ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 6, No. 1, 134-145 (2021)
www.astesj.com
Special Issue on Multidisciplinary Innovation in Engineering Science & Technology

ASTES Journal
ISSN: 2415-6698

Balancing Exploration-Exploitation in the Set Covering Problem Resolution with a Self-adaptive Intelligent Water Drops Algorithm

Broderick Crawford[*,1], Ricardo Soto[1], Gino Astorga[2], José Lemus-Romani[1], Sanjay Misra[3], Mauricio Castillo[1], Felipe Cisternas-Caneo[1], Diego Tapia[1], Marcelo Becerra-Rozas[1]

[1]*Pontificia Universidad Católica de Valparaíso, Valparaíso, 2340000, Chile*

[2]*Universidad de Valparaíso, Valparaíso, 2340000, Chile*

[3]*Covenant University, Ota, 100001, Nigeria*

ARTICLE INFO

Article history:
Received: 31 August, 2020
Accepted: 23 December, 2020
Online: 15 January, 2021

Keywords:
Metaheuristics
Autonomous Search
Combinatorial Optimization
Set Covering Problem
Intelligent Water Drops

ABSTRACT

The objective of the metaheuristics, together with obtaining quality results in reasonable time, is to be able to control the exploration and exploitation balance within the iterative processes of these methodologies. Large combinatorial problems present ample search space, so Metaheuristics must efficiently explore this space; and exploits looking in the vicinity of good solutions previously located. The objective of any metaheuristic process is to achieve a "proper" balance between intensive local exploitation and global exploration. In these processes two extreme situations can occur, on the one hand an imbalance with a bias towards exploration, which produces a distributed search in the search space, but avoiding convergence, so the quality of the solutions will be low, the other case is the bias towards exploitation, which tends to converge prematurely in local optimals, impacting equally on the quality of the solutions. To make a correct balance of exploration and exploitation, it is necessary to be able to control adequately the parameters of the Metaheuristics, in order to infer in the movements taking advantage of the maximum capacity of these. Among the most widely used optimization techniques to solve large problems are metaheuristics, which allow us to obtain quality results in a short period of time. In order to facilitate the use of the tools provided by the metaheuristic optimization techniques, it is necessary to reduce the difficulties in their configuration. For this reason, the automatic control of parameters eliminates the difficult task of obtaining a correct configuration. In this work we implemented an autonomous component to the Intelligent Water Drops algorithm, which allows the control of some parameters dynamically during the execution of the algorithm, achieving a good exploration-exploitation balance of the search process. The correct functioning of the proposal is demonstrated by the Set Covering Problem, which is a classic problem present in the industry, along with this we have made an exhaustive comparison between the standard algorithm and the autonomous version that we propose, using the respective statistical tests. The proposal presents promising results, along with facilitating the implementation of these techniques to industry problems.

1 Introduction

This paper is an extension of work "An Adaptive Intelligent Water Drops Algorithm for Set Covering Problem", originally presented in 19th International Conference on Computational Science and Its Applications (ICCSA) [1].

The post-pandemic economic recovery brings with it a number of challenges for the industry, ranging from rethinking business models to making good use of every available resource. In this sense, optimization is an important tool to achieve the desired recovery [2]–[6].

There are various economic sectors where there are problems that need to be optimised such as the airport sector where, given the environmental restrictions, it is becoming increasingly difficult to build new airports and for this reason it is necessary to optimise each of the tasks of the current installations [7]–[9]. Another ex-

[*]Corresponding Author: Broderick Crawford, broderick.crawford@pucv.cl

ample is the educational sector, in particular the universities where it is necessary to optimize the use of rooms and the curriculum of students [10, 11]. In summary, we can find a series of problems that can be treated through optimization techniques that have a positive impact on economic recovery.

In this sense the metaheuristics that are supervised heuristics [12], are a real option to give solution to the industrial optimization problems since their attraction is that they allow to deliver a high quality solution that can even be the optimal one in limited computation times. There is a great variety of metaheuristics which have been used to solve NP-Hard problems [13]–[16].

In general, metaheuristics have a great amount of parameters, which allow controlling the high and low level strategies of these, having direct relation with the performance of the techniques in the different problems [17], the optimal configuration of parameters constitutes in itself an optimization problem [18, 19]. The importance of a good parameter assignment to metaheuristic algorithms impacts on the quality of solutions, but on the other hand it is not possible to consider transversal parameters for all the problems to be solved, since the assignment of values depends directly on the problem and the instances to be solved [20].

The values that can assume the parameters can have a great amount of combinations, for that reason we can occupy different strategies to approach the optimal configuration of parameters, that according to the literature are divided in two great groups, off-line configuration and on-line control. The off-line configuration of parameters corresponds to the determination of the metaheuristic algorithm outside the run, that is, before its execution begins [21], while the on-line control dynamically updates the values of the algorithm during the execution [22].

Given the current context of using metaheuristic techniques to solve the problems of the industry, we must facilitate the use of these techniques to non-expert users, so the use of autonomous techniques for the control of parameters, is a great contribution to bring this type of tools to all users of the industry.

The present work, solves a classic problem of the real world, as it is it Set Convering Problem (SCP) [23], which has diverse applications in the industry, using metaheuristic techniques to solve problems of great dimension. We have used the metaheuristic technique Intelligent Water Drops (IWD), which is inspired by the physical behavior of water droplets in a river bed [24], Along with this, we have incorporated autonomous elements for the on-line control of its parameters, which has two main components, the first is the obtaining of external information on the problems to be solved, and the second is the obtaining of internal information on the behaviour of the algorithm. With this we can better combine the information of each problem to be solved, along with the internal behavior of the algorithm, which generates that our proposal is adapted to the various problems that are solved.

The work is structured in the following form, in Section 2 what has been done regarding parameter tuning is presented, in Section 3 the functioning of IWD is explained, while in Section 4 the adaptive implementation proposed for IWD is detailed, in Section 5 set covering problem is shown, in Section 6 the corresponding experiments and statistical analyses are presented, ending in Section 7 with the conclusions and future work.

2 Parameter Tuning

In this section we will review the various techniques of parameterization of algorithms in order to find a correct configuration.

The proper selection of parameter values for the algorithms is not an easy task and has an important impact. Usually a great deal of time is spent, using previous experiences and expert knowledge for the correct assignment of values which also constitutes a not easy task for non-expert users. In consideration of the latter, the ideal condition from the point of view of the non-expert user is that he or she should give general information about the problem and with the least possible interaction receive an adequate response (Fig. 1). This concept, called Autonomous Search, was proposed by [18], using indicators to determine the best strategy to use.

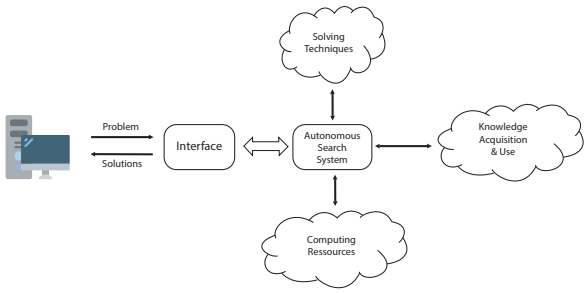


Figure 1: Autonomous Search.

The techniques for finding a good configuration for the parameters were initially grouped into two categories: offline configuration and online control. However, a new proposal is made in [21]: Simple Generate-Evaluate Methods, Iterative Generate-Evaluate Methods and High-Level Generate-Evaluate Methods.

2.1 Offline configuration

Finding a convenient parameter configuration previous to the execution of the algorithm is known as offline configuration.

This is primarily a trial and error process and can consume considerable time in research. The efficiency of this depends mainly on the insight and knowledge of the researcher or author of the algorithm. These processes are typically undocumented and are not reproducible, often driving to an unequal adjustment of different algorithms.

The following techniques exist within this group:

- F-race is an inspired method of racing algorithms in automatic learning, in particular, races in [25, 26]. This algorithm was presented in [27] and studied in detail in Birattari's PhD thesis [28]. The purpose of these methods is to iteratively evaluate a set of candidate configurations for a given set of instances. A set is removed when there is sufficient statistical evidence and there are survivors within the continuous race. This method is used after each evaluation of the candidate configurations, in which nonparametric Friedman two-way variance analysis [29] determines if there is evidence that at less than one of the configurations is significantly different

from the other ones. If the hypothesis is null and there is no difference, it is rejected; Friedman's subsequent tests are applied to remove these candidate configurations that are significantly worse than the best. This approach had been used by the ACO in [30].

- ParamILS was introduced by [31] as a versatile stochastic local search approach for automatic algorithm configuration. ParalILS includes methods to optimize the performance of an algorithmic, deterministic or stochastic object in a particular class of problems by switching a set of ordinal and/or categorical parameters. This proposal is based on an iterated local search algorithm that uses a combination of configurations chosen, a priori or at random, at initialization. The underlying idea of local search is an iterative "best first" method, which uses a variety of random movements to disturb the solution and avoid getting into local minima. The method always accepts configurations that improve or match the performance of the best configuration and restarts the search from a random configuration under a certain probability. The local search method moves through the search space by modifying only one parameter each time[32]. This approach has been used to configure ACO to solve transportation planning problems in [33].

- The other approach to optimization, focused on the sequential paradigm, includes improving the parameter initial values by the alternation of experimental design and the parameter recognition.

  In this paradigm, statistical significance takes a preponderant role, since each new experiment contributes with information about the performance of the parameters used, which are then referenced to the new stages.

  In the case of a parallel method, different experiments are formulated concurrently (all of them taking the same parameter nominal values); experiments are then carried out. The parameters are calculated and their suitability tested using the data obtained in all parallel experiments. If the data obtained from the parallel experiments are inadequate after the parameter estimation the procedure can be replicated [34, 35].

- In [36], the author used the graphic radial technique. The four basic metrics are employed for this method: worst case, best case, average case and average run-time. The region under the radar plot curve is obtained with these four metrics to set the best setting.

- The meta-optimization method was initially studied by [37] and is characterized by using an algorithm to optimize the parameters of another optimization algorithm, that is, there is an algorithm at a higher level that optimizes the parameters of a lower-level algorithm. This method has been used for covering problems in facility locations in [38], stand management in [39], and parameter selection in machine learning in [40].

2.2 Online configuration

One of the areas of research that has taken great strength in recent years is the online configuration, as it gives the algorithms the ability to adapt to the characteristics of a particular instance for better performance. For this online configuration to be useful, the exploration and exploitation phases must be clearly identified because the internal adjustments may be different for each of the phases. Thus, the online configuration can improve the results when the algorithms are used in situations very different from those that were built.

Different techniques exist, and the most simple is to define the parameter variation rules before executing the algorithm.
One possibility is the use of parameter adaptation, where the parameter modification scheme is defined for some behavior statistics of the algorithm.

- Absolute evidence: An example is the threshold of the distance between the solutions.

- Relative evidence: Considers that the relative difference in the performance with parameters of different values adapt to those with the best behavior.

2.3 Simple Generate-Evualate Methods

In this method the principle of generating and evaluating is used. Candidate configurations are occupied and then each of the parameters are evaluated to find the best configuration (Figure 2).

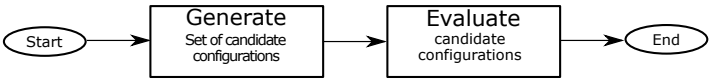


Figure 2: Simple Generate-Evualate Methods.

2.4 Iterative Generate-Evualate Methods

Unlike the first method, this one repeatedly performs the generation and evaluation steps. In this method the historical information allows to guide the generation, so that the search space for parameters is better explored and is more efficient in large search spaces (Figure 3).

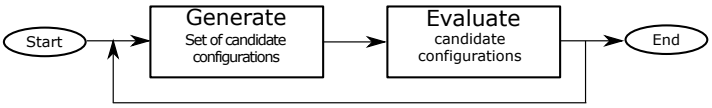


Figure 3: Iterative Generate-Evaluate Methods.

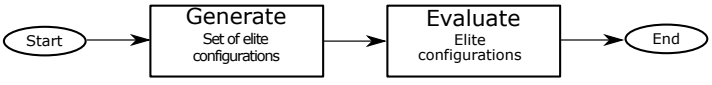


Figure 4: High-Level Generate-Evualate Methods.

2.5 High-Level Generate-Evualate Methods

This approach generates candidate configurations to then evaluate them and select the best configuration. The idea is to generate a set of high quality configurations quickly using few computational resources and select the best one. The idea is to greatly reduce the resources used when exploring candidate configurations and use them to thoroughly evaluate these configurations (Figure 4).

3 Intelligent Water Drops Algorithm

During the year 2007 in [24] he presented the algorithm that is classified as constructive since it builds a solution from 0. Its inspiration is based on nature and corresponds to the displacement that water drops make through the flow of a river and its friction with the earth, considering that when a drop is moved from one point to another it displaces earth from the river bed making the friction less and less which makes the drops increase their speed and in turn incorporate part of the extracted earth.

In the Figure 5, we can see how the soil of the river behaves, on the one hand during the passage of the drop is removed soil from the bottom of the river and on the other hand is incorporated into the drop which directly influences the speed that is acquired, this is demonstrated in that two drops of similar size but with different speeds at the end of its movement are with different size because of the soil that is incorporated. This abstraction of the behavior of water drops in river soil, allows us to build diversified solutions at the beginning, while during the execution of the algorithm, these solutions converge to promising search spaces, which allows to intensify the search.

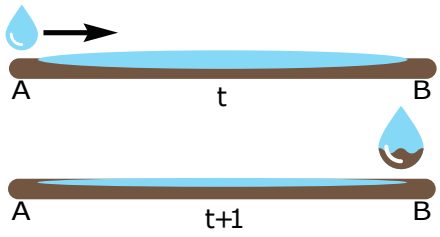


Figure 5: Water drop removing riverbed soil and adding to its own soil.

This algorithm has been used to solve mainly scheduling problems: Cooperative Search Path Optimization problem [41], Work Flow Scheduling Algorithm for Infrastructure as a Service (IaaS) cloud [42], Capacitated Vehicle Routing Problem [43], Trajectory Planning of Unmanned Combat Aerial Vehicle (UCAV) is a rather complicated Global Optimum Problem in UCAV Mission Planning [44], Method for Optimal Location and Sizing of Distributed Generation (DG) [45].

This algorithm has an initialization phase where both the static parameters number of drops, number of iterations, initial soil, initial speed and constants a_s, b_s, c_s, a_v, b_v, c_v are initialized and also the dynamic parameters Speed of drop k and soil value of drop k are initialized.

Then there is a construction phase where every drop builds a solution by visiting the nodes. The Eq. 1 is used to determine which node to visit:

$$p_i^k(j) = \frac{f(soil(i,j))}{\sum_{\forall l \notin VC^k} f(soil(i,l))} \tag{1}$$

where $soil(i,j)$ corresponds to the amount of soil between i and j and is calculated according to the following equation (Equation 2):

$$f(soil(i,j)) = \frac{1}{\varepsilon + g(soil(i,j))} \tag{2}$$

where ε is a small positive value to avoid division by zero.

The function g(soil(i,j) always obtains a positive soil value (Equation 3):

$$g(soil(i,j)) = \begin{cases} soil(i,j) & \text{if } \min_{\forall l \notin VC^k} soil(i,l) \geq 0, \\ soil(i,j) - \min_{\forall l \notin VC^k} soil(i,l) & \text{otherwise} \end{cases} \tag{3}$$

Once the drop has selected the new node, its velocity should be updated (equation 4):

$$vel^k(t+1) = vel^k(t) + \frac{a_v}{b_v + c_v \cdot soil^2(i,j)} \tag{4}$$

where a_v, b_v and c_v correspond to static parameters, and $soil(i,j)$ represents the amount of soil between i and j.

Next, it is necessary to update the soil after the step of the drop; for this, we use the following equation 5:

$$soil(i,j) = (1-\rho) \cdot soil(i,j) - \rho \cdot \Delta soil(i,j) \tag{5}$$

where ρ is a small positive value between 0 and 1, and $\Delta soil(i,j)$ is the amount of soil removed from paths i and j.

The value of $\Delta soil(i,j)$ is obtained with the following equation (6):

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s \cdot time(i,j : vel^k(t+1))} \tag{6}$$

where a_s, b_s and c_s are static parameters, and $time(i,j : vel^k(t+1))$ is calculated as follows (7):

$$time(i,j : vel^k(t+1)) = \frac{HUD(i,j)}{vel^k(t+1)} \tag{7}$$

where HUD is a heuristic that measures the degree of undesirability of the drop to jump from one node to another.

The Reinforcement phase is responsible for updating the global soil with the best drop of the iteration, and the following equation is used (8):

$$soil(i,j) = (1 + p_{iwd}) \cdot soil(i,j) - p_{iwd} \cdot \frac{1}{q(T^{IB})} \tag{8}$$

where p_{iwd} is a small positive value between 0 and 1, and $q(T^{IB})$ is the fitness value.

The best solution for each iteration is compared with the best global solution and updated according to the following equation (9):

$$T^{TB} = \begin{cases} T^{IB} & \text{if } q(T^{IB}) < q(T^{TB}) \\ T^{TB} & \text{otherwise} \end{cases} \tag{9}$$

Finally, the Termination phase is responsible for the completion of the algorithm process when the stop condition is met, which can be the number of iterations.

In 6 the original algorithm of the metaheuristics is shown.

Algorithm 1 Intelligent Water Drops Algorithm

{**Initialization phase**}
Initialize : Amount iterations, Amount drops.
Initialize : a_s, b_s, c_s
Initialize : a_v, b_v, c_v
repeat
  for $k = 1$ to N **do**
    Initialize VC^k list as empty.
    Initialize $soil^k$ value as zero.
    Initialize vel^k value as $InitVel$.
    Create the k^{th} water drop (IWD^k).
    Select randomly a node for IWD^k.
    Update VC^k.
  end for
  {**Construction phase**}
  for $k = 1$ to N **do**
    Choose a path for IWD^k.
    Update velocity (vel^k).
    Compute the amount of soil ($\Delta soil$) to be carried by IWD^k
    Remove $\Delta soil$ from the path and add it to IWD^k
  end for
  for $k = 1$ to N **do**
    Calculate $fitness^k$
  end for
  {**Reinforcement phase**}
  Update Paths of the Best Solution
until $i = MAX\_ITERATION$
{**Termination phase**}

Figure 6: Intelligent Water Drops Algorithm.

4 Adaptative Intelligent Water Drops

The configuration of initial parameters is a costly task, carried out by an expert user and performed prior to the execution of the algorithm. We can reduce the cost of this task based on the concepts from Autonomous Search [18], which obtains information from two different sources, internal, corresponding to the operation of the algorithm and external, corresponding to the problem we are solving. The use of these data sources allows us to compare the quality of the solution in the current iteration (f_{IB}) with the quality of the best found solution (f_{TB}).

The general scheme for obtaining information to guide the adjustment of parameters is presented in Figure 7.

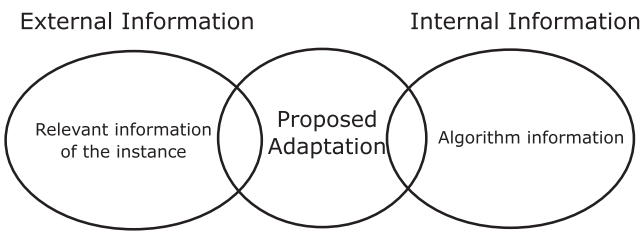


Figure 7: Information required for the proposal.

The internal information source considers the performance of the algorithm. In our proposal, we collect the number of iterations in which no improvement in the quality of the solutions is detected. The external information source comes from the data corresponding to the instance of the problem we are solving. We consider the number of columns and the density of the instance, which allows us to differentiate them. A weight is assigned to the columns of the instance that allows us to assign importance to them, multiplying it by the density, given in the Figure 8).

SET	**Instance Set**	m	n	Cost Range	Density	Optimum Solution
4	10	200	1000	[1,100]	2.00	Known
5	10	200	1000	[1,100]	2.00	Known
6	5	200	1000	[1,100]	5.00	Known
A	5	300	1000	[1,100]	2.00	Known
B	5	300	1000	[1,100]	5.00	Known
C	5	400	1000	[1,100]	2.00	Known
D	5	400	1000	[1,100]	5.00	Known
NRE	5	500	1000	[1,100]	10.00	Known
NRF	5	500	1000	[1,100]	20.00	Known

Figure 8: Standard description of the benchmark OR-library.

The algorithm has two initial parameters to which we assign values, these correspond to the Soil and Initial Speed ($Soil_{Initial}$ and $Velocity_{Initial}$ respectively).

In the first case we assign the value considering the number of columns of the instance and its density, which is done as follows:

To determine the first value we consider information of the instance considering on one hand the number of columns available and on the other hand the density that is presented. This value is obtained in the following way:

$$Soil_{Initial} = SoilInitial \cdot nColumn \cdot Density \tag{10}$$

Where $Soil_{Initial} = 1$, $nColumn$ corresponds to the number of columns in the instance group and $Density$ corresponds to the percentage of nonzeros in the matrix.

$$Velocity_{Initial} = 15 \tag{11}$$

For the case of the initial speed $Velocity_{Initial}$ the value is determined by a series of 10 previous executions of the original algorithm.

For the adaptation, in this work, it is considered the quality of the solution for which a weight is calculated that will allow us to update the local velocity, avoiding that the solution remains trapped in an optimal local. This percentage will be calculated by comparing the solution delivered in each iteration with the best solution obtained.

It has been experimentally determined that a low initial drop velocity impacts the convergence of solutions, making it more pronounced. This characteristic is taken into account in the parameter configuration.

Initial soil and velocity parameters update rule is given by the following equation:

$$Percentage = (f_{IB} \cdot 100)/f_{TB} \tag{12}$$

where f_{IB} is the best solution obtained at the current iteration, and f_{TB} is the best global solution found.

$$Soil_{Initial} = Soil_{Initial} \cdot Percentage \tag{13}$$

where $Soil_{Initial}$ is the available initial soil at the current iteration and $Percentage$ is the percentage difference between f_{IB} and f_{TB}.

$$Velocity_{Initial} = Velocity_{Initial} \cdot Percentage \cdot \beta \tag{14}$$

where $Velocity_{Initial}$ is the available initial velocity at current iteration and β is a random number in the interval $[0, 1]$.

Figure 9 shows the proposed adaptation for the local soil parameter, aiming to improve the metaheuristic performance.

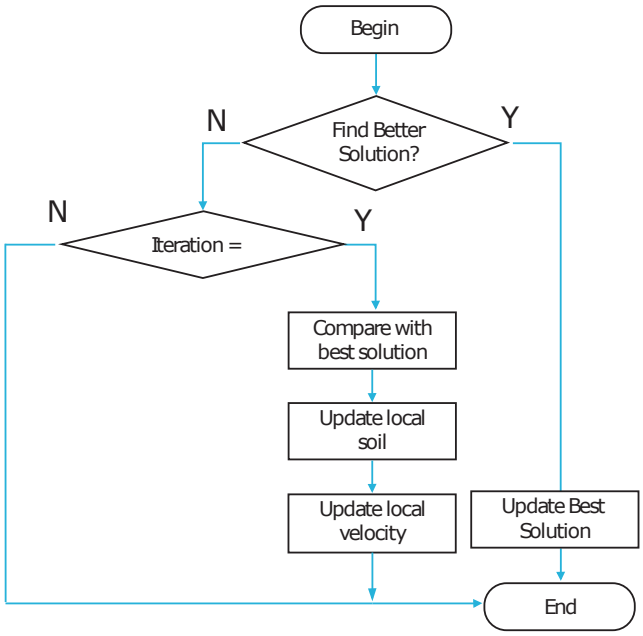


Figure 9: Parameter adaptation proposal.

At each iteration the improvement of the best solution found is evaluated. If an improvement is detected, the previous best solution found is replaced. Otherwise, a counter for not improved iterations is increased. An α value is set for the maximum number of iterations without improvement. If the counter is equal compared with the α parameter, the percentage difference of the current solution compared to the best solution is obtained and used to increase the local soil for the next iteration.

To obtain the value for the α parameter we perform 10 executions of the standard algorithm for each instance. The elapsed iteration number until a fitness improvement was evaluated. The analysis of this number allows us to determine the value for the α parameter to 5. A greater α value causes the fitness improvement probability to decrease, and the computation cost increases. The results of this analysis is shown in Figure 10, the column % represents the occurence percentage for each α, and *%Acum* column shows the accumulated percentage for each α. Figure 11 shows the fitness change for two instances.

α	Occurrences	%	% Acum
1	41	59%	59%
2	10	14%	74%
3	6	9%	83%
4	4	6%	88%
5	1	1%	90%
6	2	3%	93%
7	0	0%	93%
8	0	0%	93%
9	1	1%	94%
10	1	1%	96%
11	1	1%	97%
12	0	0%	97%
13	1	1%	99%
14	0	0%	99%
15	0	0%	99%
16	1	1%	100%
17	0	0%	100%
18	0	0%	100%
19	0	0%	100%
20	0	0%	100%
20+	0	0%	100%

α	Occurrences	%	% Acum
1	33	40%	40%
2	21	26%	66%
3	8	10%	76%
4	3	4%	79%
5	7	9%	88%
6	1	1%	89%
7	4	5%	94%
8	3	4%	98%
9	0	0%	98%
10	0	0%	98%
11	0	0%	98%
12	1	1%	99%
13	1	1%	100%
14	0	0%	100%
15	0	0%	100%
16	0	0%	100%
17	0	0%	100%
18	0	0%	100%
19	0	0%	100%
20	0	0%	100%
20+	0	0%	100%

Figure 10: Experiments to determine the alpha value.

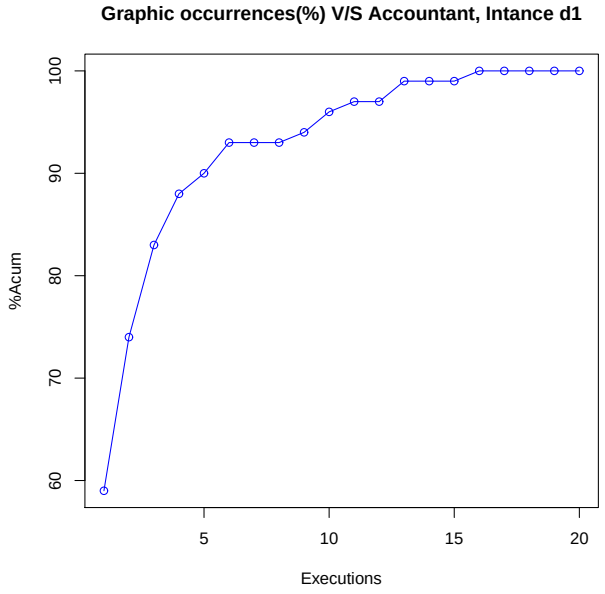


Figure 11: Chart of fitness change based on the number of iterations.

After a series of tests, we show the convergence for different initial velocities in Figure 12. When the velocity is low, the convergence shows to be premature. For this case, the initial velocity was adjusted, multiplying it by a β value in range $[0, 1]$.

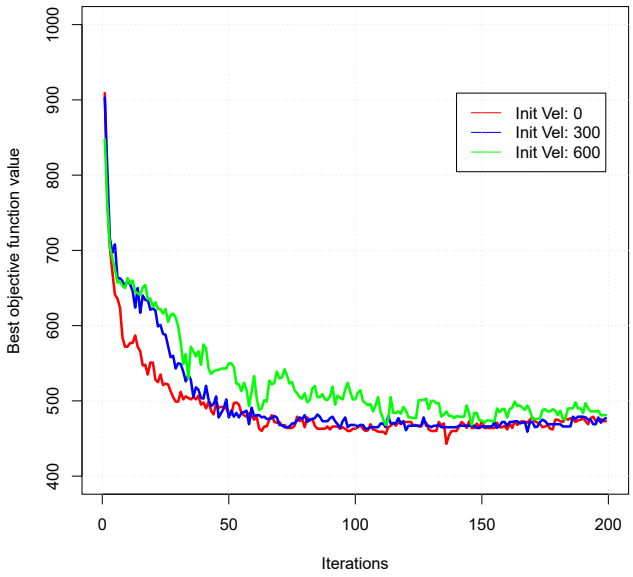


Figure 12: Convergence chart of different velocities.

5 Problem to solve

SCP is defined as a binary matrix (A) of size m-rows and n-columns, where $a_{i,j} \in \{0, 1\}$ is the value of each cell in the matrix A; i and j are of the size m-rows and n-columns, respectively:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n} \\ ... & ... & ... & ... \\ a_{m,1} & a_{m,2} & ... & a_{m,n} \end{pmatrix} \quad (15)$$

Defining column j satisfies a row i, if a_{ij} is equal to 1, which is a contrary case if a_{ij} is equal to 0. In addition, an associated cost $c \in C$ is incurred, where $C = \{c_1, c_2, ...c_n\}$, together with $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$, which are the sets of rows and columns, respectively.

The problem has an objective of minimizing the cost of the subset $S \subseteq J$, with the constraint that all rows $i \in I$ are covered by at least one column $j \in J$. When column j is in the subset of solution S, this is equal to 1 and is 0 otherwise.

The SCP can be defined as follows:

$$\text{Min} \quad Z = \sum_{j=1}^{n} c_j x_j \quad (16)$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \in I \quad (17)$$

$$x_j \in \{0, 1\} \quad \forall j \in J \quad (18)$$

6 Experiments

6.1 Benchmark Instances

In order to validate the proposed approach, the 9 sets of the Beasleys library were used, executing 55 instances that are known and that allow a finished experimental evaluation (Figure 8).

The proposed algorithm was built using Java Language and was executed on an Intel(R) Core(TM) i7-6700 CPU with a speed of 3.40 GHz, with 16GB of memory and using a 64-bit Windows 10 operating system.

6.2 Parameter Setting

As far as the configuration of the parameters used by this algorithm is concerned, IWD, as it is generally known, is a very difficult task that requires a great use of resources. Our approach takes care of working two of the most important parameters in IWD operation which are the initial soil and the initial speed, IntSoil and InitVelocity respectively. To verify the impact of these two parameters, a series of experiments with different values for the selected parameters were carried out. The idea is to adapt these two parameters dynamically in order to use the quality of the solution as an indicator of change. The action occurs if the best overall solution does not change after a defined number of iterations.

The parameters used for the different experiments are shown in the following Figure 13.

Parameter	Detail	Value
N	Amount of drops.	5000
$MAX\_ITERATION$	Number ofiterations.	50
$InitialSoil$	Initial soil value	1000
a_s	Soil parameter “a"	900000
b_s	Soil parameter “b"	0.01
c_s	Soil parameter “c"	1.0
$InitialVel$	Initial velocity value	15
a_v	Velocity parameter “a"	10.0
b_v	Velocity parameter “b"	0.01
c_v	Velocity parameter “c"	1.0

Figure 13: Experimental parameters.

6.3 Experimental results

The results obtained with the adaptive test are shown in the tables 14 to 22 and the comparison between the original and adaptive algorithm is shown in the Figure 23.-.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
4.1	429	440	467.18	2.56
4.2	512	551	612.88	7.62
4.3	516	545	614.84	5.62
4.4	494	510	565.88	3.24
4.5	512	533	586.84	4.10
4.6	560	589	576.26	5.18
4.7	430	451	492.28	4.88
4.8	492	517	564.26	5.08
4.9	641	728	809.2	13.57
4.10	514	540	581.4	5.06

Figure 14: Results of Group 4.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
5.1	253	270	301.1	6.72
5.2	302	333	366.36	10.26
5.3	226	234	254.72	3.54
5.4	242	259	278.38	7.02
5.5	211	222	245.22	5.21
5.6	213	234	246.56	9.86
5.7	293	319	353.2	8.87
5.8	288	309	340.4	7.29
5.9	279	293	320.42	5.02
5.10	265	288	306.88	8.68

Figure 15: Results of Group 5.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
6.1	138	149	176.96	7.97
6.2	146	162	202.66	10.96
6.3	145	149	200.1	2.76
6.4	131	136	157.26	3.82
6.5	161	183	218.34	13.66

Figure 16: Results of Group 6.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
A.1	253	287	334.74	13.44
A.2	252	276	330.3	9.52
A.3	232	251	296.4	8.19
A.4	234	277	315.48	18.38
A.5	236	255	295.64	8.05

Figure 17: Results of Group A.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
B.1	69	79	113.52	14.49
B.2	76	96	130.84	26.32
B.3	80	89	141.92	11.25
B.4	79	90	145.16	13.92
B.5	72	81	121.38	12.50

Figure 18: Results of Group B.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
C.1	227	258	347.48	13.66
C.2	219	248	344.98	13.24
C.3	243	278	371.94	14.40
C.4	219	258	335.32	17.81
C.5	215	245	346.7	13.95

Figure 19: Results of Group C.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
D.1	60	77	134.72	28.33
D.2	66	81	149.44	22.73
D.3	72	89	179	23.61
D.4	62	75	162.26	20.97

Figure 20: Results of Group D.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
E.1	29	44	91.84	51.72
E.2	30	49	137	63.33
E.3	27	41	104.42	51.85
E.4	28	37	103.5	32.14
E.5	28	39	110.76	39.29

Figure 21: Results of Group E.

Instances	Z_{BKS}	Z_{min}	Z_{avg}	RPD
F.1	14	21	50.68	50.00
F.2	15	22	39.3	46.67
F.3	14	21	40.38	50.00
F.4	14	19	38.78	35.71
F.5	13	17	42.8	30.77

Figure 22: Results of Group F.

6.4 Statistical Analysis

Figure 24 shows a general scheme of the existing statistical techniques, where in this study the statistical analysis included the different tests:

- Kolmogorov-Smirnov-Lilliefors [46] is used to establish the independence of the samples.
- Wilcoxon's Signed Rank [47] is used to verify that the IWD algorithm with AD is better than the Standard IWD algorithm.

Both statistical tests considered a significance level of 0.05, so that values lower than 0.05 indicate that the null hypothesis cannot be assumed, i.e., H_0 is rejected.

		IWD_{STD}			IWD_{AD}			
Intances	Z_{BKS}	Z_{min}	Z_{avg}	RPD	Z_{min}	Z_{avg}	RPD	$diff_{rpd}$
4.1	429	454	503.48	5.83	440	467.18	2.56	3.26
5.1	253	285	335.28	12.65	270	301.1	6.72	5.93
6.1	138	177	217.96	28.26	149	176.96	7.97	20.29
A.1	253	310	350.68	22.53	287	334.74	13.44	9.09
B.1	69	92	124.44	33.33	79	113.52	14.49	18.84
C1	227	297	351.74	30.84	258	347.48	13.66	17.18
D.1	60	87	146.9	45.00	77	134.72	28.33	16.67
E.1	29	43	92.22	48.28	44	91.84	51.72	-3.45
F.1	14	21	50.68	50.00	21	50.68	50.00	0.00

Figure 23: Results IWD_{STD} AND IWD_{AD}.

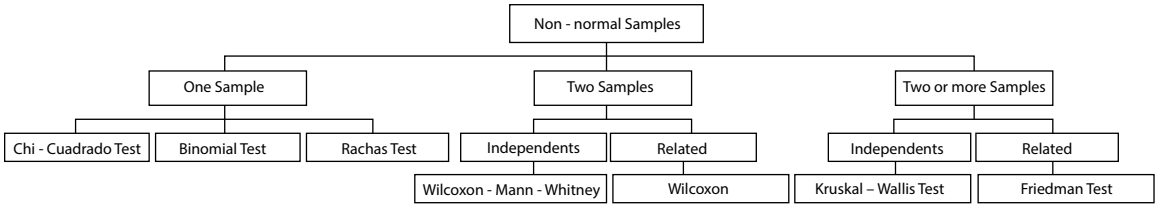


Figure 24: Statistical techniques.

For the determination of data independence the following hypothesis is used:

H_0 = The data follow a normal distribution.
H_1 = The data do not follow a normal distribution.

Given the P values obtained in the tests, the hypothesis is rejected. That is, the data do not follow a normal distribution.

H_0		IWD_{STD}	IWD_{AD}
4.1	IWD_{STD}	-	4.30^{-18}
	IWD_{AD}	>0.05	-
5.1	IWD_{STD}	-	6.45^{-18}
	IWD_{AD}	>0.05	-
6.1	IWD_{STD}	-	2.15^{-18}
	IWD_{AD}	>0.05	-
A.1	IWD_{STD}	-	2.15^{-18}
	IWD_{AD}	>0.05	-
B.1	IWD_{STD}	-	1.55^{-16}
	IWD_{AD}	>0.05	-
C.1	IWD_{STD}	-	9.88^{-16}
	IWD_{AD}	>0.05	-
D.1	IWD_{STD}	-	1.46^{-16}
	IWD_{AD}	>0.05	-
E.1	IWD_{STD}	-	>0.05
	IWD_{AD}	3.27^{-07}	-
F.1	IWD_{STD}	-	>0.05
	IWD_{AD}	1.33^{-09}	-

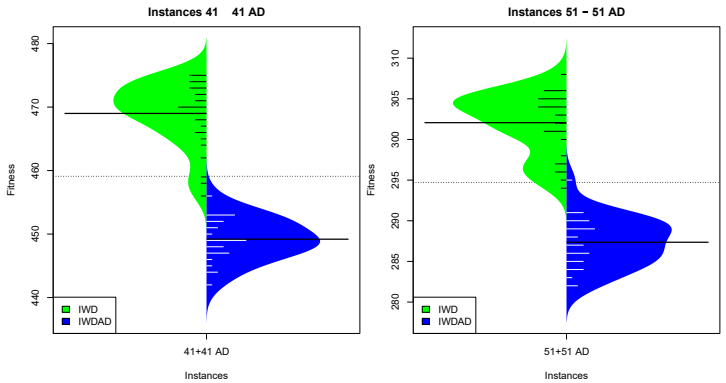
Figure 25: Statistical Analysis Results.

When it is obtained that the data does not follow a normal distribution, the Wilcoxon-Mann-Whitney [47] test is applied. This test is applied to verify that the version with AD is better, where the hypotheses are as follows:

H_0 = Standard IWD algorithm ≥ IWD algorithm with AD
H_1 = Standard IWD algorithm < IWD algorithm with AD

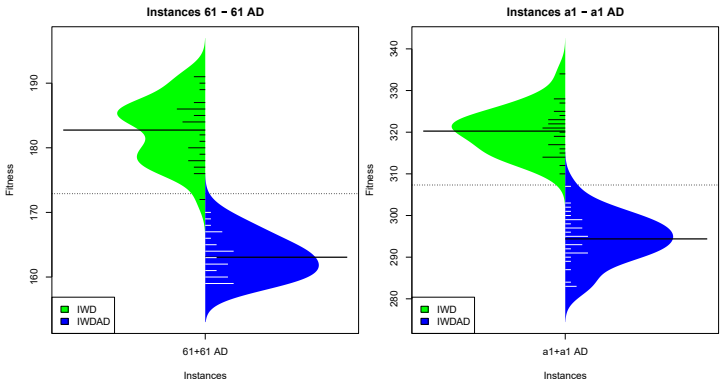
To obtain the p-values we use the programming language R, where obtaining a p-value < 0.05 implies rejecting H_0, and accepting H_1. The AD version is statistically better than the standard version, which extends to each instance of the benchmark (Figure 25. In addition, what is indicated by the statistical tests supports what has been obtained through verification by RPD.

The results of each instance can be seen graphically in the figure 26 to 30. IWD algorithm with AD has better results in all instances except for instances E and F.



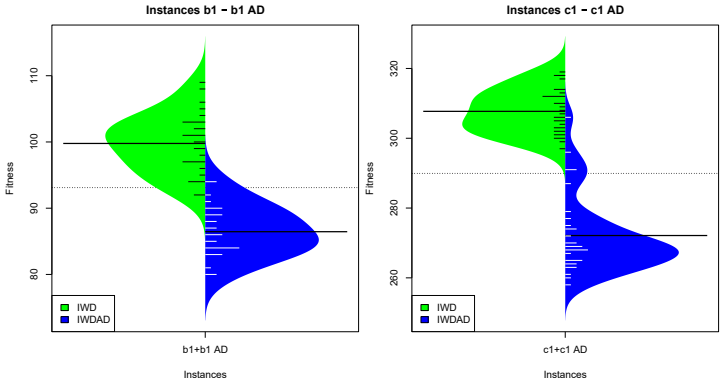
(a) Instance 4.1 (b) Instance 5.1

Figure 26: Instances 4.1 and 5.1.



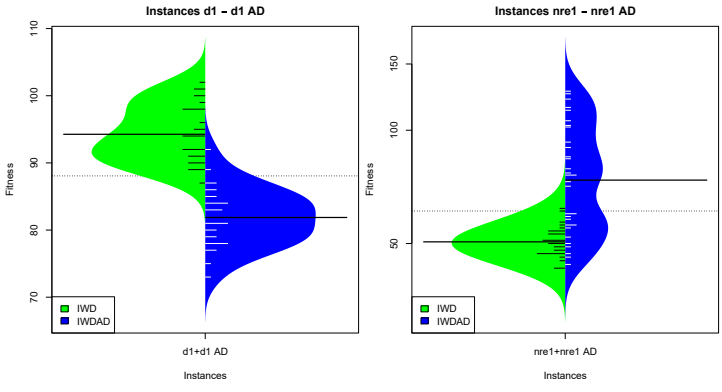
(a) Instance 6.1 (b) Instance A.1

Figure 27: Instances 6.1 and A.1.



(a) Instance B.1 (b) Instance C.1

Figure 28: Instances B.1 and C.1.



(a) Instance d.1 (b) Instance E.1

Figure 29: Instances D.1 and E.1.

7 Conclusion

In this work, we have presented an adaptive version of the Intelligent Water Drops algorithm, which is a tool that facilitates the use of metaheuristic techniques to non-expert users, reducing the difficulties in the configuration of parameters when implementing a metaheuristic in a real problem, a situation that will become increasingly common in the scenario of recovery from the economic crisis in post-pandemic times.

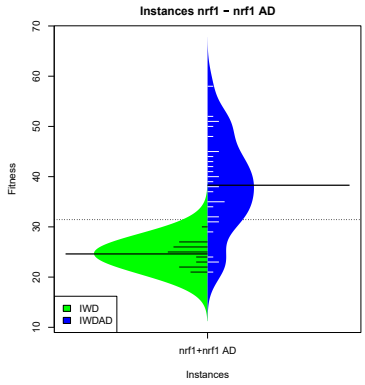


Figure 30: Instance F.1.

The adaptive element incorporated, obtains the necessary information to make decisions, based on external sources, coming from the problem and internal sources, typical of the normal functioning of the Intelligent Water Drops algorithm. With the correct combination of this information, it was possible to improve the balance of the exploration and exploitation of the search space of most of the solved instances, for Set Covering Problem, where the improvement was in average of 13.04%. In addition, the difference obtained by our adaptive proposal is significant in 77,78% of the compared instances, compared to the corresponding statistical comparisons.

The adaptive configuration is an interesting focus of future work, since there are a large number of static parameters that can be self-adjusting, in order to improve the performance of implementations, in order to decrease the influence of the off-line configuration in the performance when solving diverse instances. Along with this, it is necessary to replicate the implementation developed in this work, for other NP-Hard problems, validating the correct functioning of the incorporated adaptive components. Moreover, in the face of the growing line of research concerning the interaction between metaheuristics and machine learning techniques, a wide range of machine learning possibilities is presented in the decision making of the values to be taken by the adaptive parameters, adding to this, the option of choosing between different ways of calculating parameters according to internal and external information.

Conflict of Interest The authors declare no conflict of interest.

Acknowledgment Felipe Cisternas-Caneo and Marcelo Becerra-Rozas are supported by Grant DI Investigación Interdisciplinaria del Pregrado/VRIEA/PUCV/039.324/2020. Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1171243. Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. José Lemus-Romani is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2019-21191692.

References

[1] B. Crawford, R. Soto, G. Astorga, J. Lemus-Romani, S. Misra, J.-M. Rubio, "An Adaptive Intelligent Water Drops Algorithm for Set Covering Problem," in 2019 19th International Conference on Computational Science and Its Applications (ICCSA), 39–45, IEEE, 2019, doi:10.1109/ICCSA.2019.000-6.

[2] A. Aloi, B. Alonso, J. Benavente, R. Cordera, E. Echániz, F. González, C. Ladisa, R. Lezama-Romanelli, Á. López-Parra, V. Mazzei, et al., "Effects of the COVID-19 Lockdown on Urban Mobility: Empirical Evidence from the City of Santander (Spain)," Sustainability, **12**(9), 3870, 2020, doi:10.3390/su12093870.

[3] J. B. Sobieralski, "COVID-19 and airline employment: Insights from historical uncertainty shocks to the industry," Transportation Research Interdisciplinary Perspectives, 100123, 2020, doi:10.1016/j.trip.2020.100123.

[4] M. Sigala, "Tourism and COVID-19: impacts and implications for advancing and resetting industry and research," Journal of Business Research, 2020, doi:10.1016/j.jbusres.2020.06.015.

[5] F. Hao, Q. Xiao, K. Chon, "COVID-19 and China's Hotel Industry: Impacts, a Disaster Management Framework, and Post-Pandemic Agenda," International Journal of Hospitality Management, 102636, 2020, doi:10.1016/j.ijhm.2020.102636.

[6] T. Laing, "The economic impact of the Coronavirus 2019 (Covid-2019): Implications for the mining industry," The Extractive Industries and Society, 2020, doi:10.1016/j.exis.2020.04.003.

[7] D. Wang, X. Zhao, L. Shen, Z. Yang, "Industry choice for an airport economic zone by multi-objective optimization," Journal of Air Transport Management, **88**, 101872, 2020, doi:10.1016/j.jairtraman.2020.101872.

[8] L. Adacher, M. Flamini, M. Guaita, E. Romano, "A model to optimize the airport terminal departure operations," Transportation Research Procedia, **27**, 53–60, 2017, doi:10.1016/j.trpro.2017.12.151.

[9] N. A. Ribeiro, A. Jacquillat, A. P. Antunes, A. R. Odoni, J. P. Pita, "An optimization approach for airport slot allocation under IATA guidelines," Transportation Research Part B: Methodological, **112**, 132–156, 2018, doi:10.1016/j.trb.2018.04.005.

[10] C. Castro, B. Crawford, E. Monfroy, "A quantitative approach for the design of academic curricula," in Symposium on Human Interface and the Management of Information, 279–288, Springer, 2007, doi:10.1007/978-3-540-73354-6_31.

[11] C. Castro, B. Crawford, E. Monfroy, "A genetic local search algorithm for the multiple optimisation of the balanced academic curriculum problem," in International Conference on Multiple Criteria Decision Making, 824–832, Springer, 2009, doi:10.1007/978-3-642-02298-2_119.

[12] C. Blum, A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM computing surveys (CSUR), **35**(3), 268–308, 2003, doi:10.1145/937503.937505.

[13] C. Vásquez, B. Crawford, R. Soto, J. Lemus-Romani, G. Astorga, S. Misra, A. Salas-Fernández, J.-M. Rubio, "Galactic Swarm Optimization Applied to Reinforcement of Bridges by Conversion in Cable-Stayed Arch," in International Conference on Computational Science and Its Applications, 108–119, Springer, 2019, doi:10.1007/978-3-030-24308-1_10.

[14] B. Crawford, R. Soto, G. Astorga, J. Lemus, A. Salas-Fernández, "Self-configuring Intelligent Water Drops Algorithm for Software Project Scheduling Problem," in International Conference on Information Technology & Systems, 274–283, Springer, 2019, doi:10.1007/978-3-030-11890-7_27.

[15] R. Soto, B. Crawford, F. González, E. Vega, C. Castro, F. Paredes, "Solving the manufacturing cell design problem using human behavior-based algorithm supported by autonomous search," IEEE Access, **7**, 132228–132239, 2019, doi:10.1109/ACCESS.2019.2940012.

[16] S. Valdivia, R. Soto, B. Crawford, N. Caselli, F. Paredes, C. Castro, R. Olivares, "Clustering-Based Binarization Methods Applied to the Crow Search Algorithm for 0/1 Combinatorial Problems," Mathematics, **8**(7), 1070, 2020, doi:10.3390/math8071070.

[17] M. Birattari, J. Kacprzyk, Tuning metaheuristics: a machine learning perspective, volume 197, Springer, 2009, doi:10.1007/978-3-642-00483-4.

[18] Y. Hamadi, E. Monfroy, F. Saubion, "What is autonomous search?" in Hybrid Optimization, 357–391, Springer, 2011, doi:10.1007/978-1-4419-1644-0_11.

[19] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," Operations Research Perspectives, **3**, 43–58, 2016, doi:10.1016/j.orp.2016.09.002.

[20] F. Hutter, H. H. Hoos, K. Leyton-Brown, "Automated configuration of mixed integer programming solvers," in International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, 186–202, Springer, 2010, doi:10.1007/978-3-642-13520-0_23.

[21] C. Huang, Y. Li, X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," IEEE Transactions on Evolutionary Computation, **24**(2), 201–216, 2019, doi:10.1109/TEVC.2019.2921598.

[22] R. Battiti, M. Brunato, F. Mascia, Reactive search and intelligent optimization, volume 45, Springer Science & Business Media, 2008, doi:10.1007/978-0-387-09624-7.

[23] J. E. Beasley, "An algorithm for set covering problem," European Journal of Operational Research, **31**(1), 85–93, 1987, doi:10.1016/0377-2217(87)90141-X.

[24] H. Shah-Hosseini, "Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem," International Journal of Intelligent Computing and Cybernetics, **1**(2), 193–212, 2008, doi:10.1108/17563780810874717.

[25] O. Maron, A. W. Moore, "Hoeffding races: Accelerating model selection search for classification and function approximation," in Advances in neural information processing systems, 59–66, 1994.

[26] O. Maron, A. W. Moore, "The racing algorithm: Model selection for lazy learners," in Lazy learning, 193–225, Springer, 1997, doi:10.1023/A:1006556606079.

[27] M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, 11–18, Morgan Kaufmann Publishers Inc., 2002, doi:10.5555/2955491.2955494.

[28] M. Birattari, M. Dorigo, "The problem of tuning metaheuristics as seen from a machine learning perspective," 2004, doi:10.1007/978-3-642-00483-4.

[29] W. Conover, "Statistics of the Kolmogorov-Smirnov type," Practical nonparametric statistics, 428–473, 1999.

[30] T. Liao, T. Stützle, M. A. M. de Oca, M. Dorigo, "A unified ant colony optimization algorithm for continuous optimization," European Journal of Operational Research, **234**(3), 597–609, 2014, doi:10.1016/j.ejor.2013.10.024.

[31] F. Hutter, H. H. Hoos, T. Stützle, "Automatic algorithm configuration based on local search," in Aaai, volume 7, 1152–1157, 2007, doi:10.5555/1619797.1619831.

[32] F. Hutter, H. H. Hoos, K. Leyton-Brown, T. Stützle, "ParamILS: an automatic algorithm configuration framework," Journal of Artificial Intelligence Research, **36**, 267–306, 2009, doi:10.1613/jair.2861.

[33] P. Lin, J. Zhang, M. A. Contreras, "Automatically configuring ACO using multilevel ParamILS to solve transportation planning problems with underlying weighted networks," Swarm and Evolutionary Computation, **20**, 48–57, 2015, doi:10.1016/j.swevo.2014.10.006.

[34] G. Franceschini, S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," Chemical Engineering Science, **63**(19), 4846–4872, 2008, doi:10.1016/j.ces.2007.11.034.

[35] F. Hutter, H. H. Hoos, K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in International Conference on Learning and Intelligent Optimization, 507–523, Springer, 2011, doi:10.1007/978-3-642-25566-3_40.

[36] J. García, B. Crawford, R. Soto, C. Castro, F. Paredes, "A k-means binarization framework applied to multidimensional knapsack problem," Applied Intelligence, **48**(2), 357–380, 2018, doi:10.1007/s10489-017-0972-6.

[37] R. E. Mercer, J. Sampson, "Adaptive search using a reproductive meta-plan," Kybernetes, **7**(3), 215–228, 1978, doi:10.1108/eb005486.

[38] B. Crawford, R. Soto, E. Monfroy, G. Astorga, J. García, E. Cortes, "A meta-optimization approach for covering problems in facility location," in Workshop on Engineering Applications, 565–578, Springer, 2017, doi:10.1007/978-3-319-66963-2_50.

[39] X. Jin, T. Pukkala, F. Li, "Meta optimization of stand management with population-based methods," Canadian Journal of Forest Research, **48**(6), 697–708, 2018, doi:10.1139/cjfr-2017-0404.

[40] M. Camilleri, F. Neri, M. Papoutsidakis, "An algorithmic approach to parameter selection in machine learning using meta-optimization techniques," WSEAS Transactions on systems, **13**(2014), 202–213, 2014.

[41] X. Sun, C. Cai, S. Pan, Z. Zhang, Q. Li, "A cooperative target search method based on intelligent water drops algorithm," Computers & Electrical Engineering, **80**, 106494, 2019, doi:10.1016/j.compeleceng.2019.106494.

[42] M. Adhikari, T. Amgoth, "An intelligent water drops-based workflow scheduling for IaaS cloud," Applied Soft Computing, **77**, 547–566, 2019, doi:10.1016/j.asoc.2019.02.004.

[43] E. Teymourian, V. Kayvanfar, G. M. Komaki, M. Zandieh, "Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem," Information Sciences, **334**, 354–378, 2016, doi:10.1016/j.ins.2015.11.036.

[44] H. Duan, S. Liu, J. Wu, "Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning," Aerospace science and technology, **13**(8), 442–449, 2009, doi:10.1016/j.ast.2009.07.002.

[45] D. R. Prabha, T. Jayabarathi, R. Umamageswari, S. Saranya, "Optimal location and sizing of distributed generation unit using intelligent water drop algorithm," Sustainable Energy Technologies and Assessments, **11**, 106–113, 2015, doi:10.1016/j.seta.2015.07.003.

[46] S. S. Shapiro, M. B. Wilk, "An analysis of variance test for normality (complete samples)," Biometrika, **52**(3/4), 591–611, 1965, doi:10.2307/2333709.

[47] H. B. Mann, D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," The annals of mathematical statistics, 50–60, 1947, doi:10.1214/aoms/1177730491.