

$$quality = \frac{Q_f}{Q_t} \tag{2}$$

$$Q_t = \frac{\sum_{i=1}^{k} s_i}{c} \tag{3}$$

The sets $\boldsymbol{Y}$ for scenario 1 ($\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$) and scenario 2 ($\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$) are being built, mapping these performance features to each algorithm with the specific order as Expression 4; where $n = 2$, is the total of algorithms executed for each set. The values of $quality_{11}$, $time_{11}$ meaning the performance of algorithm 1 for problem instance 1; the values of $quality_{12}$, $time_{12}$ meaning the performance of algorithm 2 for problem instance 1; the values of $quality_{m1}$, $time_{m1}$ meaning the performance of algorithm 1 for problem instance m; the values of $quality_{m2}$, $time_{m2}$ meaning the performance of algorithm 2 for problem instance m, and so on. Remembering, $m = 324$ problem instances of set $\boldsymbol{P}$.

$$Y = \left\{ \begin{array}{c} \left\{\left\{quality_{11}, time_{11}\right\}, \left\{quality_{12}, time_{12}\right\}, \ldots, \left\{quality_{1n}, time_{1n}\right\}\right\} \\ . \\ . \\ . \\ \left\{\left\{quality_{m1}, time_{m1}\right\}, \left\{quality_{m2}, time_{m2}\right\}, \ldots, \left\{quality_{mn}, time_{mn}\right\}\right\} \end{array} \right\} \tag{4}$$

The function $s(a_q, i)$ considers information of algorithm a_q from set $\boldsymbol{Y}$ ($\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ - scenario 1, $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ - scenario 2) for one problem instance i (consider a_q as algorithm that will be object of study). Expression 5 describes the function $s(a_q, i)$. This function returns the performance scope of algorithm a_q compared to other algorithms for problem instance i. The set $\boldsymbol{S}a_q$ contains the values of scope of algorithm a_q for all problem instances (see Expression 6). A value 1 means that algorithm a_q was the best to solve the problem instance in turn, in terms of performance metrics, *quality* and *time*; otherwise, it has a value 0.

$$s\left(a_q, i\right) = \left\{ \begin{array}{c} 1, if\,(Y\left(quality_{iq}\right) = Y\left(quality_{i\propto}\right) and Y\left(time_{iq}\right) > Y\left(time_{i\propto}\right)) \\ or \\ Y\left(quality_{iq}\right) > Y\left(quality_{i\propto}\right), \forall \propto \neq t; quality_{iq} \in Y; quality_{i\propto} \in Y; \\ 0, otherwise. \end{array} \right\} \tag{5}$$

$$Saq = \left\{s(a_q, 1), s(a_q, 2), \ldots, s(a_q, m)\right\} \tag{6}$$

The function s applied to an algorithm a_q permits obtain the algorithm total scope TSa_q, by means Expression 7, so too, the domination region of algorithm, which is described by the sets $\boldsymbol{C}$ and $\boldsymbol{W}$.

$$TSa_q = \sum_{i=1}^{m} s(a_q, i) \tag{7}$$

$\boldsymbol{C} = \{C_1, C_2, \ldots, C_n\}$ a partition of $\boldsymbol{P}$, where $|\boldsymbol{A}|=|\boldsymbol{C}|$, and.
$\boldsymbol{W} = \{(a_q \in \boldsymbol{A}, C_q \in \boldsymbol{C}) \mid s(a_q, x) = 1$ compared to $\alpha, \forall\, \alpha \in (\boldsymbol{A} - \{a_q\}), \forall\, x \in C_q\}$, is a set of domination regions, ordered pairs (a_q, C_q), where each dominant algorithm $a_q \in \boldsymbol{A}$ is associated with one element C_q of partition $\boldsymbol{C}$, because this gives the best performance scope to partition C_q.

Each case of study (see Table 4) for the scenario 1 (one method for generating neighbor solutions), consists of comparing two variants of algorithms Thresholding and Tabu Search belonging to sets $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, which only are different by method for generating the initial solution (random, deterministic), so too, the total scope of performance of variants is calculated and also the sets of domination regions are analysed with more detail. The same procedure for cases of study of scenario 2 (several methods for generating neighbor solutions).

Table 4. Study cases

Scenario	Study case	set $\boldsymbol{A}$	Algorithm	Variants Comparison	Algorithms Scope Total	Domination Regions (set $\boldsymbol{W}$)
1	1	A_1	Threshold	a_1 vs. a_2	TSa_1, TSa_2	(a_1, C_1), (a_2, C_2)
	2	A_2	Tabu Search	a_3 vs. a_4	TSa_3, TSa_4	(a_3, C_3), (a_4, C_4)
2	3	A_1	Threshold	a_1 vs. a_2	TSa_1, TSa_2	(a_1, C_1), (a_2, C_2)
	4	A_2	Tabu Search	a_3 vs. a_4	TSa_3, TSa_4	(a_3, C_3), (a_4, C_4)

3 Results Analysis

3.1 Graphical and Statistical Analysis

Scenario 1. Analysing initial solution methodology on a searching with one method
Case of study 1 – Threshold Accepting (TA)

The Fig. 1 and Fig. 2 show the results of *quality* and *time*, at lower values, better *quality* and *time*, for the variants of Threshold Accepting algorithm (TA): a_1 performs the methodology of algorithm 1 for generating a random initial solution and a_2 performs the methodology of algorithm 2 for generating a deterministic initial solution. In this scenario 1, from the generated initial solution, whatever random or deterministic, the variants generate neighbour solutions with one method (see Algorithm 3) and perform the searching in this way. As it can see in the figures, there are big differences of time where variant a_2 seems to be better than a_1. So too, there are small differences of solution quality, where it is not possible to distinguish which of the two variants is better. The total scope of variants in terms of domination by means Expression 7, $TSa_1 = 147$, $TSa_2 = 177$; thereafter determining their total scope percentage $PTSa_1 = 147/324 = 45.37\%$ and $PTSa_2 = 177/324 = 54.62\%$.

Analysing the set $\boldsymbol{W}$, specifically the regions of domination of variants a_1, (a_1, C_1) and a_2, (a_2, C_2) in more detail. In the partition C_1, a_1 wins 147 in *quality*, the *time*

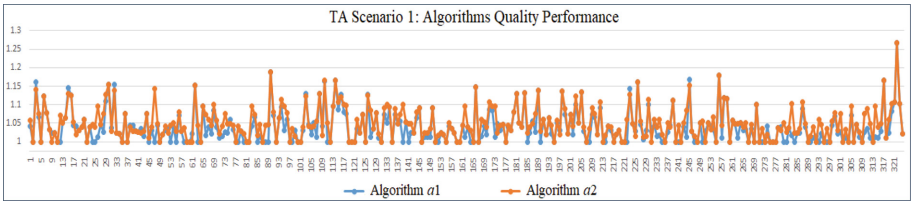

Fig. 1. Variants (algorithms a_1 and a_2) Quality Performance in Scenario 1 - Case of Study 1 for Threshold Accepting Algorithm

in these instances is bigger than algorithm a_2. The differences in *time* are too big (see Fig. 2), therefore it is not necessary to apply a statistical test. However, it is necessary to verify the values of *quality* metric (see Fig. 1), specifically when algorithm a_1 has the best *quality* C_1 (147 instances). The values of performance measures *quality* and *time* for each one of variants of all studies cases do not assume a normal distribution. Thus, a nonparametric statistical test of two independent samples is applied (the two sample two-side wilcoxon signed rank test) for significance levels 95% and 99%. The Dataplot statistical software (www.itl.nist.gov) was used for all tests.

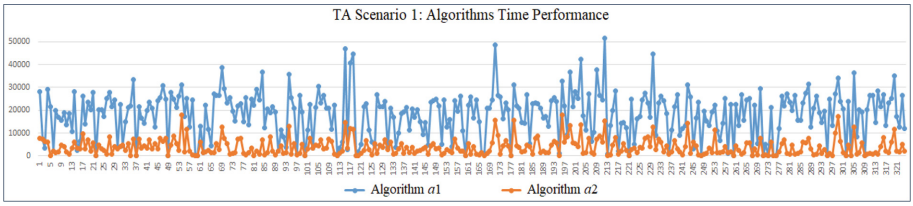

Fig. 2. Variants (algorithms a_1 and a_2) Time Performance in Scenario 1 - Case of Study 1 for Threshold Accepting Algorithm

The test statistic was -10.3383 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between qualities. The algorithm a_1 is better than a_2 in quality for these instances C_1 (147 instances); however, its time is greater than a_2.

Analyzing the partition C_2 (177 instances) of algorithm a_2, all instances have the best *time* (differences are too big), 33 instances have the best *quality* and 144 the same *quality* as algorithm a_1, but a_2 wins due its *time*.

Case of study 2 – Tabu Search Algorithm (TS).

The Fig. 3 and Fig. 4 show the results of *quality* and *time* for the variants of Tabu Search algorithm (TS): a_3 performs the methodology of algorithm 1 for generating a random initial solution and a_4 performs the methodology of algorithm 2 for generating a deterministic initial solution. In this scenario 1, from the generated initial solution, whatever random or deterministic, the variants generate neighbour solutions with one method (see Algorithm 3) and perform the searching in this way.

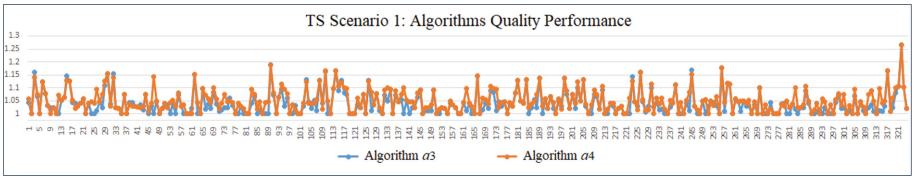

Fig. 3. Variants (algorithms a_3 and a_4) Quality Performance in Scenario 1 - Case of Study 2 for Tabu Search Algorithm

As it can see, there are big differences of time where variant a_4 seems to be better than a_3. So too, there are small differences of solution quality, where it is not possible to distinguish which of the two variants is better. The total scope of is $TSa_3 = 164$, $TSa_4 = 160$ and their total scope percentage $PTSa_3 = 164/324 = 50.62\%$, $PTSa_4 = 160/324 = 49.38\%$.

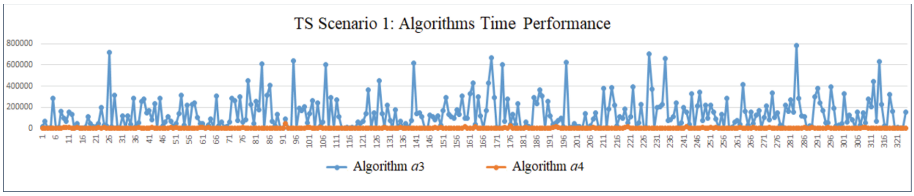

Fig. 4. Variants (algorithms a_3 and a_4) Time Performance in Scenario 1 - Case of Study 2 for Tabu Search Algorithm

Analysing the set $\boldsymbol{W}$, the regions of domination of variants a_3, (a_3, C_3) and a_4, (a_4, C_4) in more detail, a_3 wins 164 in *quality*, partition C_3 (164 instances), the *time* in these instances is bigger than algorithm a_4. The differences in *time* are too big (Fig. 4), therefore it is not necessary to apply a statistical test for a second analysis. However, it is necessary to verify the values of *quality* metric (Fig. 3), specifically when algorithm a_3 has the best *quality* C_3 (164 instances). The test statistic was -11.1079 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between qualities. Analysing the partition C_4 (160 instances) of algorithm a_4, all instances have the best *time* (differences too big), 67 instances have the best *quality* and 93 the same *quality* as algorithm a_3, but a_4 wins due its *time*.

3.2 Graphical and Statistical Analysis

Scenario 2. Analysing the initial solution methodology on a searching with several methods

Case of study 3 – Threshold Accepting Algorithm (TA)

The Fig. 5 and 6 show the results of *quality* and *time* for the variants of Threshold Accepting algorithm (TA): a_1 performs the methodology of algorithm 1 for generating a random initial solution and a_2 performs the methodology of algorithm 2 for generating a deterministic initial solution. In this scenario 2, from the generated initial solution, whatever random or deterministic, the variants generate neighbour solutions with several methods (see Algorithm 4) and perform the searching in this way. As it can see, it is not possible to distinguish which of the two variants is better. The total scope of variants is $TSa_1 = 65$, $TSa_2 = 259$, and their total scope percentage $PTSa_1 = 65/324 = 20.06\%$, $PTSa_2 = 259/324 = 79.94\%$. Analysing the set $\boldsymbol{W}$, the regions of domination of variants a_1, (a_1, C_1) and a_2, (a_2, C_2) in more detail. In the partition C_1 (65 instances), a_1 wins 65 in *quality*. The test statistic was -6.68 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between qualities. The algorithm a_1 is better than a_2 in *quality* for those instances C_1 (65); however, its time is greater than a_2.

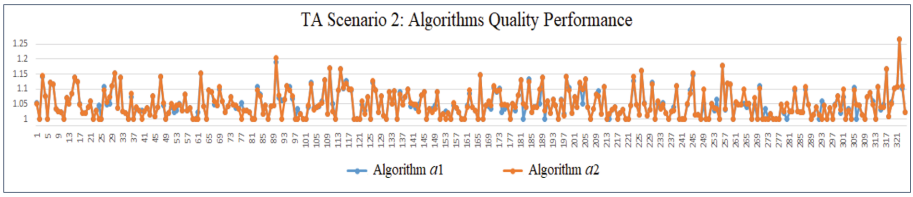

Fig. 5. Variants (algorithms a_1 and a_2) Quality Performance in Scenario 2 - Case of Study 3 for Threshold Accepting Algorithm

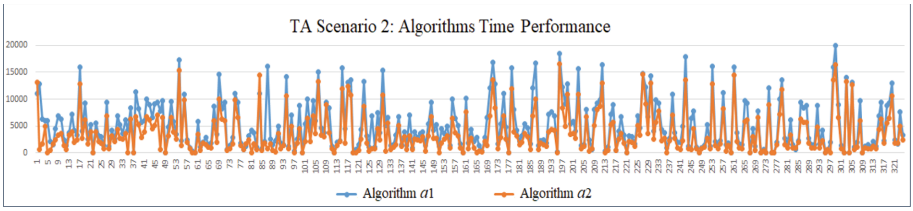

Fig. 6. Variants (algorithms a_1 and a_2) Time Performance in Scenario 2 - Case of Study 3 for Threshold Accepting Algorithm

Analysing the partition C_2 (259 instances) of algorithm a_2, all instances have the best *time*, 48 instances have the best *quality* and 211 the same *quality* as algorithm a_1, where a_2 wins due its *time*. However, it is necessary to verify these values of *time* metric. The test statistic was 11.9530 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between times.

Case of study 4 – Tabu Search Algorithm (TS)

The Fig. 7 and Fig. 8 show the results of *quality* and *time* for the variants of Tabu Search algorithm (TS): a_3 performs the methodology of algorithm 1 for generating a random initial solution and a_4 performs the methodology of algorithm 2 for generating a deterministic initial solution. In this scenario 2, from the generated initial solution, whatever random or deterministic, the variants generate neighbour solutions with several methods (see Algorithm 4) and perform the searching in this way. As it can see, it is not possible to distinguish which of the two variants is better.

The total scope of variants is $TSa_3 = 132$, $TSa_4 = 192$ and their total scope percentage $PTSa_3 = 132/324 = 40.74\%$, $PTSa_4 = 192/324 = 59.26\%$. Analyzing the set $\boldsymbol{W}$, the regions of domination of variants a_3, (a_3, C_3) and a_4, (a_4, C_4) in more detail. In the partition C_3 (132 instances) of algorithm a_3, indicates a_3 wins 38 instances in *quality*; where the time for these instances is bigger that algorithm a_4. So too, the algorithm a_3 wins 94 instances in *time* when the *quality* is the same. In the case of *quality* (38), the test statistic was -5.3734 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between qualities. In the case of *time* (94), when the quality is the same, the test statistic was -8.4146 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between times.

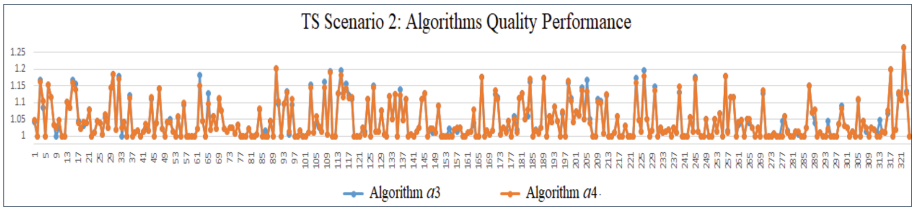

Fig. 7. Variants (algorithms a_3 and a_4) Quality Performance in Scenario 2 - Case of Study 4 for Tabu Search Algorithm

In the same context, but now, analyzing the region of domination of algorithm a_4 with more detail, (a_4, C_4), C_4 (192 instances), the algorithm a_4 wins in *time* 146 when the *quality* is the same and it wins 46 instances in *quality*, where in 20 instances has better *time* than a_3 and 26 instances has worse *time* than a_3. However, it is necessary to verify the values of *time* metric for these 146 instances. The test statistic was 10.482 and the critical values were 1.96, 2.5758 for significance levels 95% and 99%. The null hypothesis ($\mathbf{H}_0$: equal means) is rejected and it means that there is a significant difference between times.

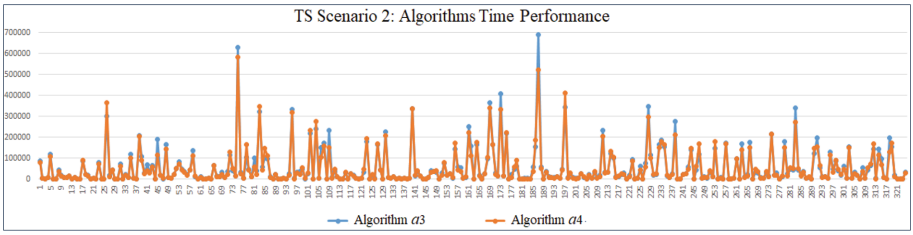

Fig. 8. Variants (algorithms a_3 and a_4) Time Performance in Scenario 2 - Case of Study 4 for Tabu Search Algorithm

4 Comparison and Discussion of Results

Table 5 shows case studies of scenarios 1 and 2: the domination region of algorithms in terms of performance total scope for the *Quality* and *Time* metrics; all values of test statistic and critical values from Wilcoxon statistical tests applied to differences between values *quality* and *time* from algorithm variants. The results of statistical test indicated that there is a significant difference between solutions qualities and times, the null hypothesis $\mathbf{H_0}$ is rejected in all studies cases from two scenarios.

Table 5. Domination Regions, Statistical Results and Observations

			Domination Region (set W)		Statistical Results*			Observations		
		variants	Quality	Time	Test Statistical *	Critical Value	Null Hypothesis $\mathbf{H_0}$ (means equal)	Same Quality		Worst Time
1	1 (TA)	a_1(147)	147*		-10.3383	1.96/2.58	Rejected			147$^+$
		a_2(177)		144$^+$				144		
			33*	33$^+$	5.0121	1.96/2.58	Rejected			
	2 (TS)	a_3(164)	164*		-11.1079	1.96/2.58	Rejected			164$^+$
		a_4(160)		93$^+$				93		
			67*	67$^+$	7.1151	1.96/2.58	Rejected			
2	3 (TA)	a_1(65)	65*		-6.68/6.8519	1.96/2.58	Rejected			65*
		a_2(259)		211*	11.9530	1.96/2.58	Rejected	211		
			48*	48*	6.03/5.24	1.96/2.58	Rejected			
	4 (TS)	a_3(132)		94*	-8.4146	1.96/2.58	Rejected	94		
			38*		-5.373/4.97	1.96/2.58	Rejected			38*
		a_4(146)	46*	20	5.9053	1.96/2.58	Rejected			26
				146*	10.482	1.96/2.58	Rejected	146		

+ The differences are too big (shown in Figs. 2, 4). There had not necessity to apply statistical test

The results are similar for the study cases 1 (Threshold Accepting algorithm - TA) and 2 (Tabu Search algorithm - TS) from scenario 1. Remembering, this scenario consists of generating an initial solution random (algorithm 1) or deterministic (algorithm 2) and from this generating neighbour solutions with one method (algorithm 3) and performs the searching in this way. Therefore, the results indicate that the domination region of variants of algorithms TA and TS that perform the methodology of algorithm 1 for generating a random initial solution (variants a_1 and a_3) have a great cost in *time*, their time is too big, for winning in *quality* to variants of algorithms TA and TS that perform the methodology of algorithm 2 for generating a deterministic initial solution (variants a_2 and a_4). It is to say, these variants (deterministic initial solution) finish faster than those variants a_1, a_3 (random initial solution), where their cost in *quality* (variants a_2 and a_4) is a mean difference 0.021 from 147 problem instances and 0.033 from 164 problem instances, respectively, which are too small. So too, the domination region of these variants (variants a_2 and a_4) always wins in *time* (time is too small) with the best or same *quality*.

The Fig. 9 shows scatter plots for study cases of scenarios 1 (Above) and 2 (Below). As it can see for scenario 1 (Above), the scatter plots help to reinforce what was mentioned before. It is to say, variants a_2 and a_4 (deterministic initial solution) in the scenario 1 seem to be a better alternative than variants a_1 and a_3 (random initial solution) when you are looking for the best *time* (time is too small) and can be reached in some cases with best *quality* or same *quality* or worst *quality* with a minimal difference.

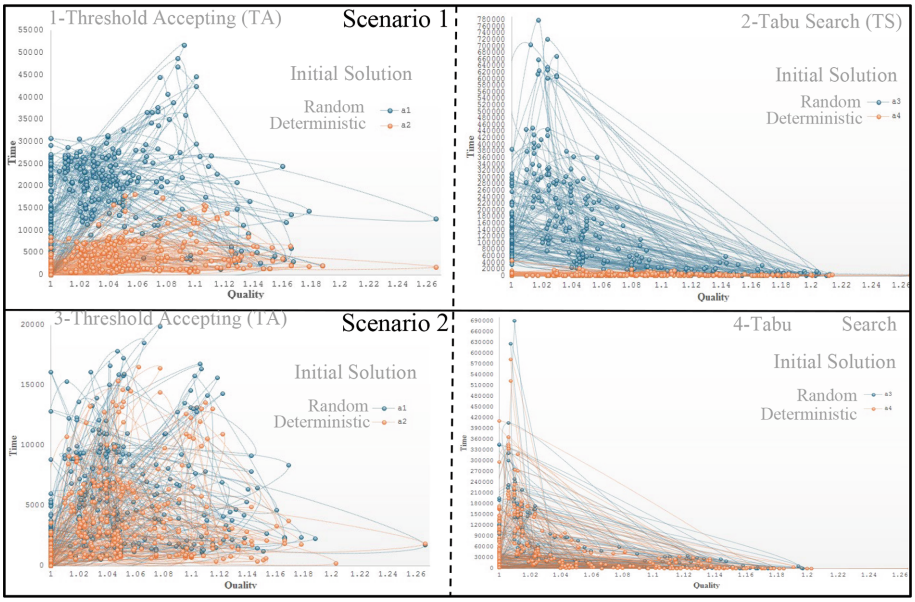

Fig. 9. Scatter plots for study cases 1,3 (TA) and 2,4 (TS) of scenarios 1-Searching with one method (Above) and 2-Searching with several methods (Below)

On the other hand, in the scenario 2, remembering that it consists of generating an initial solution random (algorithm 1) or deterministic (algorithm 2) and from this generating neighbour solutions with several methods (algorithm 4) and performs the searching in this way. The results in Table 5 of study case 3 indicate that variant a_2 (deterministic initial solution) of Threshold Accepting algorithm (TA) seem to be the better in time than variant a_1 (random initial solution) in all problem instances when it has better, same quality and worst quality with a very minimal difference 0.015 from 59 problem instances. So too, as it can see for this study case 3, the scatter plot (below left) of scenario 2 (Below) in Fig. 9 help to reinforce this result, which is some very similar for TA algorithm in scenario 1, with the difference of that it (scenario 1) has less source code in the program. Otherwise in the study case 4, the results in Table 5 indicate that variants a_3 (random initial solution) and a_4 (deterministic initial solution) of Tabu Search algorithm (TS) have a good time with better, same quality and worst quality. The scatter plot (below right) of Fig. 9 help to reinforce this, where there is no difference in *quality* and *time* between them.

5 Conclusions

There are four principal parts on logical internal structure of approximation algorithms: tuning parameter, generating initial solution, searching and generating neighbour solutions, stopping algorithm execution. The scientific community has performed improvements in some of these logical parts for solving problems as sorting, forecasting, classification, clustering, constraint satisfaction, decision, optimization. A reviewing of state of art indicates that in most cases it has performed improvement on logical part for tuning parameter. A reflection about it shows that up to now the algorithms have been seen as a partial black-box and it is necessary and important to open completely the box, analyse other logical parts relevant to the connection of the algorithm and problem during the solution process.

A framework is proposed to perform an analysis on the algorithm logical parts in different scenarios of methodologies for: generating the initial solution; generating neighbour solutions and searching into the problem space. Scenarios and cases of study were configured on: the one-dimension Bin Packing problem, considering important repositories of problem instances, which are known by scientific community; the Threshold Accepting and Tabu Search algorithms, which have become successful for solving hard combinatorial optimization problems. Graphical and statistical analysis were performed in terms of domination regions, considering solution quality and execution time. In general terms, the results indicated that an initial solution generated by deterministic methodology, either for Threshold Accepting algorithm or Tabu Search algorithm, is adjust better when from this, neighbour solutions are generated using one method for searching into the problem space. It is to say, a good initial seed could direct to a searching without intensify it (less source code in the program) and finish in much less time with solutions of better or same quality and in situations of worst quality but with a minimal difference, where the time is more important in a real situation than this minimal difference. As future work, continuing opening the black box of algorithms logical internal structure, it would be interesting to perform an analysis on initial solution

methodology in conjunction with the searching methodology for other problems and approximation algorithms. The results could give formal guidelines in the development of self-adapting algorithms that do not waste effort to give the best solution to complex problems, depending on specific needs in various real-life situations.

References

1. Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S.: A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Eur. J. Oper. Res. **152**(1), 148–158 (2004)
2. Martinez-Rios, F., Frausto-Solis, J.: An hybrid simulated annealing threshold accepting algorithm for satisfiability problems using dynamically cooling schemes. Electrical and Computer Engineering Series WSEAS, pp. 282–286 (2007)
3. Abdelaziz, A.Y., Mohamed, F.M., Mekhamer, S.F., Badr, M.A.L.: Distribution system reconfiguration using a modified Tabu Search algorithm. Electric Power Systems Res. **80**(8), 943–953 (2010)
4. Gnewuch, M., Wahlström, M., Winzen, C.: A new randomized algorithm to approximate the star discrepancy based on threshold accepting. SIAM J. Numer. Anal. **50**(2), 781–807 (2012)
5. Pellegrini, P., Mascia, F., Stützle, T., Birattari, M.: On the sensitivity of reactive tabu search to its meta-parameters. Soft. Comput. **18**, 2177–2190 (2014)
6. Adamczewski, K., Suh, Y., Mu Lee, K.: Discrete tabu search for graph matching. In Proceedings of the IEEE International Conference on Computer Vision, pp. 109–117 (2015)
7. Abyazi-Sani, R., Ghanbari, R.: An efficient tabu search for solving the uncapacitated single allocation hub location problem. Comput. Ind. Eng. **93**, 99–109 (2016)
8. Ansótegui, C., Heymann, B., Pon, J., Sellmann, M., Tierney, K.: Hyper-reactive tabu search for MaxSAT. In Learning and Intelligent Optimization: 12th International Conference, LION 12, Kalamata, Greece, June 10–15, 2018, Revised Selected Papers 12, pp. 309–325, Springer International Publishing (2019)
9. Leite, N., Melício, F., Rosa, A.C.: A fast threshold acceptance algorithm for the examination timetabling problem. Handbook of Operations Research and Management Science in Higher Education, pp. 323–363 (2021)
10. Awad, F.H., Al-Kubaisi, A., Mahmood, M.: Large-scale timetabling problems with adaptive tabu search. J. Intell. Syst. **31**(1), 168–176 (2022)
11. Wu, L., Lin, K., Lin, X., Lin, J.: List-based threshold accepting algorithm with improved neighbor operator for 0–1 knapsack problem. Algorithms **17**(11), 478 (2024)
12. Pérez, O.J., Pazos, R.R., Cruz, R.L., Reyes, S.G., Basave, T.R., Fraire, H.H.: Improving the efficiency and efficacy of the k-means clustering algorithm through a new convergence condition. In: Computational Science and Its Applications–ICCSA 2007: International Conference, Kuala Lumpur, Malaysia, August 26–29. Proceedings. Part III 7, pp. 674–682, Springer Berlin Heidelberg (2007)
13. Barr, R., Golden, B., Kelly, J., Resende, M., Stewart, W.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics **1**(1), 9–32 (1995)
14. Weise, T., Wu, Y., Chiong, R., Tang, K., Lässig, J.: Global versus local search: the impact of population sizes on evolutionary algorithm performance. J. Global Optim. **66**, 511–534 (2016)
15. Beasley, J.E.: OR-Library. Brunel University (2006). http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html
16. Scholl, A., Klein, R.: http://www.wiwi.uni-jena.de/Entscheidung/binpp/ (2003)

17. Lin, C.K.Y., Haley, K.B., Sparks, C.: A comparative study of both standard and adaptive versions of threshold accepting and simulated annealing algorithms in three scheduling problems. Eur. J. Oper. Res. **83**, 330–346 (1995)
18. Glover, F.: Tabu search - part i, first comprehensive description of tabu search. ORSA-Journal on Computing **1**(3), 190–206 (1989)
19. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Computational Physics, pp.161–175 (1990)
20. Fleszar, K., Hindi, K.S.: New heuristics for one-dimensional bin packing. In: Computers and Operations Research, Elsevier Science Ltd, **29**, 821–839 (2002)