Optimized Artificial Neural Network System to Select an Exploration Algorithm for Robots on Bi-dimensional Grids

Liesle Caballero[1,2](✉), Mario Jojoa[1](✉), and Winston Percybrooks[1](✉)

[1] Universidad del Norte, Barranquilla, Colombia
{lieslec, jojoam, wpercyb}@uninorte.edu.co
[2] Institución Universitaria ITSA, Barranquilla, Colombia

Abstract. This article shows how Machine learning techniques are tested to predict the performance of different exploration algorithms: Random Walk, Random Walk WSB and Q Learning, for robots moving on a bi-dimensional grid. The overall objective is to create a tool to help select the best performing exploration algorithm according to a configurable testing scenario, without the need to perform new experiments, either physical or simulated. The work presented here focuses on optimizing the topology of an Artificial Neural Network (ANN) to improve prediction results versus a previously proposed approach. The Hill Climbing algorithm is tested as optimization method, compared with manual trial and error optimization. The ANN was selected because it has the best performance indicators in terms of Relative Absolute Error and Pearson Correlation Coefficient compared with Random Forest and Decision Trees. The metric used to measure the performance of the exploration algorithms is Maximum Number of Steps to target.

Keywords: Machine learning · 2D grid exploration · Artificial neural network Optimization algorithms · Robots

1 Background

The problem of exploration of an unknown environment has utmost importance in the area of mobile robotics due to its wide real-world applications, such as space exploration, search & rescue, hazardous material handling and military operations, among others [1, 2]. As a result, numerous publications about exploration algorithms have appeared in recent years [3–10]. In all these works, the problem of how to comparatively evaluate different competing exploration strategies is solved using a limited set of experimental conditions that are typically run in simulators. From the simulation results, evaluation metrics are quantified and used to select the algorithm with the best performance. If any change is made on the testing scenarios, then new simulations are required to determine the comparative performance of the algorithms under consideration. The work presented here shows an alternative approach to evaluate and compare the performance of exploration algorithms. The general objective is to develop a

J. E. Serrano C. and J. C. Martínez-Santos (Eds.): CCC 2018, CCIS 885, pp. 16–27, 2018.
https://doi.org/10.1007/978-3-319-98998-3_2

practical tool that predicts the performance of a given exploration algorithm under a configurable testing scenario, without needing additional experiments (either physical or simulated). The proposed approach, first described by the authors in [11], is based on predicting, as opposed to measuring through new experiments, the performance of the algorithms under the new testing scenarios. This method uses a prediction model extracted from measured experimental algorithm's performance under initial testing scenarios. Under this approach, algorithm's performance is treated as a random variable that can be modeled for prediction.

Traditional solutions for predicting the behavior of a random variable use statistical linear regression models and estimation of its probability density function (PDF) [12–15]. However, as it is shown in [11], these solutions have many limitations related to meeting model requirements when they are applied to this particular application. For example, the number of initial simulations needed to obtain good PDF estimations is too large to be practical, because a different PDF is generated for each intersection point on the bi-dimensional grid.

Given the shortcomings of classical prediction models for estimating navigation algorithm performance, in [11] it was proposed a Machine Learning (ML) algorithm as prediction method. Among the ML techniques tested in [11], Artificial Neural Networks (ANN) were identified as the best performers. The objective of the present work is to improve the results obtained in [11] using a optimization model to tune the parameters of the ANN. This optimized method is tested with the same two exploration algorithms considered in [11], Random Walk (RW) and Random Walk Without Step Back (RW WSB), plus a third exploration algorithm called Q Learning which is based on reinforcement learning. Then the main contribution of this paper is to measure the impact of automatic parameter optimization on the performance of the prediction algorithms, compared to the performance of the same algorithms but with manual optimization as done in [11].

In the reviewed literature, it was not possible to find additional studies that use ML to estimate and compare the performance of exploration algorithms. Using the proposed approach, it is possible to compare algorithms in environments that were not initially considered, that is, to make predictions under conditions not contemplated in the initial tests. This method saves up additional data collection, which can be costly and time consuming, becoming an invaluable tool for faster assessment of algorithmic alternatives. With ML, a set of new experimental conditions is entered as input and then the model computes a prediction of the corresponding performance metric for the algorithm under consideration. If effective, ML is a practical alternative in which experimental data needs to be collected only once for training the predictor.

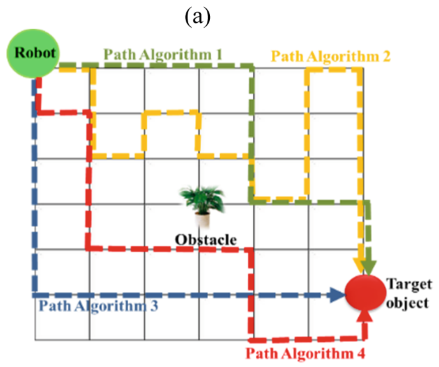
The rest of this document is organized as follows: Sect. 2 presents the experimental conditions and gives a detailed description of the proposed prediction system; Sect. 3 presents the methods and algorithms used for ANN optimization; in Sect. 4, the experimental results of the optimization methods are showed and analyzed; Finally, Sect. 5, provide conclusions and discusses results from the prediction models and pointers for future work.

2 Experimental Setup

2.1 Grid Based Scenario

In robotics, grid maps are often used for solving tasks like collision checking, path planning and localization [16]. The exploration algorithms chosen for testing are well known and relatively simple, the reasoning behind this decision is to reduce ambiguity in the interpretation of the new results. Since there is already a solid theoretical and empirical expectation about how those exploration algorithms perform when compared to each other, it allows us to focus solely on finding out if the machine learning predicted performance matches the expected results.

For the experiment considered here, a single robot (green dot in Fig. 1(a)) will explore a rectangular grid with the task of finding a target object (red dot in Fig. 1(a)), by moving from intersection to intersection. The robot is limited to move a single intersection at a time on four possible directions: up, down, right and left. The robot knows its starting point, but does not know the location of the target object, nor of any obstacles that may be on the grid. The robot must keep moving until it reaches the target object, as illustrated in Fig. 1(a). For physical experiments, a digital computer executes the exploration algorithms, while a wireless communication module transmits motion commands to a robotic platform, which has proximity and color sensors to identify both, obstacles and target object. Square grids with 3 × 3, 4 × 4, 5 × 5, 6 × 6 and 7 × 7 intersections are selected considering the dimensions of the real experimental scenario available for validation. For each grid, a number of obstacles ranging from 0 to 4 are used. Because the objective of this paper is to improve the result obtained in reference [11] then our optimized method must be tested in the same experimental scenario described above, where a bi-dimensional grids, obstacles, and known initial point were used. A 7 × 7 grid example is shown in Fig. 1(b).



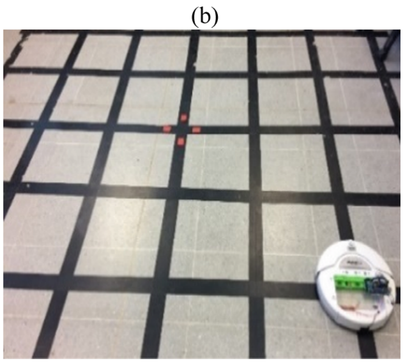
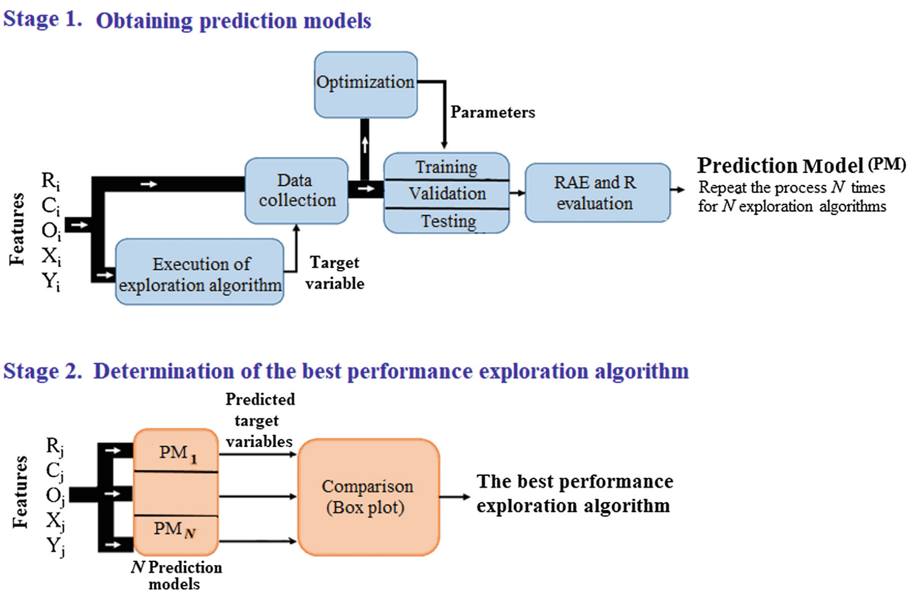


Fig. 1. Bi-dimensional grid and robotic platform in experimental scenario (Color figure online)

2.2 General Description of the ML Prediction System

Machine Learning (ML) is a scientific discipline in the field of Artificial Intelligence (AI) that creates systems that can learn automatically from experience and includes techniques capable of generalizing behaviors from information supplied in the form of examples, identifying complex patterns present in the input data [17]. The procedure described below, and illustrated in Fig. 2, is proposed to solve the issue of performance prediction to select the most suitable exploration algorithm.


Stage 1. Obtaining prediction models
Optimization
Parameters
Features
R_i C_i O_i X_i Y_i
Data collection
Training
Validation
Testing
RAE and R evaluation
Prediction Model (PM)
Repeat the process N times for N exploration algorithms
Execution of exploration algorithm
Target variable
Stage 2. Determination of the best performance exploration algorithm
Predicted target variables
Features
R_j C_j O_j X_j Y_j
PM_1
PM_N
N Prediction models
Comparison (Box plot)
The best performance exploration algorithm

*R=Number of rows of the grid, C=Number of columns of the grid, O=Number of obstacles, X= x coordinate of target object, Y= y coordinate of target object

Fig. 2. Diagram for building the proposed ML regression system

- **Execution of Exploration Algorithm**

To build the prediction models, each exploration algorithm under test must be executed under different experimental conditions (grid size, number and location of obstacles, location of target object) to compute training values for the chosen performance indicator variable. The Random Walk, Random Walk WSB and Q Learning algorithms were used for testing and evaluation in this work. These are all well known algorithms, therefore easing the task of validating the prediction results.

For the Random Walk algorithm each step is randomly taken on one of the possible directions (up, down, right and left). After each step the robot checks if it has reached the target object, and if not, looks for obstacles on the adjacent intersections to update its internal map and decide where to take the next step.

The Random Walk WSB (Without Step Back) algorithm emerges as an improvement over the basic Random Walk described above. It works similarly to the Random Walk algorithm, but in this case the robot remembers its previous position, and so it will never choose to go immediately back to that point, unless forced by obstacles. In this way the robot avoids back-and forward loops possible in the basic random walk [11].

The last exploration algorithm under test is based on Q Learning. Here, the robot chooses its path in a way to maximize some reward measure. After a learning period, the robot finally settles on a relatively stable path as long as the scenario does not change.

- **Data collection**

The data set obtained from the simulations of the Random Walk, Random Walk WSB and Q Learning algorithms is described in Table 1.

The performance indicator variable is computed for each exploration algorithm. From the work in [11], the appropriate indicator variable for the RW and RW WSB

Table 1. Characteristics and experimental conditions to evaluate Random Walk, Random Walk WSB and Q Learning algorithms

Configurable variables	Performance indicator variable	Study cases	Total dataset
- Number of grid's rows (R) - Number of grid's columns (C) - Number of obstacles (O) - Target object's X coordinate (X) - Target object's Y coordinate (Y)	*Max_NS*	- RxC: 3×3, 4×4, 5×5, 6×6 and 7×7 - O: 0, 1, 2, 3, 4 - Iterations per training scenario: • 15000 simulations for each experimental condition for RW and RW WSB cases • 200 learning cycles for Q Learning case	300 distinct scenarios

algorithms is *Max_NS*, defined as "the maximum number of steps in which the robot reached the target object, for a fixed set of conditions". Both exploration algorithms, RW and RW WSB, are executed 15000 times to obtain a value for *Max_NS* for each experimental condition.

For the case of the Q Learning algorithm, initial tests were run with 1000, 5000, 10000, 15000, 20000 and 25000 learning cycles. However, it was identified that the learning occurred between the first 100 learning cycles in all cases. Therefore, for the results reported here, data collection was done with only 200 learning cycles for each experimental condition. *Max_NS* is then computed from the last 100 cycles, when the variance of the data is minimal.

After the data is collected, techniques of machine learning such as Artificial Neural Networks (ANNs) and decision trees are used to solve the prediction problem.

- **Prediction Performance Parameters - RAE and R**

The ML prediction is evaluated using the following two parameters:

– Percentage relative absolute error ($\%RAE$): This compares true values with their estimates but relating it to the scale of its true value [18].

$$\%RAE = \frac{\sum_{i=1}^{M}\left|\widehat{Max\_NS_\imath} - Max\_NS_i\right|}{\sum_{i=1}^{M}\left|\overline{Max\_NS} - Max\_NS_i\right|} \times 100\% \tag{1}$$

Where, M is the number of experimental conditions evaluated, Max_NS_i are the experimental values that the variable Max_NS takes, $\widehat{Max\_NS_\imath}$ are the predicted values and $\overline{Max\_NS}$ is a mean value of Max_NS. The variable Max_NS could be in different ranges for each exploration algorithm, for this reason it was necessary to use a performance measure that can be easily compared across different variable ranges. Using RAE the errors should be comparable.

– Pearson correlation coefficient (R): This measures the similarity between the true values and the predicted values of the variables, that is, their linear dependence. Independently of the scale of measurement of the variables, the correlation measures the similarity between Max_N and $\widehat{Max\_NS}$. This function gives values between -1 and 1, where 0 is no relation, 1 is very strong, linear relation and -1 is an inverse linear relation [18]. In terms of the covariance of Max_NS and $\widehat{Max\_NS}$, the correlation coefficient R is definied as:

$$R = \rho\left(Max\_NS, \widehat{Max\_NS}\right) = \frac{cov\left(Max\_NS, \widehat{Max\_NS}\right)}{\sigma_{Max\_NS} \cdot \sigma_{\widehat{Max\_NS}}} \tag{2}$$

Where σ_{Max_NS} is the standard deviation of Max_NS and $\sigma_{\widehat{Max\_NS}}$ is the standard deviation of $\widehat{Max\_NS}$.

For the selection of the most suitable ML algorithm the following criteria were followed:

– Low percentage Relative Absolute Error ($\%RAE$), means that exist a narrow distance between the prediction data and true data.
– High correlation coefficient (R) between true values and predicted values, means that the predicted data is similar with the true data, that is, the variables have a high degree of relationship.

A low $\%RAE$ and R close to 1 will generate a good prediction and in consequence, can be reliably used to select the best performing exploration algorithm.

- **Training, validation and testing**

The selected machine learning algorithms will be evaluated using a data partition technique, cross-validation, which is commonly used in prediction problems. Cross-validation is a way to divide the input data in one testing set and k−1 training sets to evaluate its performance. The k value is chosen according to the size of the data and the building of the sets is done randomly. Cross-validation then involves k-folds that are randomly chosen and of roughly equal size. This process is repeated k times as each subset is used once for validation [19].

- **Comparison**

The box plot is a graph based on quartiles that visualizes the distribution of a data set and allows two or more data sets to be compared. For our testing scenario, lower values of the performance indicator variable *Max_NS* are preferred. Consequently, the best performing algorithm is the one that has the box with the lowest height.

3 ML Optimization

As a first approach, optimization by manual trial and error is applied on the following ML algorithms: ANN and Decision tree. These algorithms are commonly used to solve prediction problems. According to [11], ANN delivers better prediction results than Decision tree for RW and RW WSB which is confirmed with the results from the dataset used in this work, as shown in Table 2. For this reason, in this work the focus is on testing parameter optimization algorithms to improve the prediction indicators for the ANN.

For an ANN, the number of layers and neurons can be adjusted in order to improve the performance indicators, i.e. lower $\%RAE$ and raise R. According to [11], the Bayesian regularization and Levenberg-Marquardt training algorithms achieve the lowest $\%RAE$ and the highest R for the RW and RW WSB exploration algorithms, when using manual trial and error to optimize the number of layers and neurons of the ANN. Table 2 shows the results obtained with these training methods (and manual optimization) with a new data set for three exploration algorithms: RW, RW WSB and Q Learning.

The previous results, obtained using manual trial and error optimization, are used as baseline performance indicators for the proposed estimation algorithm. To improve those results, alternative optimization solutions using conventional and unconventional methods are reviewed. The work in [20] defines the optimization of a problem P, as the task of looking for parameter values that applied to P, satisfy certain expectations. For each problem P, an objective function f that measures the suitability of each possible solution of P is defined. The domain of f, that is, the set of points that can be proved as a solution to the problem, is called the space of solutions S. In an optimization problem, it is possible to find many areas of S with relatively good solutions (local optimums),

Table 2. Comparison between the correlation coefficients and relative absolute error obtained for three exploration algorithms and two technique ML using different training algorithms.

Exploration algorithm	Technique ML	Training algorithm	R	$RAE[\%]$	No. neurons per layer
Random Walk	ANN	Bayesian regularization	**0.9194**	**33.89**	**[27]**
		Levenberg-Marquardt	0.8911	44.26	[27]
	Decision tree	Trees: Random Forest	0.8872	36.1397	N/A
		Trees: RandomTree	0.8135	46.2505	N/A
Random Walk WSB	ANN	Bayesian regularization	**0.9203**	**34.5400**	**[7]**
		Levenberg-Marquardt	0.9019	38.2600	[7]
	Decision tree	Trees: Random Forest	0.9093	36.4538	N/A
		Trees: RandomTree	0.8783	43.5831	N/A
Q Learning	ANN	Bayesian regularization	**0.8830**	**42.89**	**[9]**
		Levenberg-Marquardt	0.8658	46.03	[9]
	Decision tree	Trees: Random Forest	0.8655	43.21	N/A
		Trees: RandomTree	0.7837	54.98	N/A

while a single area of *S*, or a few ones in the best case, provide the best overall solution (global optimum) [20].

For the work presented here, Hill Climbing optimization is proposed to search for the ANN topology (i.e. number of layers and neurons per layer) that maximizes R and minimizes RAE. In order to improve the changes of finding a global optimum as opposed to a local one, Hill Climbing is complemented with random restart, i.e. Hill Climbing is restarted several times using randomly chosen starting points [21].

According to [11] neural networks of 1 or 2 layers with few neurons generated predictors with good performance, for this reason, in this work the number of possible ANN layers has been limited to 1 or 2, while the number of neurons has been limited to a natural number in the range [1, 20]. As a result the solution space *S* for this optimization problem is bounded, which also increases the chances of finding the global optimum using Hill Climbing.

4 Experimental Results

4.1 Topology Optimization Using Hill Climbing Method

The results showed on Table 3 were obtained using *Levenberg-Marquardt* algorithm to train the ANN while performing parameter optimization with three different methods: manual trial-error, Hill Climbing and Hill Climbing (with Re-training).

Table 3. Comparison between the correlation coefficients and relative absolute error obtained for three exploration algorithms and three optimization methods to get the ANN topology using Levenberg-Marquardt training algorithm

Exploration algorithms	Method for optimizing ANN topology	*R*	*RAE*[%]	No. neurons per layer
Random Walk	Trial and error	0.8911	44.26	[27]
	Hill Climbing	0.9013	40.84	[8]
	Hill Climbing (Re-training)	**0.9815**	**12.79**	**[16:20]**
Random Walk WSB	Trial and error	0.9019	38.26	[7]
	Hill Climbing	0.9206	37.03	[11]
	Hill Climbing (Re-training)	**0.9817**	**15.49**	**[10:19]**
Q Learning	Trial and error	0.8658	46.03	[9]
	Hill Climbing	0.8838	46.63	[2:2]
	Hill Climbing (Re-training)	**0.9628**	**19.40**	**[20:20]**

The Hill Climbing algorithm with re-training obtained the best results for all three exploration algorithms. This indicates that the retraining process got a better adjustment of the ANN weights for the task. It is very important to use cross-validation to reduce the effect of over-fitting. For this reason, only three cycles of retraining were used.

The results showed on Table 4 were obtained in a similar way to the ones on Table 3 but replacing *Levenberg-Marquardt* training with *Bayesian regularization*.

Table 4. Comparison between the correlation coefficients and relative absolute error obtained for three exploration algorithms and three optimization methods to get the ANN topology using Bayesian regularization training algorithm

Exploration algorithms	Method for optimizing ANN topology	*R*	*RAE*[%]	No. neurons per layer
Random Walk	Trial and error	0.9194	33.89	[8]
	Hill Climbing	0.9296	35.35	[9:5]
	Hill Climbing (Re-training)	**0.9887**	**9.94**	**[19:4]**
Random Walk WSB	Trial and error	0.9203	34.54	[12]
	Hill Climbing	0.9297	32.19	[5:3]
	Hill Climbing (Re-training)	**0.9864**	**10.85**	**[19:4]**
Q Learning	Trial and error	0.8830	42.89	[9]
	Hill Climbing	0.8900	40.83	[4:19]
	Hill Climbing (Re-training)	**0.9613**	**8.51**	**[20:8]**

4.2 Comparison Between Real Data and Predicted Data

The topologies that achieved the best $\%RAE$ and R results from the previous subsection were used to build the ANN predictors. Each ANN computes an estimation of the corresponding value for *Max_NS* given a new set of scenario parameters, obtaining immediate results without the need for further experiments.

In order to evaluate the reliability of the predicted results to compare the performance of different exploration algorithms, we compare the predictions against the real experimental results when the exploration algorithms are used on the testing scenarios. The box plot chart in Fig. 3 shows the predicted data in contrast with the real data for each algorithm. The shape of the graphic for both sets of data is very similar. This suggests that the predicted results can be used as reliable decision parameters when choosing between exploration algorithms, since using them will gave the same conclusion than using actual experimental data. In our particular comparison the predicted data shows that the Q learning algorithm has a better performance than the Random Walk and Random Walk WSB algorithms, since the box plot corresponding to the predicted data for Q Learning, has lower values in each quartile with respect to the box plot for Random Walk and Random Walk WSB. Such result is consistent with what is indicated by the actual experimental data.

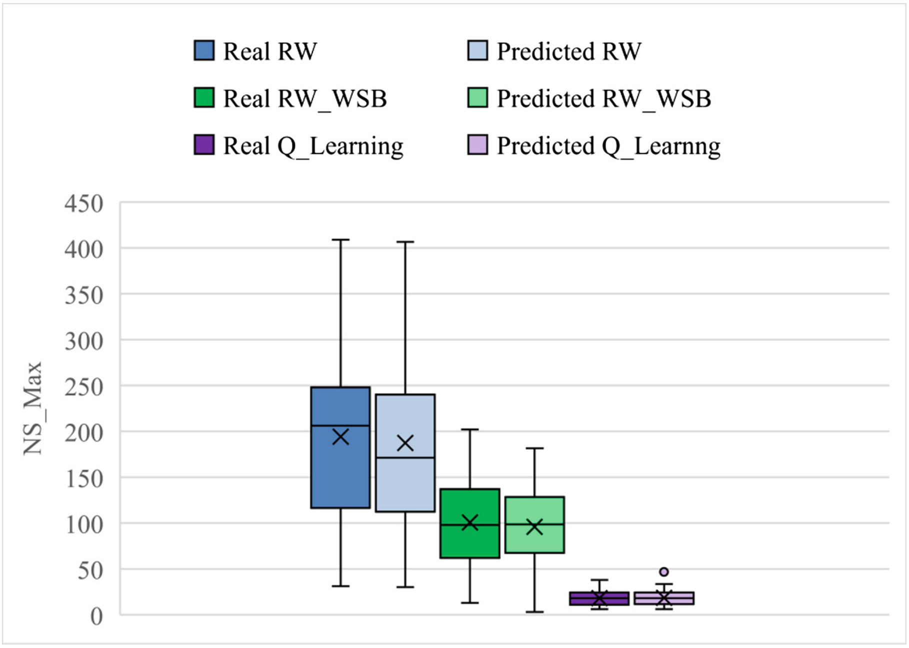


Fig. 3. Comparison between real and predicted target variable *NS_Max*

5 Conclusions

This research work extends previous work by the authors [11] demonstrating that an ML system can predict the performance of different exploration algorithms for robots moving on a bi-dimensional grid. The predicted values can then be compared to select the best exploration algorithm. A dataset of 300 different examples is enough to find a predictor with good performance, as measured by the corresponding $\%RAE$ and R values. Different experimental scenarios, not contemplated in the initial training dataset, are used for testing.

This work focuses on optimizing the topology of a neural network, that is, the number of layers and number of neurons in order to improve the initial results reported in [11]. Further effectiveness of the proposed method is also established by introducing a new exploration algorithm (Q Learning) into the tests. Three different optimization strategies are compared: manual trial and error, Hill Climbing and Hill Climbing with re-training. With single Hill Climbing the prediction performance improves versus the manual trial and error, however Hill Climbing with re-training outperforms the other two methods. Hill Climbing with re-training reduced $\%RAE$ from 33,89% to 9,94%, from 34,54% to 10,85% and from 42,89% to 8,51% respectively for the RW, RW WSB and Q learning exploration algorithms, when compared to manual trial and error optimization. Simultaneously, R was increased from 0,9194 to 0,9887, from 0,9203 to 0,9864 and from 0,8830 to 0,9313. Combining the results presented here with the ones previously reported [11] indicates the viability of building a Machine Learning-based tool to compare exploration algorithms under configurable testing scenarios. In future works, alternative optimization methods would be considered looking to reduce the current computational cost.

References

1. Zhang, Y., Gong, D., Zhang, J.: Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing **103**, 172–185 (2013)
2. Wu, H., Tian, G., Huang, B.: Multi-robot collaboration exploration based on immune network model. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM 2008, pp. 1207–1212 (2008)
3. Andreychuk, A., Bokovoy, A., Yakovlev, K.: An empirical evaluation of grid-based path planning algorithms on widely used in robotics raspberry pi platform. In: The 2018 International Conference on Artificial Life and Robotics (ICAROB 2018), pp. 1–4 (2018)
4. Akutsu, T., Yaoi, S., Sato, K., Enomoto, S.: Development and comparison of search algorithms for robot motion planning in the configuration space. In: Proceedings IROS 1991: IEEE/RSJ International Workshop on Intelligent Robots and Systems '91, pp. 429–434 (1991)
5. Faigl, J., Simonin, O., Charpillet, F.: Comparison of task-allocation algorithms in frontier-based multi-robot exploration. In: Bulling, N. (ed.) EUMAS 2014. LNCS (LNAI), vol. 8953, pp. 101–110. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17130-2_7
6. Kulich, M., Juchelka, T., Přeučil, L.: Comparison of exploration strategies for multi-robot search. Acta Polytech. **55**(3), 162 (2015)
7. Holz, F., Behnke, S., Basilico, N., Amigoni, F.: Evaluating the efficiency of frontier-based exploration strategies. In: ISR/Robotik 2010, vol. 1, no. June, p. 8 (2010)
8. Juliá, M., Gil, A., Reinoso, O.: A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Auton. Robots **33**(4), 427–444 (2012)
9. Amigoni, F.: Experimental evaluation of some exploration strategies for mobile robots. In: IEEE International Conference on Robotics and Automation (2008)
10. Martínez Puerta, J.J., Vallejo Jiménez, M.M.: Comparación de estrategias de navegación colaborativa para robótica móvil. Universidad Autónoma de Manizales (2016)
11. Caballero, L., Benavides, C., Percybrooks, W.: Machine learning-based system to estimate the performance of exploration algorithms for robots in a bi-dimensional grid. In: 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), pp. 1–6 (2018)

12. Pitarque, A., Ruiz, J.C., Roy, J.F.: Las redes neuronales como herramientas estadísticas no paramétricas de clasificación. Psicothema **12**(SUPPL. 2), 459–463 (2000)
13. Tabachnick, B., Fidell, L.: Using Multivariate Statistics. Harper & Row, New York (1996)
14. Cohen, J., Cohen, P.: Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, 545 p. Taylor & Francis, Milton Park (1983)
15. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning, 8th edn. Springer, New York (2017). https://doi.org/10.1007/978-1-4614-7138-7
16. Lau, B., Sprunk, C., Burgard, W.: Efficient grid-based spatial representations for robot navigation in dynamic environments. Rob. Auton. Syst. **61**(10), 1116–1130 (2013)
17. Flach, P.: Data, Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press, New York (2012)
18. machine learning - How to interpret error measures? - Cross Validated (2017). https://stats.stackexchange.com/questions/131267/how-to-interpret-error-measures. Accessed 28 Feb 2018
19. Cross-Validation - MATLAB &amp; Simulink (2018). https://la.mathworks.com/discovery/cross-validation.html. Accessed 20 Apr 2018
20. Izquierdo, S.K., Rodó, D.M., Bakx, G.E., Iglésias, R.B.: Inteligencia artificial avanzada. Editorial UOC - Editorial de la Universitat Oberta de Catalunya (2013)
21. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)