Soft Computing (2020) 24:4441–4453
https://doi.org/10.1007/s00500-019-04206-w

METHODOLOGIES AND APPLICATION

Tuning of reinforcement learning parameters applied to SOP using the Scott–Knott method

André L. C. Ottoni[1] · Erivelton G. Nepomuceno[2] · Marcos S. de Oliveira[3] · Daniela C. R. de Oliveira[3]

Published online: 6 July 2019

Abstract
In this paper, we present a technique to tune the reinforcement learning (RL) parameters applied to the sequential ordering problem (SOP) using the Scott–Knott method. The RL has been widely recognized as a powerful tool for combinatorial optimization problems, such as travelling salesman and multidimensional knapsack problems. It seems, however, that less attention has been paid to solve the SOP. Here, we have developed a RL structure to solve the SOP that can partially fill that gap. Two traditional RL algorithms, Q-learning and SARSA, have been employed. Three learning specifications have been adopted to analyze the performance of the RL: algorithm type, reinforcement learning function, and ϵ parameter. A complete factorial experiment and the Scott–Knott method are used to find the best combination of factor levels, when the source of variation is statistically different in analysis of variance. The performance of the proposed RL has been tested using benchmarks from the TSPLIB library. In general, the selected parameters indicate that SARSA overwhelms the performance of Q-learning.

Keywords Reinforcement learning · Sequential Ordering Problem · Factorial design · Scott–Knott method · Tuning parameters

1 Introduction

The reinforcement learning (RL) is an important field of machine learning. In RL, an agent learns from success and failure in interacting with an environment (Sutton and Barto 2018). This learning process leads agents to accumulate experience from reinforcements and penalties. The RL has been applied in many fields, such as in robotics, control, multiagent systems and optimization (Gambardella and Dorigo 2000; Kober et al. 2013; Shao et al. 2014; Bianchi et al. 2015; Yliniemi and Tumer 2016; Da Silva et al. 2019; Mnih et al. 2015; Asiain et al. 2019; Alipour et al. 2018; Carvalho et al. 2019; Li et al. 2019; Low et al. 2019; Bazzan 2019; Da Silva et al. 2019). A growing interesting to apply the RL can be seen in combinatorial optimization (Gambardella and Dorigo 1995; Likas et al. 1995; Miagkikh and Punch 1999; Mariano and Morales 2000; Sun et al. 2001; Ma et al. 2008; Liu and Zeng 2009; Lima Júnior et al. 2010; Santos et al. 2014; Alipour and Razavi 2015; Alipour et al. 2018; Ottoni et al. 2018; Woo et al. 2018; Miki et al. 2018; Chhabra and Warn 2019), such as the travelling salesman problem (TSP) (Gambardella and Dorigo 1995; Alipour et al. 2018), Job-Shop Problem (Zhang and Dietterich 1995; Cunha et al. 2020), the K-Server Problem (Costa et al. 2016) and the multidimensional knapsack problem (MKP) (Arin and Rabadi 2017; Ottoni et al. 2017). Although, it seems evident that a great number of works have been devoted to solving combinatorial optimization, less attention has been paid to the sequential ordering problem (SOP) (Skinderowicz 2017) using the RL.

Communicated by V. Loia.

✉ Erivelton G. Nepomuceno
nepomuceno@ufsj.edu.br

André L. C. Ottoni
andre.ottoni@ufrb.edu.br

Marcos S. de Oliveira
mso@ufsj.edu.br

Daniela C. R. de Oliveira
daniela@ufsj.edu.br

[1] Technologic and Exact Center, Federal University of Recôncavo da Bahia, Cruz das Almas, Brazil

[2] Control and Modelling Group (GCOM) - Department of Electrical Engineering, Federal University of São João del-Rei, São João del Rei, Brazil

[3] Department of Mathematics and Statistics, Federal University of São João del-Rei, São João del Rei, Brazil

The SOP is a similar problem to the TSP (Gambardella and Dorigo 2000). In TSP, the goal is to minimize the distance traveled on a route by going through all locations once and returning the starting city at the end of the route (Dorigo and Gambardella 1997). In SOP, the visit order constraints are added. It means that a certain node j must precede node i. The SOP has several applications in real problems, such as in production planning (Escudero 1988), single-machines scheduling problems (Letchford and Salazar-González 2016; Skinderowicz 2017), vehicle routing (Ascheuer et al. 2000; Gambardella and Dorigo 2000), single vehicle routing with pickup and delivery constraints (Ascheuer et al. 2000), the routing of a stacker crane in an automatic storage system (Ascheuer et al. 2000), in routing applications where a pickup has to precede the delivery (Fiala Timlin and Pulleyblank 1992), transportation problems in flexible manufacturing systems (Ascheuer et al. 2000), helicopter routing (Fiala Timlin and Pulleyblank 1992) and switching energy minimization in compilers (Shobaki and Jamal 2015). In this aspect, most works dealing with the solution for the SOP have been using particle swarm optimization (PSO) (Anghinolfi et al. 2011) or ant colony system (ACO) (Gambardella and Dorigo 2000).

Despite many works devoted to the SOP, to the best of authors knowledge, little attention has been paid to investigate the SOP using SARSA or Q-learning, wherein the parameters are tuned by a systematic approach. In fact, parameter estimation for RL has been recognized as an important part to achieve good performance in its execution and convergence (Schweighofer and Doya 2003; Even-Dar and Mansour 2003; Ottoni et al. 2018; Cardenoso Fernandez and Caarls 2018). There are some successful attempts to deal with automatic tuning for the RL. For instance, the authors in Cardenoso Fernandez and Caarls (2018) have applied to genetic algorithms to find parameters that best fit the performance of the SARSA and Q-learning RL algorithms. In a different approach, the learning rate is tuned using a cognitive network design tool, where the RL is applied to Optimized Link State Routing (OLSR) (McAuley et al. 2012). Additionally, the authors in Barsce et al. (2017), motivated by the increase of machine learning usage by industries and scientific communities in a variety of tasks such as text mining, image recognition, self-driving cars, and automatic setting of hyper-parameter, have applied an autonomous framework that employs Bayesian optimization and Gaussian process regression to optimize the hyper-parameters of a reinforcement learning algorithm.

In this paper, we have proposed an RL structure capable of solving the SOP, through a model defined in actions, states and reinforcements and an algorithm responsible for evaluating the precedence restrictions during the learning. A complete factorial experiment (Montgomery 2017) and the Scott–Knott method (1974) have been employed to investigate the performance of Q-learning (Watkins and Dayan 1992) and SARSA (Sutton and Barto 2018). Our procedure has also considered the influence of three reinforcement functions and the ϵ-*greedy* method (Sutton and Barto 2018) in the SOP resolution. The Scott–Knott method of multiple comparison has been used to identify the best combination of factor levels, when the source of variation is statistically different in analysis of variance (ANOVA) (Montgomery 2017).

The remainder of this paper is organized as follows. Section 2 presents basic theoretical concepts of the SOP and RL. Then, Sect. 3 describes a general overview of methodology. The results are given in Sect. 4, and concluding remarks are delivered in Sect. 5.

2 Theoretical foundation

2.1 Sequential ordering problem

2.1.1 Formulation problem

The TSP can be formulated as a graph $G(N, A)$, with $N = \{1, \ldots, n\}$ is the set of nodes and c_{ij} is the cost of each arc $(i, j) \in A$ (Bodin et al. 1983; Gambardella and Dorigo 2000; Applegate et al. 2007; Letchford and Salazar-González 2016). TSP can be classified into two groups: symmetric and asymmetric. In the symmetric TSP, the cost between locations i and j is equivalent to the cost between nodes j and i, that is, $c_{ij} \neq c_{ji}$. In an asymmetric travelling salesman problem (ATSP), the cost of moving a city i to a location j (c_{ij}) may be different from the cost of going from j to i (c_{ji}), that is, $c_{ij} \neq c_{ji}$.

The SOP can be interpreted as a specific case of the ATSP with precedence constraints (Gambardella and Dorigo 2000; Escudero 1988; Guerriero and Mancini 2003; Montemanni et al. 2008; Anghinolfi et al. 2011; Shobaki and Jamal 2015). The cost for SOP can assume: $c_{ij} \geq 0$ or $c_{ij} = -1$ (with $c_{ji} \geq 0$) (Gambardella and Dorigo 2000). There is a cost associated with the arc (i, j) when $c_{ij} \geq 0$. However, for $c_{ij} = -1$, there is the constraint of ordering the SOP, indicating that the node j must precede the node i. Thus, the element j should be accessed in an instant of time (t_1) before the access to the i the instant node t_2, that is, $t_1 < t_2$. The difference between the two instants of time may be greater than one step, that is, $t_2 - t_1 \geq 1$ (Gambardella and Dorigo 2000).

A mathematical formulation based on TSP (Bodin et al. 1983) for SOP is given below:

$$\text{Min} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}, \tag{1}$$

subject to:

$$\sum_{i=1}^{N} x_{ij} = 1 \quad (\forall j = 1, \ldots, N), \tag{2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad (\forall i = 1, \ldots, N), \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad (\forall i, j = 1, \ldots, N), \tag{4}$$

$$X = x_{ij} \in S \quad (\forall i, j = 1, \ldots, N), \tag{5}$$

$$c_{ij} \geq 0 \vee c_{ij} = -1 \wedge c_{ji} \geq 0 \quad (\forall i, j = 1, \ldots, N), \tag{6}$$

where Eqs. (1)–(5) refer to the mathematical model of the TSP. N is a set of nodes and Eq. (1) depicts the goal of minimizing the total distance traveled in the route. Thus, the cost of displacement between two cities (i and j) is given by c_{ij}. The decision variable $x_{i,j}$ assumes 1 if the arc (i, j) composes the solution and 0 otherwise. Equations (2) and (3) ensure that each location is visited only once. In addition, Eq. (4) ensures that the variable x_{ij} is binary. In Eq. (5), the set S represents any set of constraints that eliminate the formation of sub-routes. Finally, the Eq. (6) is the additional precedence restriction of the SOP.

2.1.2 Literature review

The SOP was initially formulated by Escudero (1988) for application in production planning systems. Since then, several methods have already been addressed in solving this problem. The authors in Gambardella and Dorigo (2000) present a hybrid system between ant colony system (ACS) and the local search method SOP-3-exchange. Along the same lines, the works of Montemanni et al. (2007) and Skinderowicz (2017) have discussed SOP solution using ACS.

The work of Anghinolfi et al. (2011) have investigated particle swarm optimization. The authors in Papapanagiotou et al. (2015) have performed the comparison of two exact algorithms in the solution of instances of three repositories: TSPLIB, Soplib06, and Compilers. An exact algorithm for solving SOP in an application directed toward compiler performance optimization have been addressed in Shobaki and Jamal (2015). Other papers presented formulations for the SOP resolution in vehicle routing applications such as Hernández-Pérez and Salazar-González (2009) and Letchford and Salazar-González (2016).

2.1.3 TSPLIB

A travelling salesman problem library (TSPLIB)[1] (Reinelt 1991) is an open-data repository that gathers options for TSP case studies and other combinatorial optimization problems such as SOP. The repository provides instances with different levels of resolution complexity and their optimal solution value for the problem. Table 1 illustrates the structure of the cost matrix of an SOP instance provided by TSPLIB. In this sense, we can observe some precedence constraints ($c_{ij} = -1$). For instance, the value of -1 in row 5 (i) and column 2 (j) of Table 1 indicates that node 2 must be accessed before node 5.

Table 1 The cost matrix of esc07 instance (TSPLIB)

0	0	0	0	0	0	0	0	1,000,000
−1	0	100	200	75	0	300	100	0
−1	400	0	500	325	400	600	0	0
−1	700	800	0	550	700	900	800	0
−1	−1	250	225	0	275	525	250	0
−1	−1	100	200	−1	0	−1	−1	0
−1	−1	1100	1200	1075	1000	0	1100	0
−1	−1	0	500	325	400	600	0	0
−1	−1	−1	−1	−1	−1	−1	−1	0

The value -1 indicates that priority should be given to visit. For instance, the value of -1 in row 5 (i) and column 2 (j) of this table indicates that node 2 must be accessed before node 5

2.2 Reinforcement learning

We have adopted two RL algorithms: Q-learning (Watkins and Dayan 1992) and SARSA (Sutton and Barto 2018). The Q-learning is based on updating a Q matrix according to Eq. (7):

$$Q_{t+1} = Q_t(s, a) + \alpha \left[r(s, a) + \gamma \max_{a'} Q(s', a') - Q_t(s, a) \right], \tag{7}$$

where s and a are state and action at the current instant (t), respectively; s' is state and a' is action at the next instant ($t + 1$); $Q_t(s, a)$ is the value at time t in the Q matrix for the pair state $\times$ action (s, a). Q_{t+1} is the updating of the learning matrix in $t + 1$ by executing the action a in state s; $r(s, a)$ is the reinforcement by the execution of the pair (s, a); $\max_{a'} Q(s', a')$ is the utility of s', that is, the maximum value in the line of Q referring to the new state; α is the learning rate; γ is the discount factor. Algorithm 1 shows the Q-learning, where the ϵ-*greedy* policy is adopted for actions selection. This method uses the ϵ parameter in the control between greedy and randomness in decision making according to Algorithm 2.

The SARSA method (see Algorithm 3) is a modification of Q-learning. The updating of the learning matrix in SARSA is given by Eq. (8) and ϵ-*greedy* policy follows Algorithm 2.

$$Q_{t+1} = Q_t(s, a) + \alpha[r(s, a) + \gamma\, Q_t(s', a') - Q_t(s, a)]. \tag{8}$$

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

```
Set the parameters: α, γ and ϵ
For each pair s,a the matrix Q(s,a)=0 should be initialized
Observe the state s
repeat
    Select the action a using ϵ-greedy method
    Take the action a
    Receive immediate reward r(s, a)
    Observe the new state s'
    Update Q (s, a) with Eq. (7)
    s = s'
until the stopping criterion is satisfied;
```

Algorithm 1: Q-learning.

```
Receive ϵ, Q(s,a)
Generate a random number (n_a) in [0, 1]
if n_a < ϵ then
    Select the random action (a_a)
    a = a_a
else
    Select the greedy action (a*) in Q(s,a) matrix
    a = a*
end
Return a
```

Algorithm 2: ϵ-*greedy* method

```
Set the parameters: α, γ and ϵ
For each pair s,a to initialize the matrix Q(s,a)=0
Observe the state s
Select the action a using ϵ-greedy method
repeat
    Take the action a
    Receive immediate reward r(s, a)
    Observe the new state s'
    Select the new action a using ϵ-greedy method
    Update Q (s, a) with Eq. (8)
    s = s'
    a = a'
until the stopping criterion is satisfied;
```

Algorithm 3: SARSA

3 Methodology

3.1 Reinforcement learning model

The RL model defined for the SOP resolution is comprised into three main features: states, actions, and reinforcements. The formulation adopted is based on definitions in RL structures for the TSP solution addressed in previous studies (Bianchi et al. 2009; Lima Júnior et al. 2010; Ottoni et al. 2018). After examining the logic of the SOP, we have proposed the following model in this paper:

- States: The set of states (S) was defined as all localities (nodes) that should be accessed for route formation by the agent (travelling salesman) (Lima Júnior et al. 2010; Ottoni et al. 2018). In this sense, the set S varies according to the number of locations in the instance.
- Actions: Each action was defined as a plan to visit another location (state) of the problem (Lima Júnior et al. 2010; Ottoni et al. 2018). For example, in a scenario with three states, if the agent is in locality 1 at the instant t, the movement can happen to node 2 (action 2) or node 3 (action 3). If the selected action is 3, the agent will be in state 3 at the instant $t + 1$. In addition, the action 3 is not available from the instant $t + 1$, since each locality can be accessed only once during the route.
- Reinforcements: The reinforcement function defines the returns for executing actions in certain states. For combinatorial optimization problems, one way is to associate the reward with the problem cost function as described in Eq. (1) (Bianchi et al. 2009; Lima Júnior et al. 2010; Ottoni et al. 2018). In this sense, for the SOP resolution, the reinforcement function can be related to the cost of displacement between cities i and j, given by c_{ij}. In some works this cost is also denoted by d_{ij}: distance between node i and node j. In this work, the following three types of reinforcement functions are analyzed:

$$R_1 = -d_{ij}, \tag{9}$$

$$R_2 = \frac{1}{d_{ij}}, \tag{10}$$

$$R_3 = -(d_{ij})^2. \tag{11}$$

In Eq. (9), the greater the distance between two locations i and j, the more negative the reinforcement R_1 (Bianchi et al. 2009; Ottoni et al. 2018). In Eq. (10), the reinforcement is the inverse of the distance (Lima Júnior et al. 2010). Finally, R_3 grows negatively with the square of the distance between i and j.

3.2 RLSOP algorithm

Considering the previous discussion, we have proposed the RLSOP (reinforcement learning for the SOP) algorithm (see Algorithm 4). This algorithm is used for analysis of the SOP restrictions of precedence in the RL decision making. The objective of RLSOP is to check for actions that must be performed before the action (a_t) selected by the ϵ-*greedy* policy.

```
1  a_t = ε-greedy()
2  cont = 0
3  while (cont == 0) do
4      if (there are precedence constraints for the selected action)
       then
5          if (at least one action corresponding to the precedence
           constraints of a_t has not yet been selected) then
6              cont = 0
7          else
8              cont = 1
9          end
10     end
11     if (cont==0) then
12         Remove action a_t from the list of available actions at the
           instant t;
13         a_t = ε-greedy()
14     end
15 end
16 Return a
```

Algorithm 4: RLSOP Algorithm. This is an adaption of the RL algorithm for solving the SOP.

The RLSOP Algorithm starts by receiving the action (a_t) selected by the ϵ-*greedy* policy (Algorithm 2) at the instant t. The *cont* variable is responsible for controlling the *while* command. Thus, *cont* is initialized with zero and the loop is only terminated if that variable receives 1. In the proposed algorithm, some conditional tests are performed. The first *if* command checks for precedence constraints for the selected action. Then, it is analyzed if the precedence constraints have already been executed before the instant t. If there are still any pending restrictions, then the control variable remains at zero. Otherwise, all precedence constraints have already been satisfied, and thus *cont* receives 1. On the other hand, if $cont = 0$, a_t is taken from the list of actions available at instant t. In addition, a new action is selected by the ϵ-*greedy* policy. The run continues until the proposed conditions are satisfied and *cont* receives 1.

3.3 Experiments

The experimental methodology has been based in Ottoni et al. (2016, 2018). The authors in Ottoni et al. (2018) have successfully adopted response surface models to estimate two RL parameters: learning rate and discount factor. Similarly, logistics regression models have been employed for the RL parameters analysis in the Q-learning and SARSA performance in Ottoni et al. (2016). In this sense, these two papers (Ottoni et al. 2016, 2018) have conducted experiments for a statistical analysis of RL parameters.

Experiments have been performed in the MATLAB software and comprised simulations with 11 TSPLIB instances. Table 2 presents the number of nodes (locations), the number of precedence constraints ($c_{ij} = -1$) and the optimal solution known in the literature for each problem.

Table 2 TSPLIB problems studied

Problem	Nodes	Constraints	Optimal
br17.10	18	48	55
esc07	9	22	2125
esc12	14	36	1675
esc25	27	62	1681
esc47	49	127	1288
esc63	65	360	62
esc78	80	440	18,230
ft53.1	54	117	7531
ft53.4	54	864	14,425
prob42	42	100	243
rbg109a	111	5548	1038

The RL performance of the instances resolution of Table 2 was analyzed by observing three learning specifications: algorithm type, reinforcement function, and ϵ parameter (ϵ-*greedy* policy) as follows:

- Algorithms: Q-learning and SARSA;
- Reinforcement functions: R_1 [Eq. (9)], R_2 [Eq. (10)] and R_3 [Eq. (11)];
- Parameter ϵ: [0.01; 0.05; 0.10].

For each instance, a total of 18 test combinations have been conducted: 2 (algorithms) $\times$ 3 (reinforcement functions) $\times$ 3 (ϵ values). Each combination was simulated five runs (repetitions) with 10,000 episodes. Each run is independent, that is, learning is accumulated over the 10,000 episodes, but its value is reset in the beginning of a new run. It is also worth mentioning that an episode is composed of iterations, responsible for initiating, developing and finalizing a route. The measure of performance is the total distance (cost) calculated at the end of the route in an episode. The learning rate and discount factor were defined based on results described in Ottoni et al. (2018) for the TSP: $\alpha = 0.75$ and $\gamma = 0.15$.

3.4 Factorial design

The chosen control factors in this study were the type of algorithms, reinforcement functions and ϵ-*greedy* politics. Table 3 summarizes these factors followed by their respective levels. The response variable accessed by the factorial design was the best solution found during run episodes.

The full-factorial design was given by:

$$y_{ijkl} = \mu + \zeta_i + \eta_j + \theta_k + (\zeta\eta)_{ij} + (\zeta\theta)_{ik} + (\eta\theta)_{jk} + (\zeta\eta\theta)_{ijk} + \xi_{ijkl}, \quad (12)$$

Table 3 Control factors and levels

Control factors	Number of levels	Levels		
Algorithms	2	Q-learning	SARSA	
R. functions	3	R_1	R_2	R_3
ϵ	3	0.01	0.05	0.10

where μ is the overall mean effect, ζ_i is the effect of the i_{th} level of the algorithms ($i = 1, 2$), η_j is the effect of the j_{th} level of the reinforcement functions ($j = 1, 2, 3$), θ_k is the effect of the k_{th} of ϵ-greedy politics ($k = 1, 2, 3$), $(\zeta\eta)_{ij}$ is the effect of the interaction between ζ_i and η_j, $(\zeta\theta)_{ik}$ is the effect of the interaction between ζ_i and θ_k, $(\eta\theta)_{jk}$ is the effect of interaction between η_j and θ_k, $(\zeta\eta\theta)_{ijk}$ is the effect of the three-way interaction, and ξ_{ijkl} is a random error component ($l = 1, 2, 3, 4, 5$).

Factorial design has been applied to estimate the effect of a factor in different levels of the other factors. Additionally, the effect of the interaction between two or more factors can be analyzed. Through the combination of all levels of the control factors in the study, 18 tests were obtained, which were replicated five times, generating 90 observations in total ($2 \times 3^2 \times 5$). All tests were conducted in random order.

By analysis of variance (ANOVA), all the hypotheses of non-differences in treatment means were tested through the F test with a significance level α_F equal to 0.05. The normality test of Kolmogorov–Smirnov (KS) (Conover 1971) and the Levene test (Fox and Weisberg 2011) for equality of variances were performed to assure that the experimental error terms were normally distributed, and the data variance were homogeneous. See Montgomery (2017) to obtain more details about the statistical analysis.

3.5 The Scott–Knott method

When the ANOVA indicates that the average levels of a source of variation differ, it is necessary to identify which combination of the factor levels are specifically different. There are various procedures of multiple comparisons in the literature. Scott–Knott (1974) method has been recognized as efficient alternative, as it is a method of grouping means that categorizes results without ambiguity.

The Scott–Knott method begins by partitioning the groups to maximize the sum of squares between groups. After ordering the means, the number of possible partitions ($k - 1$ partitions) is reduced. The sum of squares B_0, is defined according to Eq. (13):

$$B_0 = \frac{T_1^2}{k_1} + \frac{T_2^2}{k_2} + \frac{(T_1 + T_2)^2}{k_1 + k_2} \tag{13}$$

with

$$T_1 = \sum_{i=1}^{k_1} y_{(i)} \tag{14}$$

and

$$T_2 = \sum_{i=k_1+1}^{k_1+k_2} y_{(i)} \tag{15}$$

where $y_{(i)}$: ordered mean of treatment i ($i = 1, 2, \ldots, k$); $k = (k_1 + k_2)$ is the number of means to be separated; T_1 and T_2 are the totals of the two groups with k_1 and k_2 treatments, respectively. The maximum B_0 value obtained is used to compute the statistic λ according to Eq. (16):

$$\lambda = \frac{\pi}{2(\pi - 2)} \times \frac{B_0}{\hat{\sigma}_0^2}, \tag{16}$$

where $\hat{\sigma}_0^2$ is the estimator of maximum likelihood obtained by Eq. (17):

$$\hat{\sigma}_0^2 = \frac{1}{k + v}\left[\sum_{i=1}^{k}(y_i - \bar{y})^2 + v s_y^2\right], \tag{17}$$

and $\bar{y}$ is overall mean of treatments; v is the degrees of freedom of the residuals; $s_y^2 = \frac{\text{MSE}}{r}$ is the unbiased estimator of $\frac{\sigma^2}{r}$, being r the degrees of freedom associated with that estimator.

The statistics λ is tested by the *chi-squared test*, where the condition

$$\lambda \geq X^2_{\alpha_F; \frac{k}{n-2}}$$

indicates that the two groups are statistically different and should be tested separately for new possible divisions. On the contrary, the means are considered homogeneous and, further partitioning is therefore unnecessary.

3.6 Analysis of the results

The analysis of the results has been conducted as follows:

Table 4 Best results found in the five runs of simulation for all combinations of the three factors analyzed (algorithm, reinforcement function and ϵ parameter)

Alg	R	ϵ	br17.10	esc07	esc12	esc25	esc47	esc63	esc78	ft53.1	ft53.4	prob42	rbg109a
Q-learning	R_1	0.01	57	2150	1773	2209	2816	**62**	20,255	9255	17,444	323	1279
		0.05	**55**	2225	1696	2090	2891	63	20,125	9272	17,135	323	1254
		0.10	**55**	2150	1688	1757	2782	63	20,195	9373	17,086	323	**1252**
	R_2	0.01	**55**	**2125**	2162	8416	15,470	119	28,785	17,201	20,102	779	1468
		0.05	**55**	**2125**	2115	7845	13,982	89	24,985	17,889	20,062	753	1428
		0.10	**55**	**2125**	2020	7561	13,014	93	23,745	17,440	19,000	828	1410
	R_3	0.01	57	2200	1793	2368	3005	**62**	20,195	9419	17,815	314	1289
		0.05	**55**	**2125**	1765	2135	2692	**62**	20,295	9495	17,131	338	1265
		0.10	**55**	**2125**	1688	2135	2791	63	20,590	9546	17,086	331	1260
SARSA	R_1	0.01	57	**2125**	**1675**	1788	**2390**	63	19,725	8594	17,255	313	1292
		0.05	**55**	**2125**	**1675**	**1684**	2566	65	**19,625**	**8509**	16,833	**302**	1286
		0.10	**55**	**2125**	**1675**	**1684**	2658	66	19,855	8628	17,041	306	1276
	R_2	0.01	**55**	**2125**	1801	6133	7742	68	20,580	10,085	**16,687**	351	1383
		0.05	**55**	**2125**	1826	6291	6137	73	22,650	10,170	16,913	418	1373
		0.10	**55**	**2125**	1813	6912	7428	73	23,510	11,159	17,005	447	1398
	R_3	0.01	57	**2125**	1685	1807	2902	63	19,760	8640	17,240	359	1258
		0.05	**55**	**2125**	1685	2026	2995	67	19,825	8683	17,205	377	1283
		0.10	**55**	**2125**	**1675**	2092	3364	66	20,130	8836	16,925	397	1299
Optimal			55	2125	1675	1681	1288	62	18,230	7531	14,425	243	1038

The first three columns present all the combinations of the three factors. The remainder columns present the results for the 11 TSPLIB problems investigated in this work. For the convenience of the reader, the best known result for each problem according to Table 2 is also presented here. Finally, values in bold face indicate the best result found for each problem

- Preliminary analysis: search for the best solutions in each problem. In addition, figures to exemplify the evolution of the solution by learning episodes are presented. The computational times of the algorithms are also discussed.
- Tuning RL parameters
  - Adequacy of the models: the premises that guarantee the adequacy of the ANOVA models are checked (independence, homoscedasticity and normality of the residues).
  - Descriptive analysis: for problems in which the assumptions of the model were not met, a descriptive analysis of the means of solutions obtained was performed.
  - Scott–Knott results: presents the best configuration of algorithm, reinforcement function, and ϵ parameter.

4 Results

In this section, the results for the experiments are presented in two steps: preliminary analysis and tuning RL parameters. Afterward, the advantages of the proposed technique have been pointed out compared with other works in the literature.

4.1 Preliminary analysis

Table 4 presents the best solutions calculated for each TSPLIB problem for each combination of the algorithm, reinforcement function, and control factor ϵ. It is possible to observe that these factors may influence the outcome of the SOP optimization process. The results of Table 4 show that the SARSA algorithm achieved the best results (numbers in bold) in most instances.

Figures 1 and 2 show a performance during the learning process according to the algorithm type. The reinforcement function is exemplified in Figs. 3 and 4. Notice that the SARSA remained closer to the optimal solution of the ft53.1 problem, in relation to the Q-learning, during the simulation adopting R_1 and $\epsilon = 0.01$ (Fig. 1). Figure 2 presents a zoom, where (a): episodes 1 to 500) and (b): episodes 9500 to 10,000. Figure 2a shows that at the beginning of the learning the Q-learning and SARSA algorithms obtained similar performances. However, this balance between methods is not reflected at the end of learning (Fig. 2b), where the SARSA algorithm found better solutions. These results are evident in Table 4. This also shows that SARSA presents a slow convergence, and it requires a longer period to obtain better results.

Figures 3 and 4 show mainly a lower performance of the reinforcement function R_2 in relation to others, using esc78

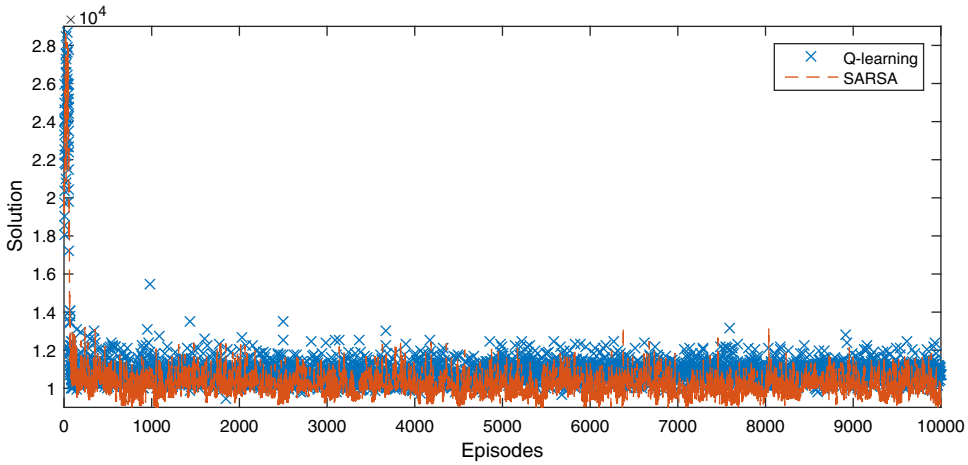

Fig. 1 Evolution of the solution by learning episodes for ft53.1. Curves for simulations of Q-learning and SARSA algorithms with the following specifications: R_1 and $\epsilon = 0.01$. In general, the SARSA solution remains closer to the optimal solution of the ft53.1 problem

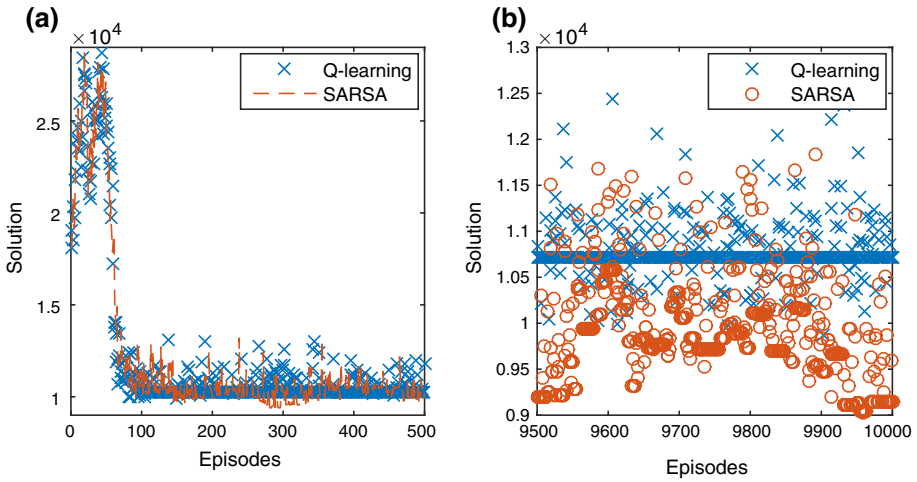

Fig. 2 Zoom of Fig. 1. **a** Episodes from 0 to 500, **b** episodes from 9500 to 10,000

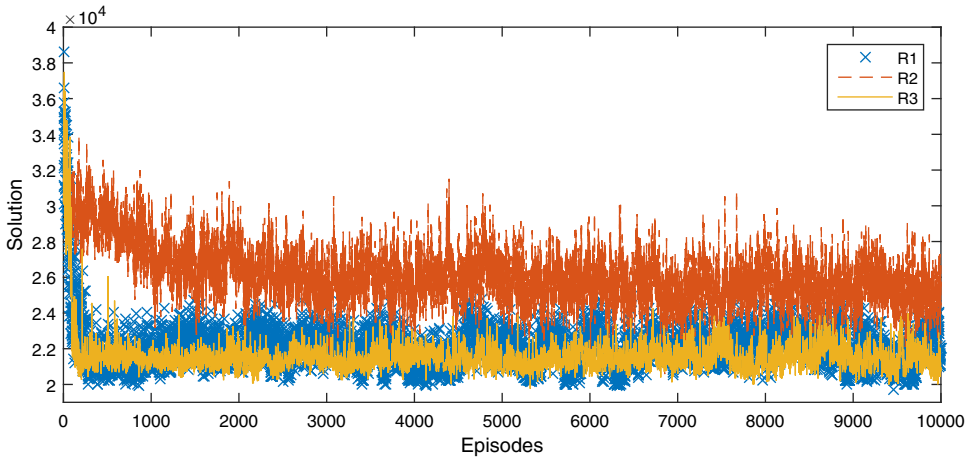

Fig. 3 Evolution of the solution by learning episodes for esc78 instance. Curves for simulations for reinforcement functions (R_1, R_2 and R_3) with the following specifications: SARSA and $\epsilon = 0.01$. It is clear here that R_2 presents worse performance than R_1 and R_3

instance, SARSA and $\epsilon = 0.01$. In this case, this difference in performance between the reinforcement functions is greater at the beginning of the learning process (Fig. 4a).

An optimal policy in RL is found when number of visits to each pair state-action approaches to infinity (Watkins and Dayan 1992). In this sense, the adoption of 10,000 episodes is an approximation to guarantee a convergence within an acceptable range. Figures 1 and 3 exemplify the importance of adopting a larger number of episodes. In Fig. 1, the best result was found by the SARSA algorithm (solution 8890) in episode 7693. In Fig. 2, the best solution was found by the reinforcement function R_1 (solution 19,725) in episode 9457.

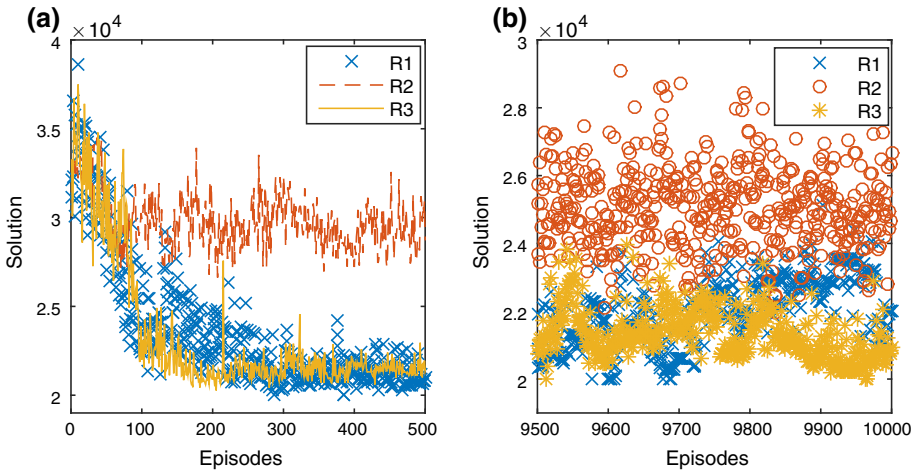

Fig. 4 Zoom of Fig. 3. **a** Episodes from 0 to 500, **b** episodes from 9500 to 10,000. The difference among the reinforcement decreases by the end of the simulation

Table 5 Average of computational time of 45 simulated runs using all combinations for the two algorithms

Problem	Q-learning	SARSA	Difference (%)
esc07	26.67	33.33	36.92
esc12	41.33	45.33	19.98
esc25	76.00	101.33	8.82
esc47	161.33	172.00	25.00
esc63	282.67	297.33	6.20
esc78	718.67	646.67	4.93
ft53.1	172.00	194.67	− 11.13
ft53.4	529.33	626.67	11.65
prob42	130.67	230.67	15.53
rbg109a	2277.33	3185.33	43.35
Average			17.25

The fourth column presents the increase in time in SARSA compared to Q-learning. SARSA presents 17.25% higher average time than Q-learning. The time is given in seconds

Table 5 presents the average of computational time of 45 simulated runs for all combinations (3 reinforcement functions $\times$ 3 ϵ values $\times$ 5 repetitions) of the two algorithms analyzed. The SARSA algorithm achieves 17.25% higher average than Q-learning.

4.2 Tuning RL parameters

The experiment analysis in factorial scheme (Montgomery 2017) was adopted to verify whether the analyzed learning conditions (algorithm, reinforcement function and ϵ-*greedy* policy) influenced the SOP optimization process. For each simulation of the 11 instances, the measure of performance is the best solution found during the episodes of a run (response variable) and an experiment model was fitted in a factorial scheme (2×3^2). The complete form has adopted with the main effects, double interactions, and triple interaction. All statistical analyses were conducted using the software R (R Core Team 2018).

4.2.1 Adequacy of the models

We have verified the assumptions that guarantee the adequacy of the models constructed from the summarization of the results of analysis of variance (ANOVA): independence, homoscedasticity, and residue normality.

First, the assumption of independent samples is accepted by the fact that the results of the run do not depend on each other, that is, each season is a simulation that starts with the reset learning matrix. Second, the assumption of residues normality was analyzed using the KS test (Conover 1971), and homoscedasticity was verified by the variance of Levene test (Fox and Weisberg 2011). For the KS test, the null hypothesis (H_0) is that the residual follows a normal distribution (p value > 0.05) and the alternative hypothesis (H_1) that do not (p value ≤ 0.05). Six problems (br17.10, esc07, esc12, esc63, ft53.1 and prob42) have not passed in the KS test. The normality assumption of other five problems (esc25, esc47, esc78, ft53.4 and rbg109a) have confirmed, as well as the assumption of homoscedasticity of variances.

4.2.2 Descriptive analysis

A descriptive analysis has been undertaken for the six problems which do not match the assumptions of normality. The average of five runs for each of the combinations of the three factors are shown in Table 6. The numbers highlighted in bold refer to the minimum value of the mean found in each instance.

Each simulated instance has its own characteristics, such as number of nodes, distances between localities, number of precedence constraints, and optimal solution defined by TSPLIB, shown in Table 2. For example, the instance br17.10 has 18 nodes with an optimal solution of 55 (distance unit),

Table 6 Computation of the average of five simulated runs for all combinations of the three factors analyzed (algorithm reinforcement function and ϵ-*greedy* policy)

Alg	R	ϵ	br17.10	esc07	esc12	esc63	ft53.1	prob42
Q-learning	R_1	0.01	58.8	2270.0	1773.0	**62.6**	9383.4	330.2
		0.05	56.6	2315.0	1754.0	63.2	9415.4	331.6
		0.10	55.8	2250.0	1707.2	64.6	9536.8	326.4
	R_2	0.01	63.6	2185.0	2404.6	141.2	17,902.0	877.0
		0.05	**55.0**	2220.0	2250.6	120.0	18,576.0	818.0
		0.10	**55.0**	2135.0	2127.6	101.8	18,049.0	913.0
	R_3	0.01	59.4	2280.0	1797.8	64.8	9524.6	321.2
		0.05	55.4	**2125.0**	1773.4	63.4	9616.0	350.8
		0.10	**55.0**	**2125.0**	1696.4	65.0	9725.4	347.4
SARSA	R_1	0.01	57.6	**2125.0**	1717.8	64.0	8785.2	317.0
		0.05	55.4	**2125.0**	1694.2	66.0	**8714.8**	**316.6**
		0.10	**55.0**	**2125.0**	**1683.0**	67.0	8841.6	328.4
	R_2	0.01	**55.0**	2140.0	1921.0	71.4	10,238.0	408.6
		0.05	**55.0**	**2125.0**	1852.2	74.8	10,593.0	439.4
		0.10	**55.0**	**2125.0**	1855.6	76.0	11,467.0	454.6
	R_3	0.01	58.2	**2125.0**	1709.2	63.8	8813.0	380.2
		0.05	55.6	**2125.0**	1691.4	67.8	8866.8	401.4
		0.10	56.2	**2125.0**	1685.2	70.0	9037.8	423.6

The best values found are in bold face. SARSA presents a better result in five of the six instances presented in this table

while esc12 has 14 nodes with an optimal solution of 1675. Observing the results, for the instance esc12 the combination SARSA, R_1 and $\epsilon = 0.01$ reached a result of 1683, while the combination Q-learning, R_2 and $\epsilon = 0.01$ presents an average of 2404.6. This difference among the results represents the influence of the definition of the RL parameters in terms of the solution (distance in the route) of the SOP instance. In a different way, the majority of best solutions for br17.10 have approached the optimal value of 55. In general, the SARSA algorithm combined with the reinforcement function R_1 presents the best results. Only in the instance esc63, a slight advantage in the use of Q-learning has been identified.

4.2.3 Scott–Knott results

For each of the instances (esc25, esc47, esc78, ft53.4, and rbg109a), which conditions of normality and homoscedasticity have been met, an experiment model was adjusted. It has been done in a complete factorial scheme, that is, with the main effects, the double interactions and the triple interaction. Table 7 presents p values of the KS (normality), Levene (homoscedasticity) and ANOVA tests and the best configuration of the Scott–Knott multiple comparison method. For example, for the instance rbg109a, the normality and homoscedasticity assumptions were satisfied, because the results of the KS and Levene tests were 0.71 and 0.47, respectively. In addition, the triple interaction (Alg $\times R \times \epsilon$) between the terms is significant by the ANOVA test ($p < 0.05$) for rbg109a. Moreover the Scott-Knott method tuning RL parameters (SARSA, R_3 and $\epsilon = 0.10$) for rbg109a instance.

In general, the analysis of Table 7 indicates that for the esc47, esc78, ft53.4, and rbg109a, the triple interaction between the factors was significant ($p < 0.05$), while for the esc25 instance, the double interactions Alg $\times R$ and Alg $\times \epsilon$ were significant. In such cases, where triple or double interaction are significant, it is necessary to deploy the degrees of freedom of a factor within each level of the other factors, and then perform multiple comparison tests (Scott–Knott method), in order to investigate which combination of factors provides the best results. The combination of the factors that provided the best results is found in the final part of Table 7. It has been noticed that the SARSA algorithm has been chosen for all instances. Regarding the reinforcement function and the ϵ-*greedy* method, it is noted that depending on the analyzed problem, there is a better combination of these two factors.

4.3 Comparison with other works

In this section, a comparative study of the proposed technique is performed. Five features are compared: combinatorial optimization problems, instances library, algorithms, tuning RL parameters, and results analysis. Table 8 presents the comparison of this proposal with different works which use reinforcement learning in combinatorial optimization problems: I (Gambardella and Dorigo 1995), II (Alipour et al.

Table 7 p values of the KS, Levene, and ANOVA tests and the best configuration of the Scott–Knott multiple comparison method

		esc25	esc47	esc78	ft53.4	rbg109a
Normality	KS	0.31	0.05	0.10	0.33	0.71
Homoscedasticity	Levene	0.05	0.05	0.01	0.01	0.47
ANOVA	Alg	0.00	0.00	0.00	0.00	0.01
	R	0.00	0.00	0.00	0.00	0.00
	ϵ	0.11	0.00	0.00	0.00	0.00
	Alg $\times$ R	**0.00**	0.00	0.00	0.00	0.00
	Alg $\times$ ϵ	**0.00**	0.00	0.00	0.00	0.00
	R $\times$ ϵ	0.88	0.00	0.03	0.00	0.21
	Alg$\times$ R $\times$ ϵ	0.24	**0.00**	**0.05**	**0.00**	**0.04**
Scott–Knott	Alg	SARSA	SARSA	SARSA	SARSA	SARSA
	R	R_2	R_1	R_2	R_1	R_3
	ϵ	0.01	0.01	0.01	0.05	0.10

Values in bold face indicate the interaction between the terms is significant

Table 8 Comparison of this proposal with different works that applied reinforcement learning in combinatorial optimization problems: I (Gambardella and Dorigo 1995), II (Alipour et al. 2018), III (Ottoni et al. 2018), IV (Ottoni et al. 2017) and V (Costa et al. 2016)

Problem		Proposed SOP	I (Gambardella and Dorigo 1995) TSP	II (Alipour et al. 2018) TSP	III (Ottoni et al. 2018) TSP	IV (Ottoni et al. 2017) MKP	V (Costa et al. 2016) K-Server
Instances	TSPLIB	✓	✓	✓	✓	–	–
Algorithms	Q-learning	✓	–	–	✓	✓	✓
	SARSA	✓	–	–	✓	–	–
	RLSOP	✓	–	–	–	–	–
Tuning RL parameters	Reinforcement function	✓	✓	✓	–	–	–
	ϵ	✓	–	✓	–	–	–
	α	–	✓	✓	✓	✓	–
	γ	–	✓	–	✓	✓	–
Analysis	ANOVA	✓	✓	–	–	–	–
	Scott–Knott	✓	–	–	–	–	–

2018), III (Ottoni et al. 2018), IV (Ottoni et al. 2017) and V (Costa et al. 2016).

An important contribution of this proposal is the application of RL to SOP solution (TSPLIB instances). In previous works, other combinatorial optimization problems were adopted, such as travelling salesman problem (TSP) (Gambardella and Dorigo 1995; Alipour et al. 2018; Ottoni et al. 2018), the multidimensional knapsack problem (MKP) (Ottoni et al. 2017), and the K-Server Problem (Costa et al. 2016).

Two traditional RL algorithms were adopted in this proposal: Q-learning and SARSA. Among the works presented in Table 8, only the authors in Ottoni et al. (2018) have also compared the performances of these two algorithms. In addition, it is worth mentioning that this work presented a method for tuning RL parameters. Other papers also analyzed the influence of the RL parameters (Gambardella and Dorigo 1995; Alipour et al. 2018; Ottoni et al. 2017, 2018). Nevertheless, they have not applied rigorous statistical techniques to choose the parameters, as we did in this paper by means of ANOVA and Scott–Knott method.

5 Conclusion

This paper has addressed the application of RL to solve the SOP. Using statistical tools, such as ANOVA and Scott–Knott method, the RL performance has been analyzed, providing a rigorous way to choose the best set of parameters. Three learning specifications have been investigated, namely algorithm type, reinforcement function, and ϵ parameter.

The novelty of this paper is the algorithm for analysis of the SOP restrictions of precedence in the RL decision making (RLSOP). In addition, a complete factorial experiment and the Scott–Knott method is then used to find the best combination of factor levels, when the source of variation

is statistically different in analysis of variance (ANOVA). Furthermore, this paper proposes a systematic analysis of the results composed of: preliminary analysis and tuning RL parameters (adequacy of the models, descriptive analysis and Scott–Knott results).

The results have shown a superiority of SARSA algorithm compared to Q-learning in the resolution of most of the simulated instances (see Tables 6, 7). The ANOVA and the Scott–Knott method showed that there is a statistically significant difference in adopting the SARSA algorithm for the solution of the esc25, esc47, esc78, ft53.4 and rbg109a. The method does not suggest a unique best solution for reinforcement function or ϵ-*greedy* method. In fact, each TSPLIB problem has presented a better combination of these two factors (see Table 7).

In future works, we intend to analyze the influence of the learning rate (α) and discount factor (γ) on the SOP solution (Ottoni et al. 2018). In addition, the performance of other RL algorithms and a comparison with traditional optimization methods, such as genetic algorithms, ant colony system, and particle swarm optimization should be also investigated. It is also intended to propose and analyze RL structures in the solution of other classical problems of combinatorial optimization.

Acknowledgements The authors are grateful to CAPES, CNPq/INERGE, FAPEMIG, UFSJ and UFRB.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Alipour MM, Razavi SN (2015) A new multiagent reinforcement learning algorithm to solve the symmetric traveling salesman problem. Multiagent Grid Syst 11(2):107–119

Alipour MM, Razavi SN, Feizi Derakhshi MR, Balafar MA (2018) A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. Neural Comput Appl 30(9):2935–2951

Anghinolfi D, Montemanni R, Paolucci M, Gambardella LM (2011) A hybrid particle swarm optimization approach for the sequential ordering problem. Comput Oper Res 38(7):1076–1085

Applegate D, Bixby RE, Chvátal V, Cook W (2007) The traveling salesman problem: a computational study. Princeton University Press, Princeton

Arin A, Rabadi G (2017) Integrating estimation of distribution algorithms versus q-learning into meta-raps for solving the 0–1 multidimensional knapsack problem. Comput Ind Eng 112:706–720

Ascheuer N, Jünger M, Reinelt G (2000) A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. Comput Optim Appl 17(1):61–84

Asiain E, Clempner JB, Poznyak AS (2019) Controller exploitation–exploration reinforcement learning architecture for computing near-optimal policies. Soft Comput 23(11):3591–3604

Barsce JC, Palombarini JA, Martinez EC (2017) Towards autonomous reinforcement learning: automatic setting of hyper-parameters using Bayesian optimization. In: 2017 XLIII Latin American Computer Conference (CLEI). IEEE, pp 1–9

Bazzan AL (2019) Aligning individual and collective welfare in complex socio-technical systems by combining metaheuristics and reinforcement learning. Eng Appl Artif Intell 79:23–33

Bianchi RAC, Ribeiro CHC, Costa AHR (2009) On the relation between ant colony optimization and heuristically accelerated reinforcement learning. In: 1st international workshop on hybrid control of autonomous system, pp 49–55

Bianchi RA, Celiberto LA, Santos PE, Matsuura JP, de Mantaras RL (2015) Transferring knowledge as heuristics in reinforcement learning: a case-based approach. Artif Intell 226:102–121

Bodin L, Golden B, Assad A, Ball M (1983) Routing and scheduling of vehicles and crews—the state of the art. Comput Oper Res 10(2):63–211

Cardenoso Fernandez F, Caarls W (2018) Parameters tuning and optimization for reinforcement learning algorithms using evolutionary computing. In: 2018 International conference on information systems and computer science (INCISCOS). IEEE, pp 301–305

Carvalho SA, Cunha DC, Silva-Filho AG (2019) Autonomous power management in mobile devices using dynamic frequency scaling and reinforcement learning for energy minimization. Microprocess Microsyst 64:205–220

Chhabra JPS, Warn GP (2019) A method for model selection using reinforcement learning when viewing design as a sequential decision process. Struct Multidiscip Optim 59(5):1521–1542

Conover WJ (1971) Practical nonparametric statistics. Wiley, New York

Costa ML, Padilha CAA, Melo JD, Neto ADD (2016) Hierarchical reinforcement learning and parallel computing applied to the k-server problem. IEEE Latin Am Trans 14(10):4351–4357

Cunha B, Madureira AM, Fonseca B, Coelho D, (2020) Deep reinforcement learning as a job shop scheduling solver: a literature review. In: Madureira A, Abraham A, Gandhi N, Varela M (eds) Hybrid intelligent systems. HIS 2018. Advances in intelligent systems and computing, vol 923. Springer, Cham

Da Silva F, Glatt R, Costa A (2019) MOO-MDP: an object-oriented representation for cooperative multiagent reinforcement learning. IEEE Trans Cybern 49(2):567–579

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66

Escudero L (1988) An inexact algorithm for the sequential ordering problem. Eur J Oper Res 37(2):236–249

Even-Dar E, Mansour Y (2003) Learning rates for Q-learning. J Mach Learn Res 5:1–25

Fiala Timlin MT, Pulleyblank WR (1992) Precedence constrained routing and helicopter scheduling: heuristic design. Interfaces 22(3):100–111

Fox J, Weisberg S (2011) An R companion to applied regression, 2nd edn. Sage, Beverly Hills

Gambardella LM, Dorigo M (1995) Ant-Q: a reinforcement learning approach to the traveling salesman problem. In: Proceedings of the 12th international conference on machine learning, pp 252–260

Gambardella LM, Dorigo M (2000) An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS J Comput 12(3):237–255

Guerriero F, Mancini M (2003) A cooperative parallel rollout algorithm for the sequential ordering problem. Parallel Comput 29(5):663–677

Hernández-Pérez H, Salazar-González J-J (2009) The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. Eur J Oper Res 196(3):987–995

Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. Int J Robot Res 32:1238–1274

Letchford AN, Salazar-González J-J (2016) Stronger multi-commodity flow formulations of the (capacitated) sequential ordering problem. Eur J Oper Res 251(1):74–84

Li D, Zhao D, Zhang Q, Chen Y (2019) Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. IEEE Comput Intell Mag 14(2):83–98

Likas A, Kontoravdis D, Stafylopatis A (1995) Discrete optimisation based on the combined use of reinforcement and constraint satisfaction schemes. Neural Comput Appl 3(2):101–112

Lima Júnior FC, Neto ADD, Melo JD (2010) Traveling salesman problem, theory and applications, chapter. In: Hybrid metaheuristics using reinforcement learning applied to salesman traveling problem. InTech, pp 213–236

Liu F, Zeng G (2009) Study of genetic algorithm with reinforcement learning to solve the TSP. Expert Syst Appl 36(3):6995–7001

Low ES, Ong P, Cheah KC (2019) Solving the optimal path planning of a mobile robot using improved q-learning. Robot Auton Syst 115:143–161

Ma J, Yang T, Hou Z-G, Tan M, Liu D (2008) Neurodynamic programming: a case study of the traveling salesman problem. Neural Comput Appl 17(4):347–355

Mariano C, Morales E (2000) A new distributed reinforcement learning algorithm for multiple objective optimization problems. In: Monard M, Sichman J (eds) Advances in artificial intelligence. Lecture Notes in Computer Science, vol 1952. Springer, Berlin, pp 290–299

McAuley A, Sinkar K, Kant L, Graff C, Patel M (2012) Tuning of reinforcement learning parameters applied to OLSR using a cognitive network design tool. In: 2012 IEEE wireless communications and networking conference (WCNC). IEEE, pp 2786–2791

Miagkikh V, Punch WF (1999) Global search in combinatorial optimization using reinforcement learning algorithms. In: Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 Congress on, vol 1

Miki S, Yamamoto D, Ebara H (2018) Applying deep learning and reinforcement learning to traveling salesman problem. In: 2018 international conference on computing, electronics communications engineering (iCCECE), pp 65–70

Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare M, Graves A, Riedmiller M, Fidjeland A, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533

Montemanni R, Smith DH, Gambardella LM (2007) Ant colony systems for large sequential ordering problems. In: 2007 IEEE Swarm intelligence symposium, pp 60–67

Montemanni R, Smith D, Gambardella L (2008) A heuristic manipulation technique for the sequential ordering problem. Comput Oper Res 35(12):3931–3944. Part Special Issue: Telecommunications Network Engineering

Montgomery DC (2017) Design and analysis of experiments, 9th edn. Wiley, New York

Ottoni ALC, Nepomuceno EG, Oliveira MS (2017) Performance analysis of reinforcement learning in the solution of multidimensional knapsack problem. Rev Bras Comput Apl 9(3):56–70

Ottoni ALC, Nepomuceno EG, de Oliveira MS (2018) A response surface model approach to parameter estimation of reinforcement learning for the travelling salesman problem. J Control Autom Electr Syst 29(3):350–359

Ottoni ALC, Nepomuceno EG, Oliveira MS, Cordeiro LT, Lamperti RD (2016) Analysis of the influence of learning rate and discount factor on the performance of q-learning and sarsa algorithms: application of reinforcement learning in autonomous navigation. Rev Bras Comput Apl 8(2):44–59

Papapanagiotou V, Jamal J, Montemanni R, Shobaki G, Gambardella LM (2015) A comparison of two exact algorithms for the sequential ordering problem. In: 2015 IEEE conference on systems, process and control (ICSPC), pp 73–78

R Core Team (2018) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna

Reinelt G (1991) TSPLIB—a traveling salesman problem library. ORSA J Comput 3(4):376–384

Santos JPQ, Melo JD, Duarte Neto AD, Aloise D (2014) Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. Expert Syst Appl 41(10):4939–4949

Schweighofer N, Doya K (2003) Meta-learning in reinforcement learning. Neural Netw 16(1):5–9

Scott AJ, Knott M (1974) A cluster analysis methods for grouping means in the analysis of variance. Biometrics 30:507–512

Shao J, Lin H, Zhang K (2014) Swarm robots reinforcement learning convergence accuracy-based learning classifier systems with gradient descent (XCS-GD). Neural Comput Appl 25(2):263–268

Shobaki G, Jamal J (2015) An exact algorithm for the sequential ordering problem and its application to switching energy minimization in compilers. Comput Optim Appl 61(2):343–372

Skinderowicz R (2017) An improved ant colony system for the sequential ordering problem. Comput Oper Res 86:1–17

Sun R, Tatsumi S, Zhao G (2001) Multiagent reinforcement learning method with an improved ant colony system. In: Proceedings of the IEEE international conference on systems, man and cybernetics, vol 3, pp 1612–1617

Sutton R, Barto A (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press, Cambridge

Watkins CJ, Dayan P (1992) Technical note Q-learning. Mach Learn 8(3):279–292

Woo S, Yeon J, Ji M, Moon I, Park J (2018) Deep reinforcement learning with fully convolutional neural network to solve an earthwork scheduling problem. In: 2018 IEEE international conference on systems, man, and cybernetics (SMC), pp 4236–4242

Yliniemi L, Tumer K (2016) Multi-objective multiagent credit assignment in reinforcement learning and NSGA-II. Soft Comput 20(10):3869–3887

Zhang W, Dietterich TG (1995) High-performance job-shop scheduling with a time-delay TD(lambda) network. In: Touretzky D, Mozer M, Hasseimo ME (eds) Advances in neural information processing systems, vol 8. MIT Press, Cambridge, pp 1024–1030