Reinforcement Learning Based Whale Optimizer

Marcelo Becerra-Rozas[1(✉)], José Lemus-Romani[4], Broderick Crawford[1], Ricardo Soto[1], Felipe Cisternas-Caneo[1], Andrés Trujillo Embry[1], Máximo Arnao Molina[1], Diego Tapia[1], Mauricio Castillo[1], Sanjay Misra[2], and José-Miguel Rubio[3]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{marcelo.becerra.r,felipe.cisternas.c,andres.trujillo.e,
maximo.arnao.m,diego.tapia.r,mauricio.castillo.d}@mail.pucv.cl,
{broderick.crawford,ricardo.soto}@pucv.cl
[2] Covenant University, Ota, Nigeria
sanjay.misra@covenantuniversity.edu.ng
[3] Universidad Bernardo O'Higgins, Santiago, Chile
josemiguel.rubio@ubo.cl
[4] School of Civil Construction, Pontificia Universidad Católica de Chile, Santiago, Chile
jose.lemus@uc.cl

Abstract. This work proposes a Reinforcement Learning based optimizer integrating SARSA and Whale Optimization Algorithm. SARSA determines the binarization operator required during the metaheuristic process. The hybrid instance is applied to solve benchmarks of the Set Covering Problem and it is compared with a Q-learning version, showing good results in terms of fitness, specifically, SARSA beats its Q-Learning version in 44 out of 45 instances evaluated. It is worth mentioning that the only instance where it does not win is a tie. Finally, thanks to graphs presented in our results analysis we can observe that not only does it obtain good results, it also obtains a correct exploration and exploitation balance as presented in the referenced literature.

Keywords: Metaheuristic · SARSA · Q-Learning · Swarm intelligence · Whale optimization algorithm · Combinatorial optimization

1 Introduction

The process of selecting the optimal solution from a collection of candidate solutions that satisfy all of the constraints of an optimization issue is referred to as optimization.

Beyond this, we find combinatorial optimization, where solutions are represented in variables that are in discrete domains. That is, the solutions can be subsets, permutations, graphs or integers, and have no polynomial solution. To

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12957, pp. 205–219, 2021.
https://doi.org/10.1007/978-3-030-87013-3_16

obtain good quality solutions in reasonable times at the expense of the guarantee of finding the optimum, approximate methods are used given the complexity of solving combinatorial optimization problems.

A metaheuristic algorithm is a framework with numerous high-level purposes that can be used to construct heuristics or heuristic optimization algorithms using a variety of different tactics [11]. Various academics have employed metaheuristics to address a wide range of combinatorial optimization problems over the years. These algorithms have a distinguishing quality in that they can be employed and adapted to a variety of optimization problems in the same domain as the metaheuristic or even in distinct domains from the domain in which the metaheuristic operates.

Reinforcement learning is a class of methods that aim to try to find an optimal policy for complicated systems or agents [13,23,25]. Being efficient methods, they have been implemented in various areas or applications: model-free control, optimal control [26], ambidextrous metaheuristics [2,3,6,17,18]. Reinforcement learning can be divided according to the way it evaluates the policy: off-policy and on-policy. The most common reinforcement learning techniques include Q-Learning, SARSA, Temporal Difference(λ) [13].

In this paper, we present a general framework incorporating SARSA to determine a specific action: the selection of binarization schemes when solving binary domain problems with continuous swarm-based metaheuristic algorithms. SARSA is used to determine the selection of binarization schemes in binary domain problems. After evaluating 45 instances of the Set Covering Problem, it can be determined that the proposed implementation with SARSA outperforms the one presented in [2], where the binarization selector is Q-Learning by a statistically significant margin.

Some sections of the paper are modeled after the work of [9], and it is organized as follows: Sect. 2 presents related work, discussing swarm-based algorithms and how reinforcement learning has supported metaheuristics. Section 3 presents the Q-Learning and SARSA techniques with their respective reward function. Section 4 mentions the classical Whale optimization algorithm and then presents the proposal of a new Whale algorithm with the implementation of SARSA. Finally, the obtained results are evaluated and analyzed and a conclusion is drawn in Sects. 5 and 6 respectively.

2 Related Work

2.1 Swarm-Based Algorithms

Inspired often from nature and the collective behavior of biological systems, the swarm intelligence algorithms, corresponds to a group of algorithms that are based on the study of self-organized and distributed systems. Those systems usually consist of a population of agents with a limited individual capacity to perceiving and modifying their local environment. Such ability makes communication between individuals possible, detecting changes in the environment generated by the behavior of their peers. Local interactions between agents usually

lead to the emergence of global behavior, which allows to the agents solve complex problems. Local interactions between agents often result in the formation of global behavior, which allows the agents to handle complicated problems successfully. Grey Wolf Optimization (GWO), Harris Hawk Optimization (HHO), Moth-Flame Optimization (MFO), Social Spider Optimization (SSO), Cuckoo Search (CS). [15] has further information about the Swarm Intelligence Algorithm and how it works.

Metaheuristics have a variety of elements that vary according to the metaphor they reflect. Metaheuristics are comprised mostly of the following elements: population, local search, instance, operators, parameters, evaluation, initialization, and decision variables [11]. Numerous experiments are required to calibrate the values of these parameters, which takes a significant amount of time and creates an imbalance between the exploitation and exploration of metaheuristics. As a result, the metaheuristics require incorporated dynamic features that can be changed as iterations go. SARSA and Q-Learning are utilized in this article to execute a dynamic selection of operators.

2.2 Reinforcement Learning into Metaheuristics

While metaheuristics have been proved to be quite beneficial in addressing difficult optimization problems [6,20–22], it takes a long time and there is no guarantee that the approach will converge to an optimal solution; in fact, the method is frequently prone to falling into a local optimum. To address this issue, machine learning techniques were applied to shorten search durations and/or produce higher-quality results [16]. In [11], Song et al. offer a classification in which Machine Learning (ML) approaches assist metaheuristics.

The following are some of the reasons why hybridization is advantageous, which we consider noteworthy:

- It causes the metaheuristic to be adaptive, which allows the algorithm to apply to different problems.
- It does not require complete information about the problem since reinforcement learning models learn by collecting experience [14].
- By using independent learning agents [12], it allows in some cases, the computational cost to be lower. Since it uses a single update formula at each step.
- If general features are used, the information learned by the reinforcement learning can be used in other parts of the same problem [19].
- The behavior of various reinforcement learning methods end in optimal state-action pairs [13], which can be exploited. As an example of this, one can see how the policy choices the next action evolves at each step.

3 Reinforcement Learning Methods

Since in reinforcement learning the agent's goal is the maximization of the value function, at time t the agent chooses the action that maximizes the expected

value of the total rewards it can obtain in the future from the state the agent is in. The expected reward R_t is generally defined as a function of the current and discounted future rewards. In general, we define the future reward from the time step t in Eq. (1).

$$R_t = \sum_{j=0}^{n} \gamma^j \cdot r_{t+j+1} \tag{1}$$

Where $\gamma \in [0, 1]$ is the discount factor, r_t is the reward when an action is taken at time t and n is often regarded as the time when the process terminates. Therefore, the agent's goal is to learn a policy capable of maximizing long-run rewards by interacting with the environment based on one's own experience. To do this, at each step t, starting from the state s_t, the agent has to compute as follows the value of the action-value function: $Q^{\pi}(s, a)$ for each possible action a_t on the basis of the policy π. The policy π can be defined as in Eq. (2):

$$Q^{\pi}(s, a) = \mathbb{P}_{\pi}\{R_t \mid s_t = s, a_t = a\} \tag{2}$$

where $\mathbb{P}_{\pi}\{R_t \mid s_t = s, a_t = a\}$. Now the agent aims at obtaining the optimal state-action function Q*(s, a) which is usually relevant to the Bellman equation. Then two methods called Q-learning and SARSA will be compared to get the optimal state-action value function.

3.1 Q-Learning

Q-Learning is one of the most traditional reinforcement learning algorithms in existence [2], Q-learning is an off-policy method, in other words, the agent learns and selects action a independently of interaction with the environment. The impact of executing action a on the environment is obtained by reward or punishment (r) and this allows to decide which is the next state s_{t+1}. Therefore, the main objective is to maximize the function $Q(s, a)$ during the learning process to obtain the best action for a particular state. The way to mathematically represent the updating equation is Eq. (3):

$$Q(s_t, a_t) \longleftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot maxQ(s_{t+1}, a_{t+1})] \tag{3}$$

where α represents the learning rate, γ is the discount factor, r is the immediate reward or penalty received and $maxQ(s_{t+1}, a_{t+1})$ tells us that a_{t+1} is the best action for state s_{t+1}.

3.2 SARSA

Instead, SARSA is an on-policy method [13], the agent learns the value of the state-action pair based on the performed action, in other words, when the value of the current state-action is updated, the next action a_{t+1} will be taken. While

in Q-Learning, the action a_{t+1} is completely greedy. Based on this, the state-action value update equation is defined as in Eq. (4):

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{4}$$

The procedure of forming the Q-table is the same for both algorithms. The only difference between them is the update rule that is being followed in every step, Eqs. (3) and (4). The different update rule enables SARSA to learn faster than the Q-learning algorithm. However, this makes SARSA a more conservative algorithm and the probability of finding the optimal policy is higher for Q-learning.

3.3 Reward Function

The big question when using reinforcement learning methods is: how to reward or punish the actions performed by the agent. The balance between reward and punishment achieves an equal variety of the selection of actions so that the best action found is more reliable.

Different learnheuristics have been found in the literature in which metaheuristics incorporate reinforcement learning techniques as a machine learning technique. The classical reward function used by these learnheuristics is adapted to the behavior of the metaheuristic.

For example, we will use a simplified version of the version proposed by Yue Xu and Dechang Pi [24] for optimal topology selection in particle swarm optimization. The simplified performance-oriented version of the metaheuristic considers reward value +1 when fitness is improved or 0 otherwise. As a result, the reward or penalty visible in Eq. (5) and is born where only the reward is given. The type of reward mentioned above is shown in Table 1.

Table 1. Types of rewards

Reference	Reward Function
[24]	$\mathrm{r}_n = \begin{cases} +1, & if\ the\ current\ action\ improves\ fitness \\ 0, & otherwise. \end{cases}$ (5)

4 Whale Optimization Algorithm

Whale Optimization Algorithm (WOA) is inspired by the hunting behavior of humpback whales, specifically, how they make use of a strategy known as "bubble netting". This strategy consists of locating the prey, and employing a spiral turn like a "9", enclosing the prey. This algorithm was invented by Mirjalili and Lewis in 2015 [8].

The WOA metaheuristic starts with a set of random solutions. At each iteration, the search agents update their positions concerning a randomly chosen search agent or the best solution obtained so far. There is a parameter "a" that is reduced from 2 to 0 to provide changes between exploration and exploitation. When the equation vector (6) has value: $\mid \overrightarrow{A} \mid \geq 1$ a new random search agent is chosen, while when $\mid \overrightarrow{A} \mid \leq 1$ the best solution is selected, all this in order to be able to update the position of the search agents.

On the other hand, the value of the parameter "p" allows the algorithm to switch between a spiral or circular motion. To assimilate this, three movements are crucial when working with the metaheuristic:

1. **Searching for prey:** The whales search for prey randomly based on the position of each prey. When the algorithm determines that $\mid \overrightarrow{A} \mid \geq 1$, then we can say that it is exploring and allows WOA to perform a global search. We represent this first move with the following mathematical model:

$$\begin{aligned} \overrightarrow{X}_i^{t+1} &= \overrightarrow{X_{rand}^t} - \overrightarrow{A} \cdot \overrightarrow{D} \\ \overrightarrow{D} &= |\overrightarrow{C} \cdot \overrightarrow{X_{rand}^t} - \overrightarrow{X}_i^t| \end{aligned} \tag{6}$$

   Where t denotes the current iteration, $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, $\overrightarrow{X_{rand}}$ is a random position vector (i.e., a random whale) chosen from the current population. The vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ can be computed according to Eq. (7):

$$\begin{aligned} \overrightarrow{A} &= 2\overrightarrow{a} \cdot \overrightarrow{r} - \overrightarrow{a} \\ \overrightarrow{C} &= 2 \cdot \overrightarrow{r} \end{aligned} \tag{7}$$

   Where, $\overrightarrow{a}$ decreases linearly from 2 to 0 over iterations (both in the exploration and exploitation phases) and $\overrightarrow{r}$ corresponds to a random vector of values between $[0, 1]$.
2. **Encircling the prey:** Once the whales have found and recognized their prey, they begin to encircle them. Since the position of the optimal design in the search space is not known in the first instance, the metaheuristic assumes that the current best solution is the target prey or is close to the optimum. Therefore, once the best search agent is defined, the other agents will attempt to update their positions toward the best search agent. Mathematically it is modeled as in Eq. (8):

$$\begin{aligned} \overrightarrow{X}_i^{t+1} &= \overrightarrow{X^*}_i^t - \overrightarrow{A} \cdot \overrightarrow{D} \\ \overrightarrow{D} &= |\overrightarrow{C} \cdot \overrightarrow{X^*}_i^t - \overrightarrow{X}_i^t| \end{aligned} \tag{8}$$

   Where $\overrightarrow{X^*}$ is the position vector of the best solution obtained so far and $\overrightarrow{X}$ is the position vector. The vector $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated as in Eq. (7). It is worth mentioning that $\overrightarrow{X}$ must be updated at each iteration if a better solution exists.

3. **Bubble net attack:** For this attack, the "shrinking net mechanism" is presented, this behavior is achieved by decreasing the value of a in the Eq. (7). Thus, as the whale spirals, it shrinks the bubble net until it finally catches the prey. This motion is modeled with the Eq. (9):

$$\begin{aligned} \overrightarrow{X}_i^{t+1} &= \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}_i^t \\ \overrightarrow{D'} &= |\overrightarrow{X^*}_i^t - \overrightarrow{X}_i^t| \end{aligned} \tag{9}$$

Where, $\overrightarrow{D'}$ is the distance of the i-th whale from the prey (the best solution obtained so far, b is a constant to define the shape of the logarithmic spiral. l is a random number between $[-1, 1]$.

It is worth mentioning that humpback whales swim around the prey within a shrinking circle and along a spiral trajectory simultaneously. In order to model this simultaneous behavior, there is a 50% probability of choosing between the encircling prey mechanism (2) or the spiral model (3) to update the position of the whales during optimization. The mathematical model is as follows:

$$\overrightarrow{X}_i^{t+1} = \begin{cases} \overrightarrow{X^*}_i^t - \overrightarrow{A} \cdot \overrightarrow{D} & \text{If } p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}_i^t & \text{If } p \geq 0.5 \end{cases} \tag{10}$$

Figure 1 shows the behavior of the metaheuristic to improve the understanding of the above.

4.1 Ambidextrous Metaheuristics

Recent studies [2,17,18] built a general framework of ambidextrous metaheuristics [3,6] that incorporates Q-Learning as a reinforcement learning module for operator selection. Specifically, they select binarization schemes derived from combinations between transfer functions and discretization functions of the Two-Step Technique.

In this paper we present modifications of the classical whale [8], where both, SARSA and Q-Learning will be incorporated for later comparison. The **actions** to be taken by the agents are the **binarization schemes**, the **states** are the **phases of the metaheuristic**, i.e. exploration or exploitation, the **episodes** where an action is selected and applied in a particular state will be the **iterations** and the **agents** will be the **individuals** of the whale algorithm.

4.2 Balancing Exploration and Exploitation

During the search process, metaheuristics make decisions to discern whether to explore the search space or exploit promising regions. The decision criteria are specific to each metaheuristic and generating a good tuning allows obtaining a better balance. Ambidextrous algorithms were designed to balance exploration

and exploitation from the point of view of decision-making. Before making a decision that affects the balancing, it is necessary to know in which phase the metaheuristic is.

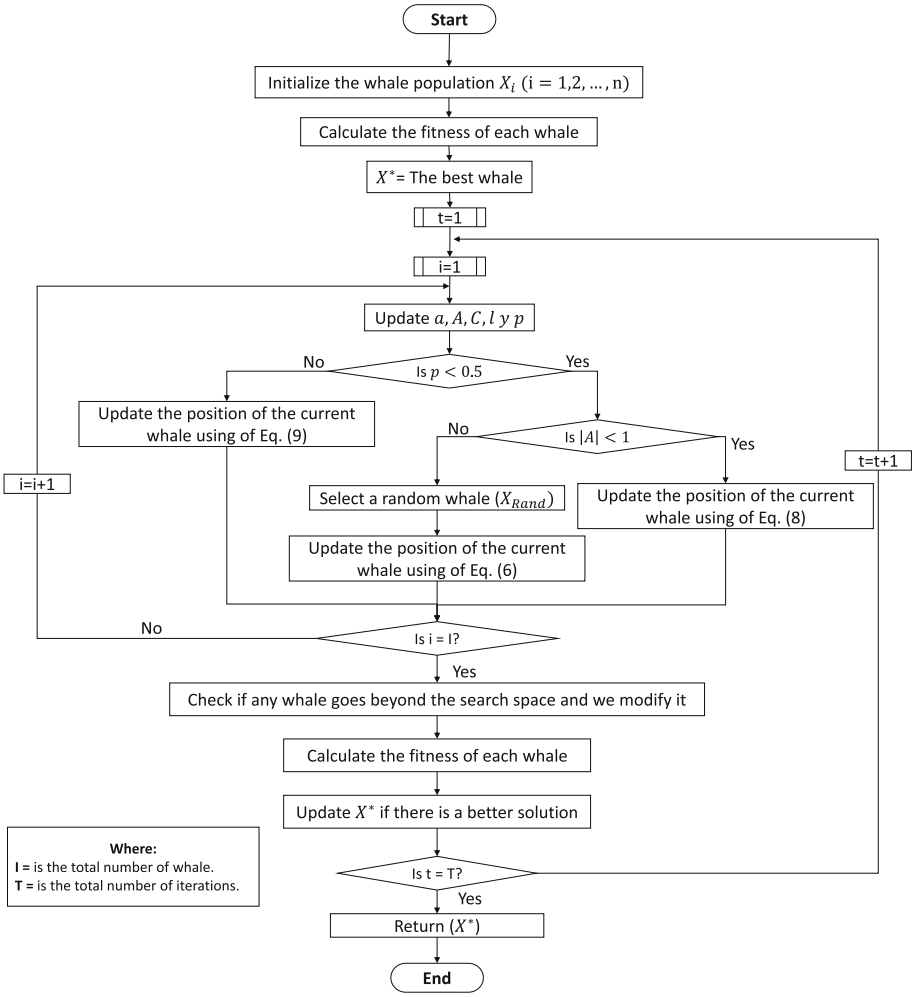

Fig. 1. Whale optimization algorithm

One of the techniques to measure the phase of a population metaheuristic is the diversity metrics. These metrics quantify diversity by the distances between individuals where distances increase during exploratory phases and decrease during exploitative phases. In the literature, there are different diversity metrics such

as Dimensional Diversity-Hussain [4]. Dimensional-Hussain Diversity is defined as:

$$Div = \frac{1}{l \cdot n} \sum_{d=1}^{l} \sum_{i=1}^{n} |\bar{x}^d - x_i^d|, \tag{11}$$

Where X is the population under analysis, $\bar{x}^d$ is the mean of the d-th dimension, n is the number of individuals in the population X and l is the number of decision variables of the optimization problem.

Morales-Castañeda et al. in [10] propose equations that use the diversity obtained from a population to calculate an estimate of the exploration (XPL%) and exploitation (XPT%) phases using percentages. The equations mentioned are as follows:

$$XPL\% = \frac{Div}{Div_{max}} \cdot 100, \tag{12}$$

and

$$XPT\% = \frac{|Div - Div_{max}|}{Div_{max}} \cdot 100. \tag{13}$$

Where Div_t corresponds to the current diversity in the t-th iteration and Div_{max} corresponds to the maximum diversity obtained in the entire search process.

These exploration and exploitation percentages can be used to roughly determine the stage of the metaheuristic as follows:

$$Phase = \begin{cases} Exploration\ if\ XPL\% \geq XPT\% \\ Exploitation\ if\ XPL\% < XPT\% \end{cases} \tag{14}$$

This metaheuristic phase estimation is used for state determination in the Q-Learning and SARSA algorithms. Thus, their state transition is determined by the diversity of individuals during the search process. The proposal of this work is shown in Fig. 2.

4.3 A New Algorithm: Binary SARSA Whale Optimization Algorithm

Under the proposed implementation, a new algorithm based on Whale Optimization Algorithm is created, together with the incorporation of the reinforcement learning algorithm SARSA, which acts as the agent in charge of selecting the best binarization scheme to be used in each iteration according to what is learned during the iterative process of the metaheuristic.

The hybridization between the metaheuristic and the reinforcement learning algorithm SARSA is explained in the Fig. 2. Where the red boxes indicate the participation of the SARSA algorithm.

5 Experimental Results

Experiments solving the Set Covering Problem with Beasley's OR-Library instances totaled 45 instances. These instances were executed with a total of 40 population and 1000 iterations, having a total of 40,000 calls to the objective function, as used in [5]. The implementation was developed in Python 3.8.5 and processed using the free Google Colaboraty service [1]. The parameter settings for the SARSA and Q-Learning algorithm have been as follows: $\gamma = 0.4$ and $\alpha = 0.1$.

The results obtained are presented in the Table 2, which in its first column presents the name of each solved instance, in the second column, the optimal value of each instance. While the next 6 columns present the best results (Best), average results (Avg), and RPD according to Eq. 15 for each of the instances and both versions: BSWOA and Binary Q-WOA (BQWOA). The version of WOA hybridized with Q-Learning has been implemented according to the work of Cisternas et al. [2]. Finally, the last two rows of the table present the average values of all instances and the p-value obtained by the Wilcoxon-Mann-Whitney test [7]. The test allows us to determine whether the results obtained are significantly different to determine which of the two hybridized versions performs better on this subset of instances.

$$\text{RPD} = \frac{100 \cdot (Best - Opt)}{Opt}. \tag{15}$$

On the other hand, the exploration and exploitation graphs are presented as presented in Sect. 4.2, obtaining Fig. 3 and 4. Where we can observe that behaviors similar to those presented by Morales-Castañeda et al. [10] is presented as a correct balance of exploration and exploitation, starting with a diversification of the solutions in the search space to subsequently intensify the search.

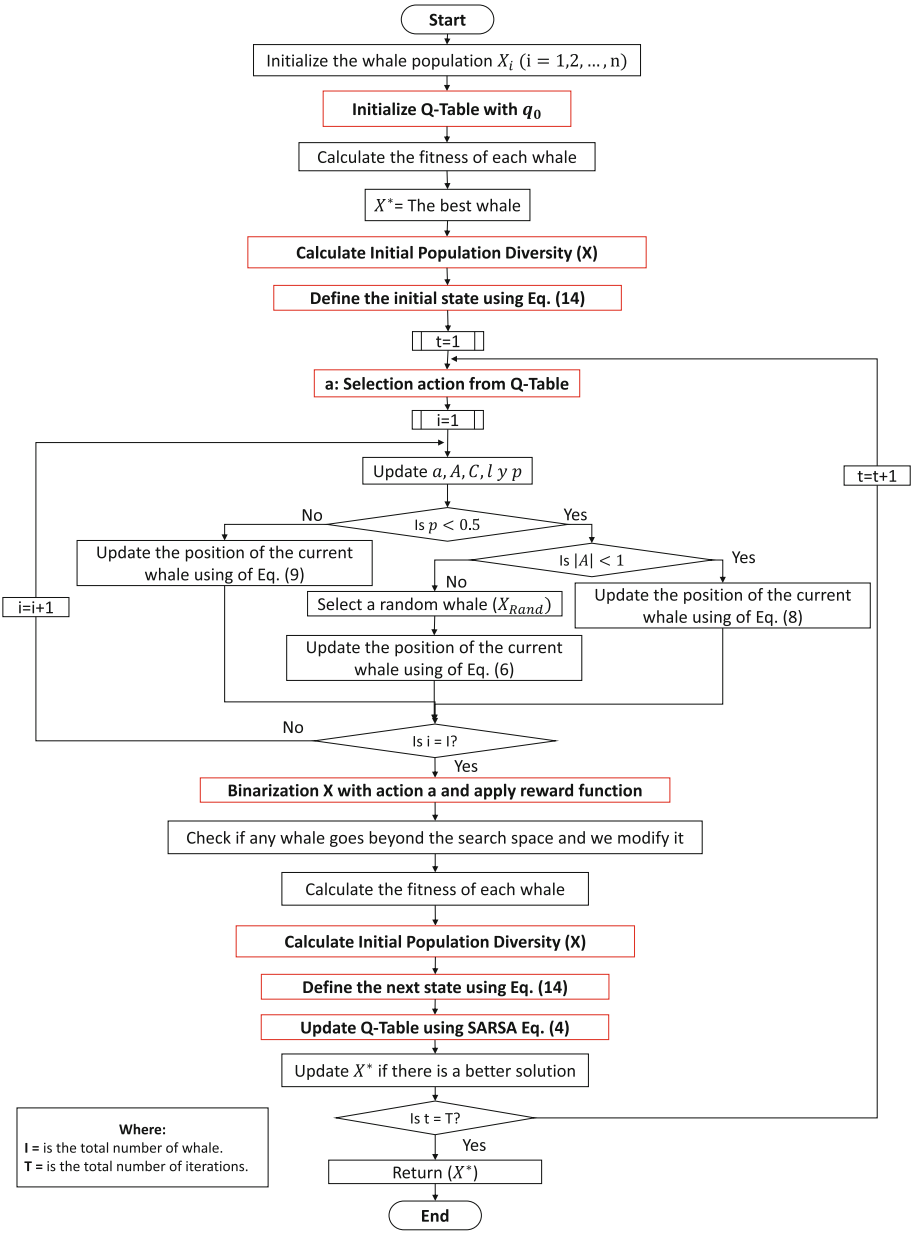

Fig. 2. Binary S-Whale optimization algorithm (Color figure online)

Table 2. Results obtained by BSWOA and BQWOA solving SCP

Inst.	Opt.	BSWOA			BQWOA		
		Best	Avg	RPD	Best	Avg	RPD
4.1	429	**430**	434.6	0.23	435	439.48	1.4
4.2	512	**523**	535.6	2.15	538	546.44	5.08
4.3	516	**525**	532.7	1.74	537	543.78	4.07
4.4	494	**497**	508.0	0.61	519	526.33	5.06
4.5	512	**523**	528.4	2.15	537	541.89	4.88
4.6	560	**567**	570.3	1.25	573	580.33	2.32
4.7	430	**435**	439.5	1.16	440	445.29	2.33
4.8	492	**496**	499.6	0.81	505	507.83	2.64
4.9	641	**671**	677.6	4.68	686	690.8	7.02
4.10	514	**521**	524.1	1.36	530	532.4	3.11
5.1	253	**258**	262.7	1.98	262	267.71	3.56
5.2	302	**319**	325.3	5.63	326	332.17	7.95
5.3	226	**230**	230.8	1.77	232	233.5	2.65
5.4	242	**247**	249.4	2.07	250	252.5	3.31
5.5	211	**212**	214.5	0.47	216	218.83	2.37
5.6	213	**218**	221.3	2.35	227	229.0	6.57
5.7	293	**302**	304.6	3.07	311	313.2	6.14
5.8	288	**291**	293.9	1.04	298	299.33	3.47
5.9	279	**282**	284.3	1.08	284	287.4	1.79
5.10	265	**267**	273.1	0.75	277	278.33	4.53
6.1	138	**141**	144.1	2.17	144	146.68	4.35
6.2	146	**147**	152.3	0.68	154	155.83	5.48
6.3	145	**147**	148.4	1.38	149	150.4	2.76
6.4	131	**131**	133.1	0.0	132	134.17	0.76
6.5	161	**163**	172.2	1.24	180	181.5	11.8
A.1	253	**260**	263.2	2.77	263	266.84	3.95
A.2	252	**261**	264.0	3.57	266	269.83	5.56
A.3	232	**240**	243.4	3.45	244	245.6	5.17
A.4	234	**238**	242.5	1.71	251	251.8	7.26
A.5	236	**241**	244.2	2.12	242	247.33	2.54
B.1	69	**69**	70.5	0.0	70	71.68	1.45
B.2	76	**76**	77.2	0.0	78	79.5	2.63
B.3	80	**81**	81.7	1.25	82	82.17	2.5
B.4	79	**79**	81.3	0.0	83	83.83	5.06
B.5	72	**72**	72.9	0.0	73	74.33	1.39
C.1	227	**234**	238.4	3.08	243	247.81	7.05
C.2	219	**229**	232.3	4.57	234	238.83	6.85
C.3	243	**249**	254.2	2.47	258	260.83	6.17
C.4	219	**227**	229.0	3.65	232	233.83	5.94
C.5	215	**221**	225.4	2.79	229	231.33	6.51
D.1	60	**61**	62.3	1.67	63	64.97	5.0
D.2	66	**67**	67.4	1.52	68	69.0	3.03
D.3	72	**73**	74.8	1.39	76	77.33	5.56
D.4	62	**62**	62.2	0.0	**62**	63.4	0.0
D.5	61	**61**	62.6	0.0	63	64.33	3.28
		258.76	262.44	1.73	264.93	267.99	4.27
p-value							0.00

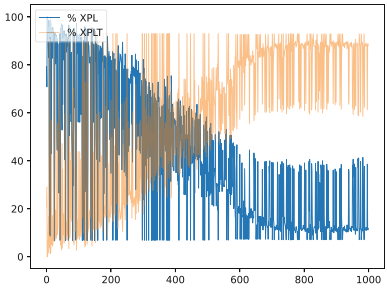

Fig. 3. Exploration and exploitation graphic - SARSA

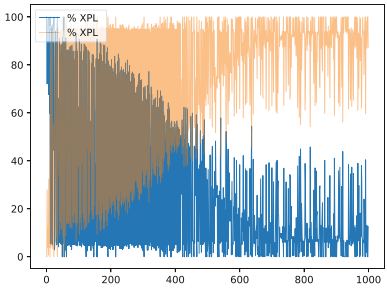

Fig. 4. Exploration and exploitation graphic - Q-Learning

6 Conclusion

The implementation of reinforcement learning techniques to metaheuristics has presented a great contribution both to the improvement of fitness obtained and to obtain a better exploration-exploitation balance. In this work, SARSA has been implemented as a binarization scheme selector in the metaheuristic Whale Optimization Algorithm, solving 45 OR-Library instances of the Set Covering Problem. Where when compared with its closest version BQWOA, it has been shown that BSWOA obtains better quality results in 44 out of 45 instances and that these results are significantly better under the Wilcoxon-Mann-Whitney Test.

The proposal achieved the optimum in 7 instances (Instances 6.4, B.1, B.2, B.4, B.5, D.4, and D.5) versus 1 optimum of the version incorporating Q-Learning (Instance D.4). As for the average RPD, the proposal achieves a value of 1.73 among all the instances executed, indicating a good performance by obtaining results that are not so far from the optimal ones.

The results obtained are promising since by implementing a binarization scheme selector, tuning times are reduced by not having to evaluate the combinations of the different binarization schemes present in the literature, providing the implementation of these techniques.

On the other hand, when observing in detail the exploration and exploitation graphs, it is observed that both have similar convergences, but in the case of BSWOA, the values tend to have variations of smaller magnitude and with the lower occurrence, which could be an indicator of its better performance in solving this problem, possibly having movements of smaller magnitude within the search space, but this statement must be validated by an analysis of the solution vectors during the iterative process.

As future work, along with the implementation of SARSA in other metaheuristic techniques, there is a need to be able to parameterize the results obtained in the exploration and exploitation graphs, although it gives us valuable information of the search process, a metric for comparison is still needed, along with the option of incorporating this same metric to the learning process of the reinforcement learning agent.

Acknowledgements. Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1210810. 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. Marcelo Becerra-Rozas is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL/2021-21210740.

References

1. Bisong, E.: Google colaboratory. In: Bisong, E. (ed.) Building Machine Learning and Deep Learning Models on Google Cloud Platform, pp. 59–64. Springer, Heidelberg (2019). https://doi.org/10.1007/978-1-4842-4470-8_7
2. Cisternas-Caneo, F., et al.: A data-driven dynamic discretization framework to solve combinatorial problems using continuous metaheuristics. In: Abraham, A., Sasaki, H., Rios, R., Gandhi, N., Singh, U., Ma, K. (eds.) IBICA 2020. AISC, vol. 1372, pp. 76–85. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73603-3_7
3. Crawford, B., León de la Barra, C.: Los algoritmos ambidiestros (2020). https://www.mercuriovalpo.cl/impresa/2020/07/13/full/cuerpo-principal/15/. Acceded 12 Feb 2021
4. Hussain, K., Zhu, W., Salleh, M.N.M.: Long-term memory Harris' hawk optimization for high dimensional and optimal power flow problems. IEEE Access **7**, 147596–147616 (2019)
5. Lanza-Gutierrez, J.M., Crawford, B., Soto, R., Berrios, N., Gomez-Pulido, J.A., Paredes, F.: Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization. Expert Syst. Appl. **70**, 67–82 (2017)
6. Lemus-Romani, J., et al.: Ambidextrous socio-cultural algorithms. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12254, pp. 923–938. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58817-5_65
7. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. 50–60 (1947)
8. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
9. Misra, S.: A step by step guide for choosing project topics and writing research papers in ICT related disciplines. In: ICTA 2020. CCIS, vol. 1350, pp. 727–744. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-69143-1_55
10. Morales-Castañeda, B., Zaldivar, D., Cuevas, E., Fausto, F., Rodríguez, A.: A better balance in metaheuristic algorithms: does it exist? Swarm Evol. Comput. 100671 (2020)
11. Song, H., Triguero, I., Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Progress Artif. Intell. **8**(2), 143–165 (2019). https://doi.org/10.1007/s13748-019-00185-z
12. Sutton, R.S.: Learning to predict by the methods of temporal differences. Mach. Learn. **3**(1), 9–44 (1988)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
14. Sutton, R.: Generalization in reinforcement learning: successful examples using sparse coarse coding. In: Advances in Neural Information Processing Systems, vol. 8 (1996)

15. Talbi, E.G.: Metaheuristics: From Design to Implementation, vol. 74. Wiley, Hoboken (2009)
16. Talbi, E.G.: Machine learning into metaheuristics: a survey and taxonomy of data-driven metaheuristics (2020)
17. Tapia, D., et al.: A Q-learning hyperheuristic binarization framework to balance exploration and exploitation. In: Florez, H., Misra, S. (eds.) ICAI 2020. CCIS, vol. 1277, pp. 14–28. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61702-8_2
18. Tapia, D., et al.: Embedding q-learning in the selection of metaheuristic operators: the enhanced binary grey wolf optimizar case. In: Proceeding of 2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA), IEEE ICA/ACCA 2021, Article in Press (2021)
19. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. J. Mach. Learn. Res. **8**(9) (2007)
20. Valdivia, S., et al.: Bridges reinforcement through conversion of tied-arch using crow search algorithm. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 525–535. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_42
21. Vásquez, C., et al.: Galactic swarm optimization applied to reinforcement of bridges by conversion in cable-stayed arch. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 108–119. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_10
22. Vásquez, C., et al.: Solving the 0/1 Knapsack problem using a galactic swarm optimization with data-driven binarization approaches. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12254, pp. 511–526. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58817-5_38
23. Wang, F.Y., Zhang, H., Liu, D.: Adaptive dynamic programming: an introduction. IEEE Comput. Intell. Mag. **4**(2), 39–47 (2009)
24. Xu, Y., Pi, D.: A reinforcement learning-based communication topology in particle swarm optimization. Neural Comput. Appl. **32**(14), 10007–10032 (2019). https://doi.org/10.1007/s00521-019-04527-9
25. Zhao, D., Zhu, Y.: MEC-a near-optimal online reinforcement learning algorithm for continuous deterministic systems. IEEE Trans. Neural Netw. Learn. Syst. **26**(2), 346–356 (2014)
26. Zhu, Y., Zhao, D., Li, X.: Using reinforcement learning techniques to solve continuous-time non-linear optimal tracking problem without system dynamics. IET Control Theory Appl. **10**(12), 1339–1347 (2016)