A Practical Tuner based on Opposite Information

Nicolás Rojas-Morales
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
nicolas.rojasm@usm.cl

María-Cristina Riff
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
maria.cristina.riff@gmail.com

***Abstract*—Most of the algorithms designed for problem solving have many parameters which values determine their performance. Tuning methods or calibrators are algorithms whose goal is to automate the process of selecting the parameter values of heuristic based algorithms to efficiently solve complex search problems. However, many algorithms are still tuned by-hand either because of the execution time required or the number of scenarios to define before a calibrator is executed. In this work, we propose a practical tuning method that uses a local search procedure that allows obtaining good calibrations in a reduced amount of time, compared to other well-known calibrators. Our tuner has an opposite-inspired learning component used to focus on the most promising areas of the parameter values search space and gathers useful parameter information that is provided to the user. We compare our proposal with two well-known tuners to calibrate two classical optimization problems. We also evaluate the relevance of the opposite-inspired learning component during the search process. A convergence and statistical analysis are presented to confirm that our approach is a good option especially when the user does not have enough time for tuning.**

***Index Terms*—Tuning algorithms, Evoca, ParamILS, NK-landscapes, Ant Knapsack**

I. Introduction

The Parameter Setting Problem (PSP) is the optimization problem that considers the definition of parameter values for a target algorithm [1]. Parameter values can be classified into two categories: (1) numerical (real or integer values) or (2) categorical (a procedure or a function). Usually, the PSP is a hard to solve problem considering the following reasons [2], [3]: (1) it involves to perform several independent executions of the target algorithm, (2) parameters are interrelated between them and their behavior is difficult to predict and understand. (3) for a particular parameter calibration, the performance of the target algorithm can substantially change between problem instances and problems and (4) considering all the possible parameter values and the set of problem instances, the size of the parameter search space is usually huge. Tuning methods are techniques used to obtain suitable parameter values for metaheuristic algorithms. Some tuning methods are SPO [4], REVAC [5], ParamILS [6], Evoca [7], irace [8]. In practice, considering that a tuning process can be a very time consuming task [9], many metaheuristic algorithms are still tuned by hand and their performance could be even better using these methods.

This work proposes a new tuning method that uses opposite information named CLOIS: Calibrating based on Local and Opposite Inspired Search. It was designed considering the following main goal: obtain good parameter calibrations for a target algorithm, considering a "reasonable" amount of effort in terms of the number of evaluations of the target algorithm and the configuration of the tuning algorithm. Inspired in opposition-inspired learning strategies [10], CLOIS contains a component for learning about unpromising parameter values to temporary avoid some regions of the search space. We hypothesize that using CLOIS, a competitive performance of a target algorithm can be obtained, compared to more sophisticated existing tuning tools. It is important to mention that our objective is not to propose the best existing tuning method. However, we are interested in designing an approach for tuning processes where their execution cannot be performed considering plenty of time and their objective is not to reach the "optimal" parameter calibration. Section III present the details of CLOIS.

The contributions of this work are:

- A new method for tuning numerical and categorical parameters of metaheuristic algorithms, that use opposite information and provides useful information to algorithm designers
- The assessment of the our proposal in two different testing scenarios.

To evaluate our approach we consider two different target algorithms: a well-known genetic algorithm proposed for solving the NK landscapes [11] and the well-known Ant Knapsack, proposed for solving the Multidimensional Knapsack Problem [12]. The idea is to study how CLOIS can be used for tuning different metaheuristic algorithms, each one with a different number of parameters. The tuning processes are explained in Section IV. To compare the performance of CLOIS, we considered two well-known tuning methods: Evolutionary Calibrator (Evoca) [7] and Parameter Iterated Local Search (ParamILS) [6]. A discussion with the main differences between our approach and these tuning methods is presented in Section V. Conclusions and some possible paths for future research are presented in Section VI. In the following section we introduce the Parameter Setting Problem.

II. Parameter Setting Problem

This section introduces some preliminary definitions that will be used in the rest of the article. Let us consider a metaheuristic algorithm $\mathcal{M}$ that will be tuned by a tuning

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

method $\mathcal{T}$. The target algorithm $\mathcal{M}$ has N parameters and the assignation of a value v_j for each parameter p_j is defined as a parameter calibration $c_c = [p_1, \dots p_N]$.

Moreover, let us define the parameter setting problem $\mathcal{P}$ as a four-tuple $\mathcal{P} = (\mathcal{M}, \Phi, budget_{max}, \mathcal{C})$ where Φ is the selected set of problem instances that will be considered to tune $\mathcal{M}$, $budget_{max}$ are the available resources for the tuning process and $\mathcal{C}$ is the parameter search space that consider all the possible parameter calibrations. $\mathcal{P}$ is a hard-to-solve optimization problem and its objective is to find a parameter calibration c^* that produces an optimal performance of $\mathcal{M}$. An expected gain $G_P(c_c)$ produced by a candidate parameter calibration $c_c \in \mathcal{C}$ can be defined, measuring its performance in terms of the quality of the obtained solutions by $\mathcal{M}$, its execution time or the number of evaluations performed.

Considering an evaluation function f that measures the performance of $\mathcal{M}$, the expected gain of using c_c can be:

$$G_P(c_c) = f_{\phi \in \Phi}(c_c, \phi) \quad (1)$$

where ϕ is a problem instance in Φ.

III. Our approach

Parameter tuning is the process of searching parameter values that produce an interesting performance of a target algorithm. This process is executed as an offline procedure and in most cases, is a highly time consuming task. Usually, the execution of a tuning method involves the application of specific knowledge about these algorithms to configure some aspects of the tuning process (e.g. domain of the parameters, input data, hyper-parameters). This work proposes a new tuning algorithm named CLOIS, designed to perform a tuning process considering a reasonable amount of resources, in terms of: (1) the number of evaluations of the target algorithm $\mathcal{M}$ and (2) the invested time in configuring the tuning method. Its simplicity produces that using our approach not requires to manage (a lot of) specific knowledge about tuning methods. Is important to mention that we are not motivated in obtaining the optimal performance of $\mathcal{M}$ for all the problem instances in Φ. We propose a tuning method that allows a target algorithm $\mathcal{M}$ to have an acceptable performance, considering that in some real-world situations, designers and final users need to obtain good solutions fast.

The existing interrelation between parameters and the substantial performance changes through the problems in Φ difficult the process of obtaining an optimal parameter calibration. Inspired in existing opposition-inspired learning strategies [10], [13], we decided to learn about when the performance of $\mathcal{M}$ is worsened, caused by some particular parameter values. As most tuning methods are focused on learning and searching for the best suitable parameter values, we define as opposite information when a worsening in the performance of $\mathcal{M}$ is performed. Moreover, in most tuning algorithms, information about the deterioration of the performance of a target algorithm is also available during the tuning process. For this, we decided to include a component in CLOIS with the idea of learning about parameter values that are related to a poor performance of $\mathcal{M}$. Inspired in the Tabu Search algorithm [14], [15], we included a modified tabu list (TL) in CLOIS. The objective is to temporary avoid the evaluation of some parameter calibrations related to these parameter values. Let us consider a candidate calibration c_j that produces a performance $G_P(c_j)$ in $\mathcal{M}$. Also, we named c'_j as the candidate calibration produced when a value v_j in c_j is replaced by v_k. If this modification produces that $G_P(c'_j) < G_P(c_j)$, v_k will be included in the tabu list (TL). The tabu list has a First In, First Out (FIFO) configuration and its size is a parameter of the algorithm (TL_{size}). We expect that the inclusion of a tabu list increase the exploration in CLOIS, avoiding areas of the parameter search space that can be related to a bad performance of $\mathcal{M}$ and as a consequence, to reduce the effort applied during the tuning process.

Algorithm 1 CLOIS

Input: $budget_{max}$, r, k_{nb}, r, TL_{size}
Output: A parameter calibration c_{ret} and Statistics about elements in TL

```
1:  while budget_max not consumed do
2:    c_c ← RandomlyGenerateCalibration()
3:    Evaluate(c_c, r)
4:    while flag_stag do
5:      c_m, v_j ← Perturb(c_c, TL)
6:      Evaluate(c_m, r)
7:      if G(c_m) ≤ G(c_c) then
8:        c_c ← c_m
9:        vis_neigh ← 0
10:     else
11:       makeTabu(TL, v_j, TL_size)
12:       vis_neigh + +
13:     end if
14:     flag_stag ← CheckStagnation(k_nb, vis_nb)
15:   end while
16:   c_ret ← StoreBest(c_c)
17: end while
18: return c_ret, Stats
```

During the design process of CLOIS, we studied different strategies applied when designers tune by hand their approaches. In general, we observed that starting from any parameter calibration, an iterative process is performed: perturb, evaluate and compare calibrations. We decided to represent this process in the design of CLOIS performing its tuning process based in a hill climbing first improvement procedure. We choose this procedure for two main reasons: (1) in most cases, it doesn't visit the whole neighborhood for each candidate parameter calibration and, (2) it doesn't imply the definition of the value of hyper-parameters. The idea is to exploit promising calibrations from different areas of the search space, guided by the tabu list.

Algorithm 1 presents the structure of our approach. CLOIS starts with a randomly generated parameter calibration c_c

(line 2). Then, c_c is evaluated considering the execution of $\mathcal{M}$ with r different random seeds and instances from Φ. To produce a new parameter calibration c_m, a perturbation to c_c is performed (line 8). Here, a randomly selected value v_j is assigned to a randomly selected parameter in $\mathcal{M}$. If a gain is obtained, the search process continues with c_m. However, if a decrement in the performance of $\mathcal{M}$ is obtained, v_j is included in the tabu list (line 11). We included a component to force the exploration during the search process of CLOIS. Usually, in some stages of the search process, metaheuristic algorithms suffer from stagnation. In CLOIS, this can be observed when it has converged to an interesting zone of the search space and it starts visiting most of the neighbors of a parameter calibration. To tackle this situation, a number of k_{nb} neighbors without an improvement are allowed to be visited (lines 9, 12 and 14). When the amount of resources $budget_{max}$ is consumed, CLOIS returns the best obtained calibration c_{ret} (line 16). CLOIS provides additional information about which parameter values were related to a decrement in the performance of $\mathcal{M}$. The information reports some statistics of which parameter was more involved to the tabu list and also, which values were included. The idea is to help the designer or the final user to understand the behavior of the parameters of its approach.

Finally, about the configuration settings in CLOIS, it is necessary to inform if each parameter is a continuous or an integer number. In order to reduce the decisions of the final user, the precision for continuous numerical parameters is set by default in 0.1. However, a higher precision can also used and defined in the execution of CLOIS.

IV. Experiments

This section presents the results obtained for two different tuning scenarios: a Genetic Algorithm named GA-NK [11] with three parameters and an Ant Colony Optimization named Ant Knapsack [12] with five parameters to be tuned. First, we evaluated the effect of using opposite information in CLOIS. To compare the performance of CLOIS, we also considered two well-known tuning methods: ParamILS [6] and Evoca [7].[1] The hardware platform adopted for all these experiments was a Power Edge R630 server with 2 Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, 128 GB of RAM under Ubuntu x64 16.10 distribution. The code of CLOIS is available in https://github.com/nicolasemilio/CLOIS.

A. Tuning GA-NK

GA-NK [11] is a genetic algorithm proposed for solving NK-landscapes. The algorithm evolves a population of ps binary string individuals, starting from a randomly initialized population. Two transformation operators are used during its search process: a bit-flip mutation operator and an uniform crossover operator. The application of these operators is controlled by a mutation rate (mr) and a crossover rate (cr), respectively. To select individuals that will be transformed, a binary tournament selection method is used.

[1]ParamILS code is available in http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/ and Evoca in http://ecco.informaticae.org/

TABLE I
Initial parameters per tuning method for tuning GA-NK

Tuning method	Parameters	Values
ParamILS	Random solutions in first phase (R)	10
	Random solutions at each iteration (s)	3
	Maximum number of executions (max_execs)	2000
	Probability of restarting the search ($p_{restart}$)	0.01
Evoca	Number of random seeds to evaluate (r)	10
	Maximum size of population ($MaxM$)	20
CLOIS	Tabu list size (TL_{size})	9
	Number of neighbors visited (k_{nb})	5
	Number of random seeds to evaluate (r)	3

TABLE II
Tuning settings for each approach tuning GA-NK

Tuning method	Parameter	Values	Precision
ParamILS	cr	{0.0,0.1,...,0.9,1.0}	-
	mr	{0.0,0.1,...,0.9,1.0}	-
	ps	{2,3,...,29,30}	-
Evoca	cr	[0.1,1.0]	0.1
	mr	[0.1,1.0]	0.1
	ps	[2,30]	-
CLOIS	cr	[0.1,1.0]	by default
	mr	[0.1,1.0]	
	ps	[2,30]	

1) Tuning settings: As was used in [11], we considered 15 problem categories, with the value of k ranging from $k = 2$ to $k = 6$. For each category, we considered 1000 problem instances. For the tuning processes, the training set contains one randomly chosen instance per category, using a total of 15 problem instances. We tuned three parameters of GA-NK: the population size (ps), the crossover rate (cr) and mutation rate (mr). To measure the performance of the target algorithm, we considered the number of evaluations performed by GA-NK until the optimal solution is obtained. A maximum number of evaluations was fixed in 1E6. We considered that when the consumed evaluations are lower than 1E6, the problem instance is solved.

Table I shows the initial parameter values used by each tuning method. We considered 5 independent executions of each tuning method, each one with a $budget_{max}$ of 1000 evaluations of GA-NK. Table II presents the tuning settings for each tuning method. In the case of ParamILS, it is necessary to provide each possible discrete value that the algorithm will evaluate, for each parameter value. In the case of Evoca, it is necessary to provide the domain of each parameter value and also, the precision considered for each continuous numerical parameter. About CLOIS, its only necessary to provide the domain of each parameter value. With these tuning settings, the parameter search space contains $2.9E3$ possible parameter calibrations.

2) Evaluation of using opposite information: First, we present an evaluation of the use of opposite information during

TABLE III
CALIBRATIONS OBTAINED BY EACH TUNING METHOD

Tuning method	Seed	cr	mr	ps
ParamILS	S1	0.8	0.3	2
	S2	0.3	0.6	2
	S3	0.2	0.6	4
	S4	1.0	0.6	2
	S5	0.9	0.5	2
Evoca	S1	1.0	0.8	4
	S2	0.7	0.5	2
	S3	0.8	0.9	2
	S4	0.4	0.4	2
	S5	0.5	0.8	2
CLOIS	S1	0.5	0.6	2
	S2	0.4	0.4	2
	S3	0.7	0.8	2
	S4	0.2	0.6	2
	S5	0.8	0.4	3

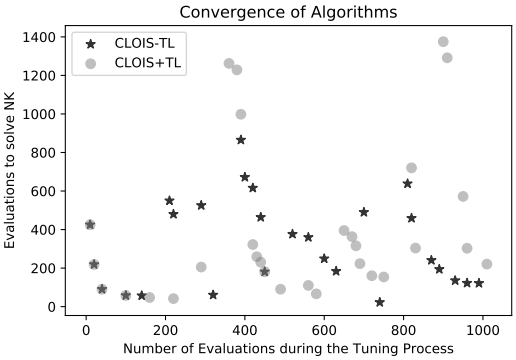


Fig. 1. Convergence of C-TL and C+TL tuning GA-NK

the search process in CLOIS. For this, we compared a version of the algorithm without the tabu list (C-TL) and CLOIS with the OIL component (C+TL). Table IV shows the results of GA-NK using the best parameter calibration provided by each algorithm. As the target algorithm solves all the problem instances using both calibrations, for each category, the table shows the average number of evaluations used by GA-NK to solve each problem instance. A value in bold denotes when an approach outperforms the other one. Results show that using the opposite information (GA-C+TL), CLOIS obtain a better quality parameter calibration.

Figure 1 shows a convergence plot where the x-axis shows the number of evaluations performed in the tuning processes and the y-axis shows the number of evaluations that GA-NK used to solve a problem instance. The plot shows that the use of opposite information modifies the search process of CLOIS, visiting different zones of the parameter search space. In the rest of the article we only consider our approach using the tabu list.

3) Tuning processes: Table III presents the parameter calibrations obtained by each tuning method. ParamILS obtains parameter calibrations that have similar values for the mutation rate and the population size. On the other hand, Evoca and CLOIS obtained parameter calibrations with similar values for the population size and mostly different values for the other two parameters. The average execution time of each tuning method, considering the 5 independent executions, is 12.98 seconds for ParamILS, 17.04 seconds for Evoca and 9.9 seconds for CLOIS.

TABLE IV
EVALUATION OF THE INCLUSION OF OPPOSITE INFORMATION

Categories	GA-C-TL	GA-C+TL
nk_2_20	42.14	**19.81**
nk_2_38	296.67	**110.26**
nk_2_52	1377.92	**378.36**
nk_3_20	84.36	**38.24**
nk_3_34	445.31	**187.74**
nk_3_48	2507.54	**827.23**
nk_4_20	150.46	**67.77**
nk_4_30	564.35	**241.43**
nk_4_40	2191.00	**876.81**
nk_5_20	259.78	**116.08**
nk_5_28	892.30	**396.44**
nk_5_38	3656.22	**1806.87**
nk_6_20	438.69	**211.91**
nk_6_26	1258.77	**531.31**
nk_6_32	3171.18	**1509.94**

TABLE V
AVERAGE NUMBER OF EVALUATIONS PERFORMED BY GA-NK

Calibration Run	GA-PILS	GA-E	GA-C
S1	**401.22**	488.01	624.79
S2	1180.83	**561.24**	616.45
S3	611.10	1200.23	**573.01**
S4	582.97	1005.25	**488.04**
S5	**625.95**	690.89	707.93
Average	680.41	789.12	**602.04**

4) Results: To test the each parameter calibrations, we considered the set of 15000 problem instances and one independent execution of GA-NK per instance. Table V shows the performance of GA-NK considering each parameter calibration obtained by ParamILS (PILS), Evoca (E) and CLOIS (C). Here, the table shows the average number of evaluations used by GA-NK to solve all the problem instances. Results show that the parameter calibrations obtained by CLOIS are competitive to the ones obtained by the other two tuning algorithms. About the execution time, the fewer the number of evaluations, the lower the execution time of the algorithm. However, the execution time is similar between the three algorithms and in most cases is less than one second (per independent execution).

5) Statistical tests: We performed two statistical tests to compare three approaches: GA-NK using one calibration obtained by ParamILS (GA-PILS), using one calibration obtained by Evoca (GA-E) and using one calibration by CLOIS (GA-C). First, we applied the pair-wise Wilcoxon non-parametric test to compare each pair of approaches. Then, in order to compare

TABLE VI
WILCOXON TESTS RESULT

Comparison	Positive Ranks	Negative Ranks	Ties	p-value
GA-PILS - GA-C	7734	6670	596	0.00
GA-E - GA-C	11145	3770	85	0.00
GA-PILS - GA-E	4070	10846	84	0.00

TABLE VII
FRIEDMAN TEST RESULTS

Tuning method	Mean Rank
GA-PILS	1.81
GA-E	2.47
GA-C	1.72

the three algorithms together, we applied the Friedman non-parametric test. The tests were performed considering the used evaluations to solve each problem instance and considering all the independent executions of the three approaches.

Table VI shows the results of applying the Wilcoxon test with the Bonferroni correction. As the objective is to minimize the number of evaluations for solving each problem instance, in the first two comparisons, the positive ranks show the number of cases when GA-C outperforms the other two approaches. In both comparisons, the parameter calibration provided by CLOIS allowed GA-NK to perform a lower number of evaluations. About the comparison between GA-PILS and GA-E, results showed that GA-PILS outperformed GA-E in more cases. Table VII presents the results of the Friedman non-parametric test. Here, the lower the mean rank value the better the performance of the algorithm. Results showed that GA-C obtained the best rank, followed by GA-PILS and by GA-E. All these computations were done using the statistical software package $PSPP$.

6) Additional information about parameters of GA-NK: CLOIS provides additional information about which parameter values were included in the tabu list. Figure 2 shows a bar plot that presents the number of times that each parameter was involved with the tabu list. Here, crossover rate values were related in more cases with a decrement of GA-NK during its tuning process. Figure 3 presents an histogram of the number of times each value was added to the tabu list. The value 0.4 was the most included value, followed by 0.3, 0.5 and 0.8 values. However, some of these values are part of the best parameter calibrations obtained by CLOIS. As there is not a clear behavior in the relationship between some values of cr and the performance of GA-NK, this information can be a hint to use a parameter control method to define the value of cr.

B. Tuning AK

Ant Knapsack (AK) is a $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MMAS}$) [16] algorithm for solving the Multidimensional Knapsack Problem. AK constructs feasible solutions using heuristic knowledge and pheromone information. The heuristic

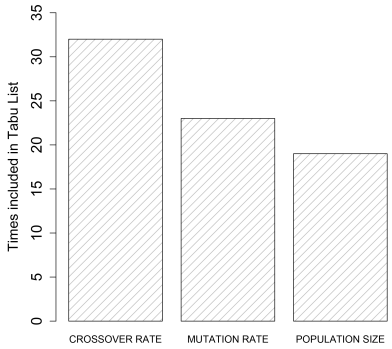


Fig. 2. Number of times a value for each parameter was included to the Tabu List

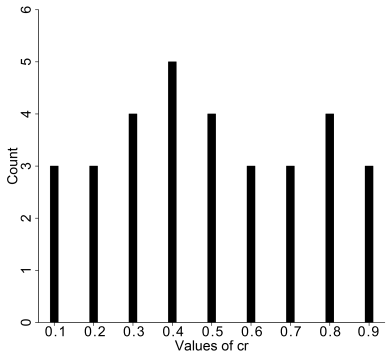


Fig. 3. Frequency of inclusion to the tabu list per crossover rate value

knowledge is focused in maximizing the profit produced by a candidate object, considering how resources are used in all the dimensions. Pheromone information is deposited in pairs of objects that are related to promising candidate solutions. Ant Knapsack has six parameters: α (the importance of the pheromone), β (the importance of the heuristic knowledge), ρ (pheromone evaporation rate), τ_{min} (minimum allowed value of pheromone), τ_{max} (maximum allowed value of pheromone) and nb_{Ants} (number of artificial ants).

TABLE VIII
TUNING SETTINGS FOR EACH APPROACH TUNING ANT KNAPSACK

Tuning method	Parameter	Values	Precision
ParamILS	α	{0.1,0.2...,9.9,10.0}	-
	β	{0.1,0.2...,9.9,10.0}	-
	ρ	{0.1,...,0.9,1.0}	-
	τ_{max}	{0.1,0.2...,9.9,10.0}	-
	nb_{Ants}	{2,3,...,49,50}	-
Evoca	α	[0.1,10.0]	0.1
	β	[0.1,10.0]	0.1
	ρ	[0.1,1.0]	0.1
	τ_{max}	[0.1,10.0]	0.1
	nb_{Ants}	[2,50]	-
CLOIS	α	[0.1,10.0]	by default
	β	[0.1,10.0]	
	ρ	[0.1,1.0]	
	τ_{max}	[0.1,10.0]	
	nb_{Ants}	[2,50]	

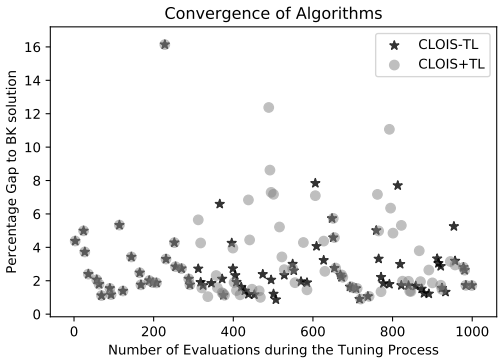


Fig. 4. Convergence of C-TL and C+TL tuning AK

1) Tuning settings: For these tuning processes, we considered four randomly selected large-scale instances from the OR-Lib (OR10x500-0.25_8, OR30x500-0.25_3, OR5x500-0.25_7 and gk01). We tuned five parameters of Ant Knapsack: α, β, ρ, τ_{max} and nb_{Ants}. The parameter τ_{min} was fixed to 0.01 in all the tuning processes. Initial parameter values are almost equally defined as in Table I (for Evoca, r is now 3 and $MaxM$ is now 10). The quality measure used in all the tuning processes was the percentage gap distance between the Best Known (BK) solution per instance and a solution found S_{found}:[2] $gap = 100 \cdot (BK - S_{Found})/BK$. We considered 5 independent executions of each tuning method, each one with a $budget_{max}$ of 1000 evaluations of Ant Knapsack. One parameter calibration evaluation considers 1000 evaluations in Ant Knapsack. Table VIII presents the configuration settings defined per parameter to be tuned. In this case, the number of parameter calibrations in the search space is $4.9E8$. The tuning scenario considers a parameter search space considerably bigger than the GA-NK scenario. This happens for two main reasons: two more parameters are being tuned and also, the number of possible values per parameter is higher.

2) Evaluation of using opposite information: Again, we first evaluate the effect of using opposite information in CLOIS considering the same seed. Figure 4 shows the convergence of the two tuning processes. Particularly, after 300 evaluations of AK, the tuning processes start to differentiate as a consequence of using the tabu list. Table IX shows the evaluation of the obtained parameter calibrations. Underline values show when an approach outperforms the other in terms of the average percentage gap (considering the 20 independent executions). Values in bold denote when one algorithm outperforms the other in terms of the percentage gap of the best quality solution reached. Results show that the use of opposite information allowed CLOIS to reach a better parameter configuration.

3) Tuning processes: Table X presents the best parameter calibration obtained by each tuning method. To obtain these calibrations, the execution time of ParamILS was 32883 seconds, Evoca was 33927 seconds and CLOIS was 32556 seconds. We also analyzed the convergence of each tuning process. ParamILS obtained its best parameter calibration after 300 evaluations of AK. Then, any better candidates were obtained. In the case of Evoca, the best parameter calibration was obtained after 888 evaluations. About CLOIS, our tuning method obtained its better configuration after 504 evaluations of the target algorithm.

4) Results: To test the parameter calibrations reported in table X, we considered 8 large-scale problem instances from the ORLib. For each problem instance, 20 independent executions were considered, each one with 1000 evaluations. In the following, we will name Ant Knapsack using the calibration obtained by ParamILS as AK-PILS, using the calibration of Evoca as AK-E and using the calibration of CLOIS as AK-C.

Table XI presents the percentage gap of the best quality solution obtained by the three approaches. Values in bold denote when an approach outperforms the other two techniques. Results show that the best calibration obtained by CLOIS is competitive, outperforming the other approaches in 5 out of 8 cases. Considering the 8 instances, as the three algorithms didn't obtain optimal solutions, they consumed all the provided evaluations and the execution times were similar. Table XII

[2]Best Known solutions are available in http://www.eecs.qmul.ac.uk/∼jdrake/bestresults.html

TABLE IX
EVALUATION OF THE INCLUSION OF OPPOSITE INFORMATION

Instance	Average		Best	
	AK-C-T	AK-C+T	AK-C-T	AK-C+T
OR10x500-0.75_4	0.270	0.166	0.164	**0.090**
OR30x500-0.50_3	0.741	0.721	0.446	**0.286**
OR30x500-0.75_3	0.350	0.392	0.287	**0.246**
OR5x500-0.50_6	0.425	0.257	0.322	**0.137**
OR5x500-0.50_7	0.519	0.390	0.372	**0.239**
OR5x500-0.75_1	0.260	0.139	0.196	**0.086**
gk02	1.395	1.222	0.909	**0.758**
gk03	1.356	1.320	1.078	**0.866**

TABLE X
BEST PARAMETER CALIBRATION OBTAINED BY EACH TUNING METHOD

Tuning method	α	β	ρ	τ_{max}	nb_{Ants}
ParamILS	2.2	7.2	0.3	9.5	34
Evoca	2.1	6.1	0.1	3.1	24
CLOIS	2.3	6.1	0.2	8.5	41

TABLE XI
THE PERCENTAGE GAP OF THE BEST QUALITY SOLUTION OBTAINED BY THE THREE APPROACHES

Instance	AK-PILS	AK-E	AK-C
OR10x500-0.75_4	0.126	0.160	**0.090**
OR30x500-0.50_3	0.467	0.502	**0.286**
OR30x500-0.75_3	**0.151**	0.257	0.246
OR5x500-0.50_6	0.803	0.244	**0.137**
OR5x500-0.50_7	0.967	0.276	**0.239**
OR5x500-0.75_1	2.039	0.107	**0.086**
gk02	**0.657**	0.808	0.758
gk03	**0.707**	0.990	0.866

TABLE XII
THE AVERAGE PERCENTAGE GAP OF THE THREE APPROACHES CONSIDERING 20 INDEPENDENT EXECUTIONS

Instance	AK-PILS	AK-E	AK-C
OR10x500-0.75_4	0.197	0.254	**0.166**
OR30x500-0.50_3	**0.668**	0.873	0.721
OR30x500-0.75_3	**0.328**	0.444	0.392
OR5x500-0.50_6	1.481	0.357	**0.257**
OR5x500-0.50_7	1.618	0.459	**0.390**
OR5x500-0.75_1	2.214	0.202	**0.139**
gk02	**1.169**	1.234	1.222
gk03	1.321	1.355	**1.320**

TABLE XIII
WILCOXON TESTS RESULT

Comparison	Positive Ranks	Negative Ranks	Ties	p-value
AK-PILS - AK-C	107	52	1	0.00
AK-E - AK-C	117	43	0	0.00
AK-PILS - AK-E	89	70	1	0.00

presents the average percentage gap obtained by the three approaches, considering the 20 independent executions. Again, results show that the parameter calibration obtained by CLOIS allowed Ant Knapsack to reach a competitive performance, outperforming the other two approaches in 5 out of 8 cases.

5) Statistical tests: As in Section IV-A5, we performed two statistical tests: the pair-wise Wilcoxon and the Friedman non-parametric tests. Table XIII shows the results of applying the Wilcoxon test with the Bonferroni correction. First, the results show that the three approaches are statistically different. As the objective is to minimize the percentage gap, the number of positive ranks show when AK-C outperforms AK-PILS and AK-E in the first two comparisons. In the last comparison, results show that AK-E outperformed AK-PILS in more cases. Table XIV presents the results of the Friedman test. Results confirm that the parameter calibration obtained by CLOIS allowed AK to obtain the better performance. All these computations were done using the statistical software package $PSPP$.

6) Additional information about parameters of AK: As we mentioned, CLOIS reports information about the use of opposite information. Figure 5 shows the number of times that each parameter was involved in a decrement of the performance of Ant Knapsack during its tuning process. Here, we observe that values of parameters β and τ_{max} were the most included in the tabu list. Figure 6 shows histograms for values of β and τ_{max} that were tabu. In the case of β, values

TABLE XIV
FRIEDMAN TEST RESULTS

Tuning method	Mean Rank
AK-PILS	2.23
AK-E	2.17
AK-C	1.60

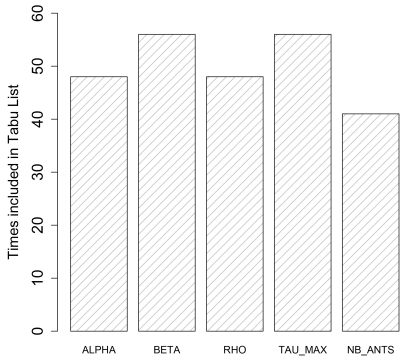
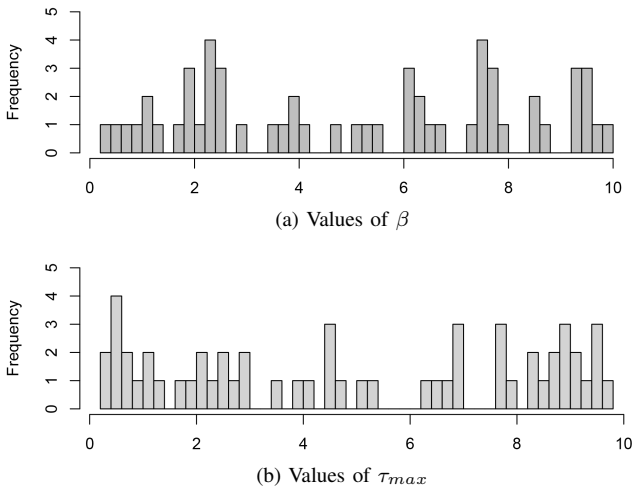


Fig. 5. Number of times a value for each parameter was included to the Tabu List



(a) Values of β

(b) Values of τ_{max}

Fig. 6. Histograms of the frequency of inclusion in the tabu list per value

near to 8.0 and near to 2.0 were included the most to the tabu list. About τ_{max}, values between 0.1 and 1.5 were the most involved with a deterioration of the performance of AK.

V. DISCUSSION

Parameter tuning methods can be classified in hand-made tuning, tuning by analogy, tuning by statistical analysis and tuning by search-based approaches [1]. Hand-made tuning is the process performed by a designer where the target algorithm is iteratively evaluated using a defined set of parameter calibrations for a selected set of problem instances. In the case of tuning by analogy, the parameter values of an algorithm are defined considering existing approaches in the literature. Tuning by statistical analysis uses statistical methods to detect which parameter values can be related to a good performance of the target algorithm. Tuning by a search-based approach are techniques designed considering a search strategy. Based in a taxonomy presented in [9], we present the main characteristics of CLOIS, ParamILS and Evoca in Table XV.

VI. CONCLUSIONS AND FUTURE WORK

This work presents CLOIS, a new tuning algorithm inspired in the tuning-by-hand methods and in the existing opposition-

TABLE XV
COMPARISON OF THE THREE TUNING ALGORITHMS

Criterion		ParamILS	Evoca	CLOIS
Search-based		Iterated Local Search	Evolutionary Algorithm	Hill Climbing and OIL strategy
Parameter space definition	Discretized	X		
	Non-discretized		X	X
Output	Best calibration	X		X
	Set of calibrations		X	
Conditional parameters		Yes	Yes	No
Hyper-parameters		4	2	3
Parameter Precision required		No	Yes	No
Stop criterion			Maximum computational budget	

inspired learning strategies. Our approach is focused in obtaining promising parameter calibrations with a simple design. CLOIS includes a component in order to learn from the "opposite", when a decrement in the performance of the target algorithm is produced. For this, a tabu list temporary stores parameter values that were involved in a poor performance of the target algorithm. A first improvement hill climbing procedure guides the tuning process to visit regions that contain suitable parameter values. In order to tackle possible stagnation during the search process, a fixed number of neighbors without an improvement are allowed to be visited.

To evaluate our tuning algorithm we consider two different scenarios: GA-NK and Ant Knapsack. Also, we compared CLOIS with two well-known tuning methods: ParamILS and Evoca. Results show that the calibrations provided by the three tuning methods allowed GA-NK to solve all the instances. Statistical analysis confirm that the parameter values obtained by CLOIS allowed GA-NK to reach a good and a competitive performance. In the second scenario, results showed that the parameter calibration provided by CLOIS allowed Ant Knapsack to reach good quality solutions. Moreover, results in Wilcoxon and Friedman tests confirm that the calibration provided by CLOIS is competitive to the ones provided by Evoca and ParamILS. In both scenarios, we evaluated the effect of using opposite information in CLOIS. Results show that the inclusion of the tabu list allowed CLOIS to reach good parameter calibrations. Finally, we observed that in both tuning scenarios, the execution time of CLOIS is quite lower than Evoca and ParamILS. For future work, we are interested in further analyze the effect of the hyperparameters of CLOIS (e.g. TL_{size}, k_{nb}).

ACKNOWLEDGMENT

Authors want to thank Elizabeth Montero for providing us the code of Evoca and to help us for defining the tuning scenarios. This work was supported by UTFSM Project PI_LI_19_16 and FONDECYT Project Number 1200126.

REFERENCES

[1] E. Montero, M.-C. Riff, and B. Neveu, "A beginner's guide to tuning methods," *Applied Soft Computing*, vol. 17, pp. 39–51, 2014.

[2] B. Doerr and C. Doerr, "Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices," *Computing Research Repository*, vol. abs/1804.05650, 2018. [Online]. Available: http://arxiv.org/abs/1804.05650

[3] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer, 2007, vol. 54, pp. 19–46.

[4] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss, "Sequential parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*. IEEE, 2005, pp. 773–780.

[5] V. Nannen and A. E. Eiben, "Efficient relevance estimation and value calibration of evolutionary algorithm parameters," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore*. IEEE, 2007, pp. 103–110.

[6] F. Hutter, T. Stützle, K. Leyton-Brown, and H. Hoos, "Paramils: An automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009. [Online]. Available: https://doi.org/10.1613/jair.2861

[7] E. Montero and M.-C. Riff, "A new algorithm for reducing metaheuristic design effort," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*. IEEE, 2013, pp. 3283–3290.

[8] M. Lopez-Ibanez, J. Dubois-Lacoste, L. Pérez, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[9] E. Montero, M.-C. Riff, and N. Rojas-Morales, "Tuners review: How crucial are set-up values to find effective parameter values?" *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 108–118, 2018.

[10] N. Rojas-Morales, M.-C. Riff, and E. Montero, "A survey and classification of opposition-based metaheuristics," *Computers & Industrial Engineering*, vol. 110, pp. 424–435, 2017.

[11] M. Pelikan, "Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes," in *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, C. Ryan and M. Keijzer, Eds. ACM, 2008, pp. 1033–1040.

[12] I. Alaya, C. Solnon, and K. Ghedira, "Ant algorithm for the multi-dimensional knapsack problem," in *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004)*. Citeseer, 2004, pp. 63–72.

[13] N. Rojas-Morales, "Opposite learning strategies for improving the search process of ant-based algorithms," Ph.D. dissertation, Universidad Técnica Federico Santa María, 2018.

[14] F. W. Glover, "Tabu search - part I," *INFORMS Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[15] ——, "Tabu search - part II," *INFORMS Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[16] T. Stützle and H. Hoos, "MAX–MIN Ant System," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.