Genetic Algorithm for Agent Hyperparameter Optimisation in the Test of Algorithmic IQ

Ondřej Vadinský[0000−0002−0910−3140], **Michal Dvořák, Matyáš Dlouhý**

Department of Information and Knowledge Engineering, Prague University of Economics and Business
Náměstí Winstona Churchilla 4, 130 67 Praha 3, Czech Republic
Email: ondrej.vadinsky@vse.cz

Abstract

A genetic algorithm is used to search for optimal hyperparameter configurations of reinforcement learning agents in the AIQ test. AIQ score thus forms a natural solution fitness metric in a population of agents whose genome is encoded by hyperparameter values captured by a real number vector. In an experiment with a Q-learning agent, the benefits and limits of the chosen approach are illustrated. The available computational power can be used to an optimum closer-focused more fine-grained search as opposed to a systematic grid-search. However, the price is a less accurate idea of low-performing agent configurations. This selection bias of the genetic algorithm can be reduced by using linear rank or roulette wheel selection modes and by increasing the search effort relative to the search granularity.

1 Introduction

When evaluating intelligence of artificial systems, Universal psychometrics (Hernández-Orallo, 2017) renounces anthropocentric perspective on intelligence. Instead, it offers universalist and unified view on human, animal and machine intelligence. It also uses algorithmic information theory (Chaitin, 1987) to provide the proposed evaluation methods with strong theoretical roots. The algorithmic intelligence quotient test (AIQ test) is an example of such theoretically well-founded, yet practically computable, environment-general intelligence metrics (Legg and Veness, 2013). To reach high AIQ score, the tested agent is required to demonstrate a high level of performance across a wide range of environments and over long interaction sequences. This, however, makes the test computationally demanding.

Reinforcement learning (RL) (Sutton and Barto, 2018) enables learning through interaction with an environment. An agent chooses its actions and receives observations and rewards. These it can use to adopt policies of useful behaviours that maximise future rewards. There are several different architectures of RL agents. Their behaviour and success in environments is further influenced by their hyperparameter settings.

To informatively assess the intelligence of RL agents in the AIQ test, it is thus necessary to search a large agent configuration space further increasing the test computational demands. While a systematic search would give the most representative picture, its computational demands are quite high. As the ultimate goal is to find well-performing configurations, the problem, alternatively, constitutes an optimisation task.

Genetic algorithms (Eiben and Smith, 2015) offer evolution inspired approach to searching the space of possible configurations in the optimisation task. These configurations constitute a population of individuals. There, the better procreate and survive to the next generation together with their offspring s that combine (and possibly mutate) properties of their parents. Evolutionary principles assure that the average fitness of the population increases over multiple generations.

This contribution will show the use of a genetic algorithm to search for the optimal hyperparameter configuration of RL agents in the AIQ test. An experiment with a Q-learning agent (Watkins, 1989) will illustrate the benefits and limits of the proposed approach.

Section 2 will shortly introduce the required background. Section 3 will present the important aspects of implementing the genetic algorithm into the AIQ test. Section 4 will describe the conducted experiment and the resulting assessment of genetic hyperparameter optimisation. Finally, Section 5 will discuss the findings and conclude the paper.

2 Intelligence Evaluation, Reinforcement Learning and Genetic Algorithms

In this paper a genetic algorithm is used to search for optimal hyperparameter configuration of Q-learning in the AIQ test. Section 2.1 will introduce the AIQ test, Section 2.2 will describe Q-learning and Section 2.3 will cover the genetic algorithm.

2.1 Algorithmic Intelligence Quotient Test

Universal psychometrics provides general intelligence evaluation methods (Hernández-Orallo, 2017) with strong theoretical foundations in the algorithmic information theory (Chaitin, 1987). The *algorithmic intelligence quotient test* of Legg and Veness (2013) is a

practically feasible example of such a method. Approximating the definition of *universal intelligence* (Legg and Hutter, 2007): "Intelligence measures an agent's ability to achieve goals in a wide range of environments," the test is an environment-general metrics of intelligence. The complexity of an environment is used to weight an agent's success in it and a wide sample of possible environments is considered for the overall evaluation.

Legg and Veness (2013) describe the *algorithmic intelligence quotient test* by Equation 1:

$$\hat{\Upsilon}(\pi) := \frac{1}{N}\sum_{i=1}^{N} \hat{V}_{p_i}^{\pi}, \text{ where } \quad \hat{V}_{p_i}^{\pi} := \frac{1}{k}\sum_{j=1}^{k} r_j, \quad (1)$$

where *the AIQ estimate of universal intelligence* $\hat{\Upsilon}$ of an agent π is given by its ability to achieve goals as measured by the empirical value function $\hat{V}_{p_i}^{\pi}$ as an average reward achieved by the agent over k interactions with an environment program p_i from a finite sample of N environment programs that are sampled according to Solomonoff's (1964) *universal distribution*: $M_{\mathcal{U}}(x) := \sum_{p:\mathcal{U}(p)=x*} 2^{-l(p)}$ using a low-level *BF reference machine* (Müller, 1993).

Vadinský (2018) improved the test's sampling procedure ensuring higher discriminative power of the used environments (Hernández-Orallo and Dowe, 2010). Recent results (Vadinský and Zeman, 2024; Vadinský and Dvořák, 2025) showed the test strengths and weaknesses when evaluating deep RL agents. This paper focuses on the identified trade-off between the computational demands that the test poses and informativeness of an agent's evaluation. Typically, the test is run with the episode length of $k = 100{,}000$ interactions (to allow the low-performing agents to converge) over the *sample size* of $N = 10{,}000$ environment programs (to achieve a small confidence interval of the AIQ estimate). This allows for a reasonably informative evaluation in the test's default setting of 5 symbols and 1 observation to be conducted on the current hardware (possibly with $N = 2,000$ for the more demanding agents).

2.2 Reinforcement Learning with Q-Learning

In reinforcement learning an agent chooses which actions to take in an environment based on received rewards and observations. The goal of the agent is to find a policy (a mapping of desired actions to observations) that maximises future rewards. Of several ways to achieve this, this paper employs a model-free, action-value, off-policy approach that is implemented in a classical *Q-learning* agent. (Sutton and Barto, 2018)

This agent uses trial and error to discover which actions are useful in observed states and thus learns an *action-value* function $Q(s, a)$. This function describes the quality of a distinct state-action pair for maximising future rewards and is updated using a well-known *temporal difference* approach. The function is represented by an exhaustive table that is used to derive a useful policy (an action that in a given state has the highest Q-value). A mechanism called *eligibility traces* is used to attribute what past action caused the current reward. Further, the agent is implemented as *epsilon-greedy* to balance random explorative behaviour with greedy exploitative behaviour as dictated by its currently known policy. (Sutton and Barto, 2018; Watkins, 1989)

Hyperparameters of the Q_λ agent in the AIQ test are as follows (Watkins, 1989; Legg and Veness, 2013):

- *Initial Q value (Q)* – value used to initialize the Q-table, defaults to 0.
- *Eligibility Traces Decay Rate* (λ) – rate in which the eligibility trace is decayed over time; if set to 0 eligibility traces are turned off.
- *Learning Rate* (α) – rate in which prediction error influences the Q-table value.
- *Minimal Exploration Rate* (ϵ) – minimal rate of explorative (random) actions.
- *Exploration Rate Decay Length (EDL)* – number of interactions over which ϵ is linearly decayed from 1 to its minimal value; if set to 0 constant exploration rate is used.
- *Discount Factor* (γ) – future rewards discount factor.

Q-learning is used as an example RL agent to be evaluated in the AIQ test since it has sufficiently complex configuration space, yet runtimes are low making an extensive hyperparameter search feasible.

2.3 Optimisation using Genetic Algorithms

When focusing on the high-performing extreme, the task of an artificial agent evaluation in the AIQ test is transformed to an *optimisation task*.

A way of tackling the optimisation task is offered by *evolutionary computing* (Eiben and Smith, 2015). Possible solutions of the task constitute a *population of individuals* (represented by their *genomes*) whose suitability (*fitness*) as a task solution is measured. By selecting the currently most suitable solutions and by their procreation (*crossover*) a new generation can arise that should include better solutions than the previous one. Variability of the individuals in the population is kept by *mutation* (a random change in an individual's genes) that can sometimes occur. Eiben et al. (1991) proved that a properly constructed genetic algorithm can find the global optimum. Furthermore, convergence to a sufficiently good solution is fast. A particular realisation of the genetic algorithm is task-specific, this paper focuses on evolutionary strategies working with genomes represented by vectors of reals that create the next generation using crossover and mutation (Schwefel, 1995).

An agent configuration constituting a possible solution is represented by a vector of hyperparameter values. To select which individuals survive to the next generation, a fitness function is used to measure their suitability. AIQ score serves as a natural fitness function. The method of selection should, however, keep the population diverse (Eiben and Smith, 2015). Jebari (2013) lists popular methods that satisfy this criterion: *roulette wheel* (selection probability is given by the ratio of the individual's fitness score to the sum of all scores), *linear rank* (selection probability depends on the order of the fitness scores), and *tournament selection* (pairs of individuals are chosen, those from the pairs with the higher fitness are selected). Crossover combines genetic information of selected parents to ascertain what is the fitness of a solution "in-between" them. As crossover happens after selection, there is a good chance that the fitness would be high. Eiben and Smith (2015) offer two methods of crossover for vectors of reals: *discrete recombination* (parts of genome are copied from parents), and *arithmetic recombination* (genome of parents is averaged). It is, however, mutation that enables genetic algorithm to explore new subspaces of the solution space as it changes a part of the genome with a given probability. Eiben and Smith (2015) offer two methods of mutation applicable to vectors of reals: *random reset* (chosen value is reset to a random value from the interval of permissible values), and *creep mutation* (chosen value is modified by random value).

3 Implementing Genetic Algorithm into the AIQ Test

A genetic algorithm hyperparameter search (GAHS) is implemented as a Python module for the AIQ test. It represents individuals as vectors of reals (the dimension of the vector being the number of hyperparameters of a particular agent). We use both crossover (using a discrete recombination) and mutation (using a random reset). One of the three selection methods (roulette wheel, linear rank, and tournament) can be chosen at a runtime. As the module is integrated with the test, all the test functions can be used. In particular, this includes the capability of loading previous results of tested agents. Thus, if a configuration was already tested, it will not be retested again. This enables us to use all the previous results as global seed to reduce the cost of certain experiments. Important parameters of GAHS include:

- *Population size* – nr. of individuals in a generation.
- *Number of epochs* – nr. of generations tested.
- *Number of survivors* – how many individuals are selected to procreate.
- *Mutation probability* – probability for each part of genome to be mutated.

The first generation is created randomly, or it can be seeded. For a next generation, parents are kept and the remainder of the population size is created in half by mutation and in half by crossover.

Agents from the AIQ test should be referred to by an *agent reference*. This reference identifies the tested agent and most importantly defines *generators* that are used to construct the genome of a random valid configuration. Here, any function that returns a valid value of a particular hyperparameter can be used. Depending on the experimental setting, some hyperparameters can be searched in finer steps than other, not search at all, or searched only in partial intervals to reflect knowledge about the agent. Multiple agent references are possible for a single AIQ test agent to facilitate experimentation.

The resulting implementation of genetic hyperparameter search is part of the Appendix.

4 Assessment of Genetic Hyperparameter Optimisation

Two sets of experiments were conducted with GAHS in the AIQ test to illustrate its capabilities. The experiment described in 4.1 focuses on the ability of GAHS to cover the hyperparameter space reasonably well. The experiment described in 4.2 focuses on finding the optimal configuration given the same computational resources.

4.1 The Coverage Experiment

In our first set of experiments, we want to ascertain that GAHS is able to reasonably cover the configuration space of an agent compared to a full grid-search.

Assumptions

A1 *At Least the Same Search Effort.* GAHS is set to generate at least the same number of agent configurations (by population initialisation and by further crossovers and mutations) as the compared grid-search. The actual number of configurations generated by GAHS can be higher since there can be duplicate parents selected. These duplicates are replaced by randomly generated configurations to keep the set population size.

A2 *The Same Search Granularity.* GAHS is set to generate configurations of agents from exactly the same values as the compared grid-search.

Research Questions

R1 What is the coverage of the agent configuration space compared to the grid-search given *A1* and *A2* assumptions?

R2 How does the coverage differ between the roulette wheel, linear rank and tournament selection?

R3 Given the assumption *A1*, what is the overhead of GAHS due to the *A2* assumption and how does it differ between the roulette wheel, linear rank and tournament selection modes?

Hypotheses

$H1_{strong}$ Given *A1* and *A2* assumptions, GAHS covers at least 95% of the agent configuration space covered by the grid-search.

$H1_{weak}$ Given *A1* and *A2* assumptions, GAHS covers at least 80% of the agent configuration space covered by the grid-search.

H2 Given *A1* and *A2* assumptions, the selection modes of GAHS do not have a significant impact on the coverage percentages.

Experiment Settings In this experiment, the AIQ test is used in its default settings with BF 5 reference machine, episode length of $k = 100{,}000$ interactions and the sample size of $N = 10{,}000$ environment programs.

Then we conducted a grid-search for Q_λ configurations trying out the following hyperparameter values:

λ 0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 0.99, and 0.995;

α 0.3, 0.5, and 0.7;

ϵ 0.005, 0.01, 0.02, 0.03, and 0.04;

γ 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, and 0.995.

Both Q and *EDL* were set to 0. Thus, the search checks only constant ϵ-greedy strategies. Nevertheless, the grid-search totals to 945 configurations.

Finally we run GAHS set in the following way:

- Population size: 63 individuals.
- Number of epochs: 22 generations.
- Number of survivors: 21 individuals.
- Mutation probability: 25%.

Since agent references were set accordingly to the grid-search, we can reuse the results. Thus, we ran this setup 33 times for each of the 3 selection modes.

Results A statistics of the configuration coverage among the different selection modes is given in Table 1.

Data Analysis As can be seen from the means in Table 1, both strong and weak formulation of *H1* can be rejected. Looking at the means and their half confidence intervals (HCI), *H2* has to be rejected as well. There is a small but significant difference in coverage that makes tournament selection slightly worse than roulette wheel.

Measure	**Statistics for Selection Mode**		
	roulette	*linear rank*	*tournament*
Mean	65.90	65.73	65.07
Standard Error	0.1637	0.2054	0.2026
HCI of the Mean	0.3208	0.4026	0.3971
Minimum	63.92	63.17	61.69
First Quartile	65.19	64.97	64.34
Median	66.03	65.82	65.08
Third Quartile	66.56	66.35	65.93
Maximum	67.94	67.72	66.88
Standard Deviation	0.9402	1.1799	1.1638

Tab. 1: Statistics of configuration space coverage (as percent of 945 grid-search configurations) for roulette wheel, linear rank, and tournament selection modes of the genetic algorithm (33 runs per selection mode).

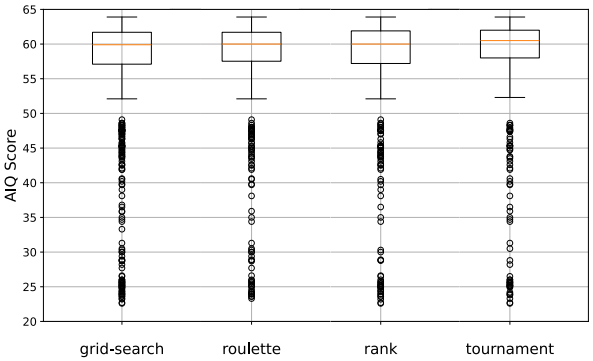
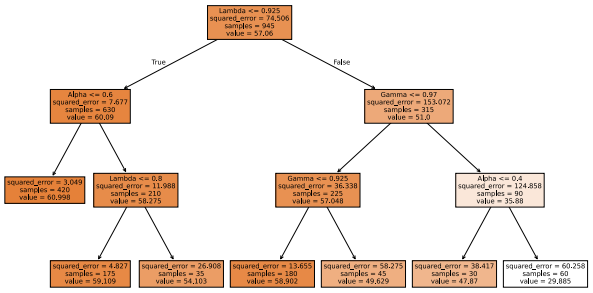


Fig. 1: Box plots showing the differences in distributions of AIQ achieved by Q-learning configurations tested by the grid-search and the genetic algorithm with roulette wheel, linear rank, and tournament selection. The same search granularity was used in all settings.

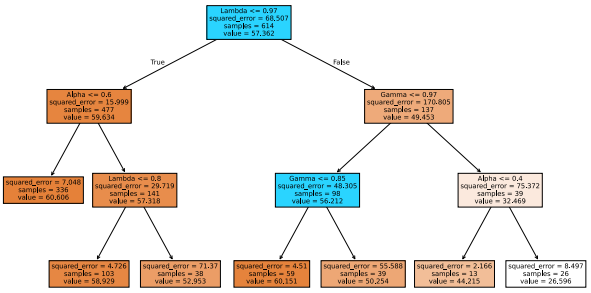
To investigate what part of the configuration space is covered, we constructed box plots that can be seen in Figure 1. Surprisingly, medians, quartiles and variability of AIQ between the grid-search and the different selection modes of GAHS are quite close to each other. As expected, GAHS focuses less on the lower performing configurations. Tournament selection sticks out a little bit with slightly higher median and quartiles and even less focus on the lower performing configurations.

A closer look on the covered configuration space is offered by exploratory regression trees displayed in Figure 2. The discovered structure of the trees describing the influence of hyperparameters on the AIQ of Q-learning remains the same between the grid-search and the different selection modes of GAHS. There are only a few small changes in split tests values.

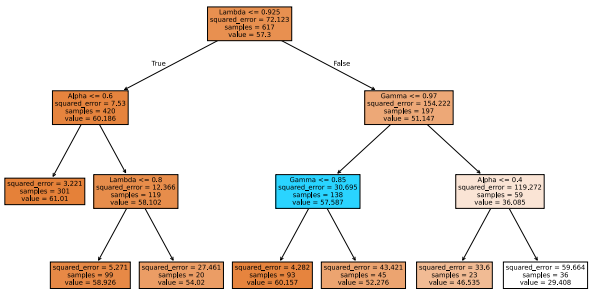
Looking at the overhead of GAHS it reaches (on average) 31.81% for roulette wheel, 31.7% for linear rank, and 35.09% for tournament selection. Such a high overhead is likely caused by the coarse granularity of the search relative to the number of generated configurations and is an expected result of the assumptions.



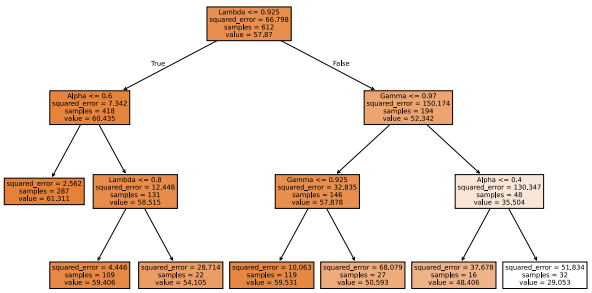
(a) grid-search



(b) roulette wheel



(c) linear rank



(d) tournament

Fig. 2: Regression trees explaining the influence of hyperparameters on the AIQ of Q-learning in the coverage experiment. Differences in split tests between the grid-search and the genetic algorithm with roulette wheel, linear rank, and tournament selection modes are highlighted.

Discussion As can be seen from the results and data analysis, the genetic algorithm hyperparameter search can cover the configuration space reasonably well compared to the grid-search under the assumptions of *the same search granularity* and *at least the same search effort*. While the genetic search covers on average about 65% of the grid-search configurations, the general properties of the AIQ distribution (medians, quartiles and variability) of these configurations remain quite close to the grid-search. This is further confirmed by exploratory regression trees that have the same knowledge structure and differ only in a few hyperparameter-value tests. As expected, the genetic search focuses less on the low-performing configurations. While some statistically significant differences among the selection modes of the genetic algorithm were observed, they were negligible. This is likely due to the constraining assumptions of the experiment that keep a high ratio between the number of configurations searched and the number of configurations that can be generated.

4.2 The Optimality Experiment

In our second set of experiments, we wish to explore the ability of the genetic search to find the optimal configuration. We also want to compare how the configuration space is searched in contrast to the grid-search.

Assumptions In this experiment, we consider only *at least the same search effort* assumption (*A1*) when comparing the results of GAHS to the results of the grid-search. In the case of comparison of GAHS selection modes, the appropriately modified *same search granularity* assumption (*A2*) also applies.

Research Questions

R4 What is the coverage of the agent configurations space compared to the grid-search given only *A1*?

R5 How does the coverage differ between the roulette wheel, linear rank, and tournament selection?

R6 What is the overhead of GAHS due to *A1*, and how does it differ between the roulette wheel, linear rank and tournament selection modes?

Hypotheses

$H3_{strong}$ Given *A1* assumption, the maximum AIQ score found by GAHS is significantly higher than the maximum score found by the grid-search.

$H3_{weak}$ Given *A1* assumption, the maximum AIQ score found by GAHS is not significantly different from the maximum score found by the grid-search.

Characteristic	Configurations for Selection Mode		
	roulette	*linear rank*	*tournament*
Unique	1,001	1,001	1,016
Cached	219	206	215
Total Generated	1,220	1,207	1,231
Indistinguishable Best	225	245	376

Tab. 2: Characteristics of configuration space coverage (as numbers of configurations) for roulette wheel, linear rank, and tournament selection modes of the genetic algorithm (1 run per selection mode).

H4 Given *A1* and *A2* assumptions, the selection modes of GAHS do not have a significant impact on the maximum AIQ score found.

Experiment Settings With the exception of reduced sample size to $N = 2{,}000$ environments for GAHS, the same AIQ test settings apply as in Section 4.1.

The grid-search for Q_λ configurations also remains the same. In the case of GAHS we keep all of its main settings, however, we modify agent references:

λ interval of [0.0,0.995] searched by 0.005;

α interval of [0.005,0.995] searched by 0.005;

ϵ interval of [0.005,0.05] searched by 0.005;

γ interval of [0.5,0.995] searched by 0.005.

As the probability of reusing results from grid-search is extremely low, we run this setup only once for each of the selection modes.

Since the decreased sample size results in wider confidence intervals, we will recompute the configurations that are statistically indistinguishable from the highest scoring one with the sample size of $N = 10{,}000$ environment programs.

Results Characteristics of GAHS for different selection modes are given in Table 2. Scatter plots showing per-hyperparameter coverage can be seen in Figure 3. A comparison of the highest performing configurations discovered by the grid-search and the different selection modes of GAHS is given in Table 3.

Data Analysis Looking at the AIQ scores and their confidence intervals in Table 3, the strong formulation of *H3* can be rejected, while the weak formulation of *H3* holds. Further, *H4* holds as well.

To investigate what part of the configuration space is covered, we constructed box plots that can be seen in Figure 5. There, medians and quartiles of AIQ between the roulette and linear rank selection of GAHS and the grid-search are quite similar, while the variability of AIQ is somewhat lower for the grid-search. The tournament selection sticks out a bit with somewhat higher median and quartiles of AIQ. This corresponds with the pronounced differences among the numbers of indistinguishably best configurations (i.e. configurations with CI overlap with the maximum AIQ) as shown in Table 2. As expected, GAHS focuses less on the lower performing configurations.

A more detailed view on the searched configuration space is given by the exploratory regression trees that can be seen in Figure 4. The structure of learned knowledge differs widely. For the configurations discovered by GAHS, trees highlight the importance of α and in some cases also ϵ. Knowledge of detrimental effects of high λ and (in some cases) high γ is, however, still present.

Looking at the overhead of GAHS, it reaches 5.93% for roulette wheel and linear rank, and 7.51% for tournament selection in terms of unique configurations actually tested. In terms of total configurations generated it reaches 29.1% for roulette wheel, 27.72% for linear rank, and 30.26% for tournament selection.

Discussion As can be seen from the results and data analysis, allowing finer search granularity and keeping *at least the same search effort* results in the genetic algorithm hyperparameter search covering mainly the high-performing region of the configuration space while rest is searched sparsely compared to the grid-search. Especially, the tournament selection mode focuses on the high-performing part so much that the general properties of the AIQ distribution get skewed towards the maximum. The focus on the high-performing part of the configuration space is further notable in explanatory regression trees that differ widely in their knowledge structure among the selection modes. The discovered importance of the learning rate setting may be caused by searching its values more thoroughly by GAHS (in contrast to the grid-search) and thus deserves further systematic study to assure representativeness.

5 Discussion and Conclusion

In this paper, we presented the use of a genetic algorithm (Eiben and Smith, 2015) to search for optimal hyperparameter values of reinforcement learning agents in the test of algorithmic IQ (Legg and Veness, 2013). To assess the proposed approach, we conducted a series of experiments with a basic Q-learning agent (Watkins, 1989). This allowed us to perform a systematic grid-search using 945 configurations of the agent as a baseline. First, we focused on the ability of the genetic algorithm to cover this configuration space reasonably well. Then, we investigated its ability to find the optimal configuration in a much finer-grained search. This created two quite opposite scenarios: one in which the genetic

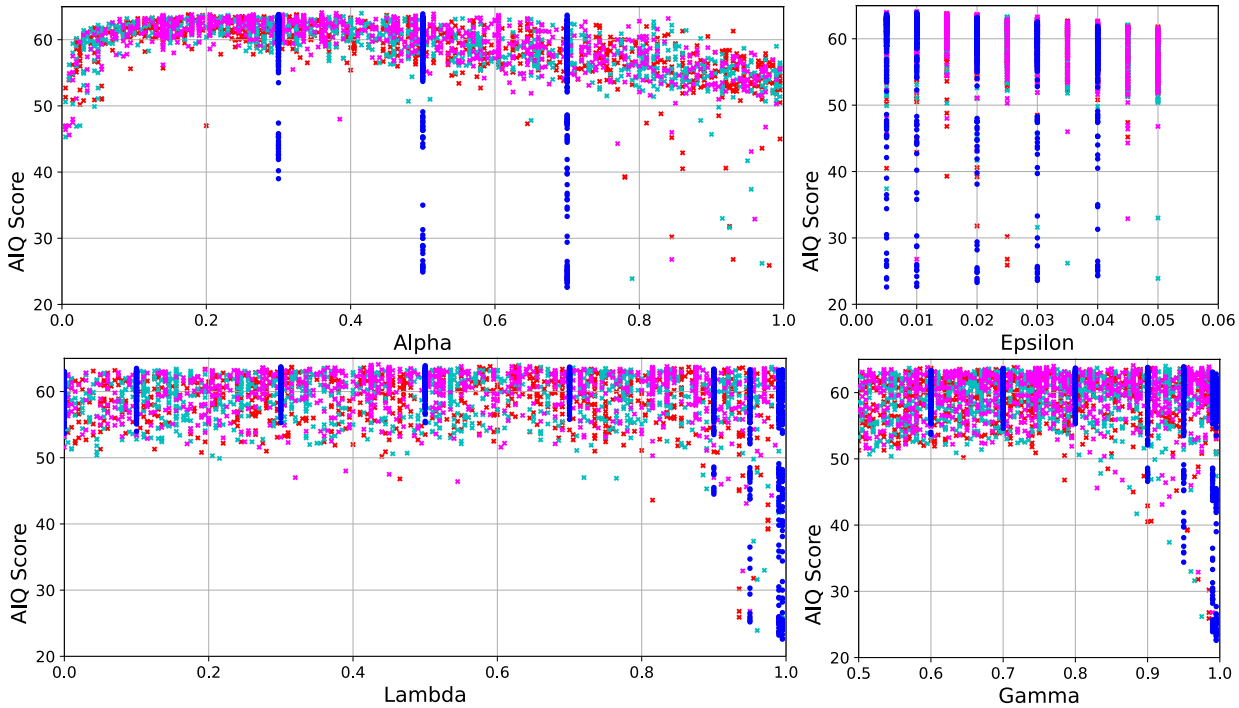
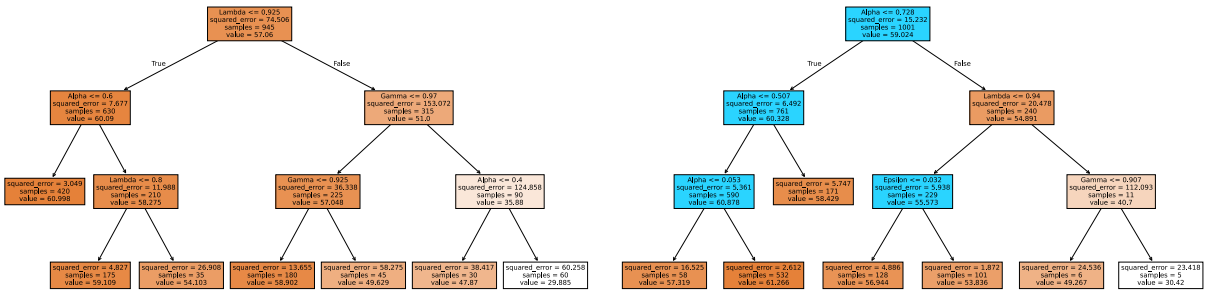
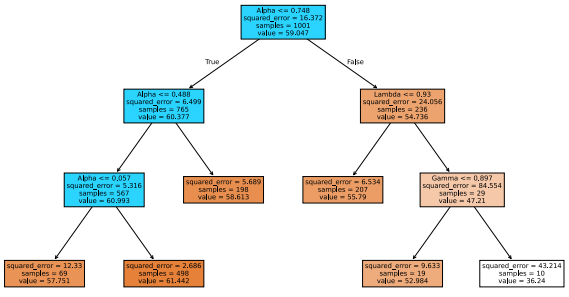


Fig. 3: Scatter plots showing the differences in hyperparameter coverage between the grid-search (•) and the genetic algorithm (×) with roulette wheel (cyan), linear rank (red), and tournament (magenta) selection modes. The search granularity differs between the grid-search and the genetic algorithm.

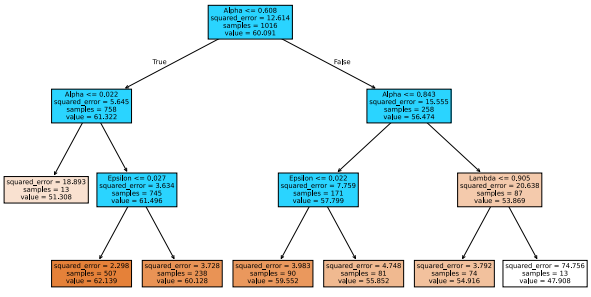


(a) grid-search

(b) roulette wheel



(c) linear rank



(d) tournament

Fig. 4: Regression trees explaining the influence of hyperparameters on the AIQ of Q-learning in the optimise experiment. Differences in knowledge structure between the grid-search and the genetic algorithm with roulette wheel, linear rank, and tournament selection modes are highlighted.

Selection Mode	Results			Q-Learning Configuration					
	AIQ	*HCI*	*SD*	*Q*	α	λ	ϵ	*EDL*	γ
Roulette	63.8	$\pm$0.4	22.0	0.0	0.75	0.19	0.02	0	0.705
Linear Rank	63.8	$\pm$0.4	22.5	0.0	0.77	0.285	0.015	0	0.825
Tournament	63.8	$\pm$0.4	22.1	0.0	0.645	0.335	0.01	0	0.76
Grid-Search	63.9	$\pm$0.4	22.0	0.0	0.5	0.5	0.01	0	0.95

Tab. 3: Comparison of highest performing configurations discovered by the grid-search and roulette wheel, linear rank, and tournament selection modes of the genetic algorithm.

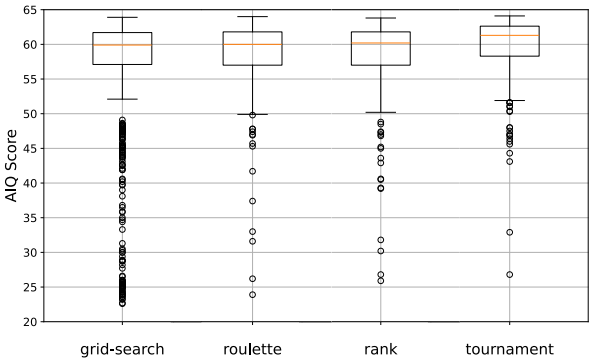


Fig. 5: Box plots showing the differences in distributions of AIQ achieved by Q-learning configurations tested by the grid-search and the genetic algorithm with roulette wheel, linear rank, and tournament selection modes. The search granularity differs between the grid-search and the genetic algorithm.

algorithm has a relatively high search effort compared to search granularity, and the other in which it has a relatively low search effort compared to the granularity. Let us now discuss the findings.

As expected, the experiments showed that the genetic algorithm is capable of finding optimal configurations of the Q-learning agent. This was true both for the coarse-grained as well as for the fine-grained search. In the coarse-grained high-search-effort scenario, the evaluation of the agent remains quite representative of the conducted grid-search. This is unsurprising as about $2/3$ of the searchable agent configurations are actually discovered by the genetic algorithm. Yet, it also means that we can get quite the same picture of the evaluated agent and save about $1/3$ of the computation resources even though the genetic algorithm has a selection bias towards high-performing configurations. In the fine-grained low-search-effort scenario, only the basic properties of the AIQ distribution remain representative of the conducted grid-search and only for roulette wheel and linear rank selection modes of the genetic algorithm. The detailed knowledge about the hyperparameters influence on the agent's AIQ differs from the grid-search. Furthermore, the tournament selection mode manifests the highest selection bias towards high-performing configurations.

As for the evaluated Q-learning agent, the experiments discovered interesting compensatory relationships between its hyperparameters, namely the learning rate α, eligibility traces decay rate λ, and discount factor γ. Extreme upper values of γ or λ result in poor performance when not compensated by a moderate value of the other hyperparameter. Also, α can get quite high in case λ is low and γ not too high. Thus, high-performing configurations span wide regions of the configuration space and are quite common. Our results concern a scenario with relatively long training times. In case of a more short-term scenario, the results may differ.

Overall, using a genetic algorithm is a viable way for reinforcement learning agents hyperparameter optimisation. Combined with a general evaluation method, such as the AIQ test, it can help with discovering useful insights into how to set up these agents in order to achieve a generally high performance. Such evaluation naturally focuses the available computation resources on the high-performing part of the configuration space while necessarily bringing a less representative view on possible performance. Depending on the ratio of genetic search effort to the search granularity, this trade-off can be further fine-tuned.

Acknowledgement

This work was funded by the Internal Grant Agency of Prague University of Economics and Business (F4/41/2023). Computational resources were provided by the project "e-INFRA CZ" (ID:90254) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Appendix

The AIQ test, GAHS module, experiment settings, results and analyses are available from:
`https://github.com/xvado00/AIQ/archive/refs/tags/v2.3.zip`
`https://github.com/xvado00/AIQ-GAHS/archive/refs/tags/v1.0.zip`
`https://github.com/xvado00/GAAHO/archive/refs/tags/v1.0.zip`

References

Chaitin, G. J. (1987). *Algorithmic Information Theory*, vol. 1 from *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 3. ed.

Eiben, A. E., Aarts, E. H. L. and Van Hee, K. M. (1991). Global convergence of genetic algorithms: A markov chain analysis. Schwefel, H.-P. and Männer, R. (ed.), In *Parallel Problem Solving from Nature*, pp. 3–12. Springer, Berlin.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, Heidelberg, 2. ed.

Hernández-Orallo, J. (2017). *Measure of All Minds, The*. Cambridge University Press, Cambridge, 1. ed.

Hernández-Orallo, J. and Dowe, D. L. (2010). Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence*, 174(18):1508–1539.

Jebari, K. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3:333–344.

Legg, S. and Hutter, M. (2007). Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444.

Legg, S. and Veness, J. (2013). An approximation of the universal intelligence measure. Dowe, D. L. (eds.), In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, vol. 7070 from *Lecture Notes in Computer Science*, pp. 236–249. Springer, Berlin, Heidelberg.

Müller, U. (1993). Dev/lang/brainfuck-2.lha in aminet.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Wiley, New York.

Solomonoff, R. J. (1964). A formal theory of inductive inference, part 1 and part 2. *Information and Control*, 7:1–22, 224–254.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 2. ed.

Vadinský, O. (2018). Towards general evaluation of intelligent systems: Using semantic analysis to improve environments in the AIQ test. Iklé, M., Franz, A., Rzepka, R. and Goertzel, B. (ed.), In *Proceedings of the 11th International Conference on Artificial General Intelligence (AGI 2018), Prague, Czech Republic*, vol. 10999 from *Lecture Notes in Artificial Intelligence*, pp. 248–258, Cham. Springer.

Vadinský, O. and Dvořák, M. (2025). Initial evaluation of deep q-learning in the algorithmic intelligence quotient test. Iklé, M., Kolonin, A. and Bennett, M. (ed.), In *Proceeding of the 18th International Conference on Artificial General Intelligence (AGI 2025), Reykjavík, Iceland, Part II*, vol. 16058 from *Lecture Notes in Artificial Intelligence*, pp. 252–263, Berlin. Springer.

Vadinský, O. and Zeman, P. (2024). Towards evaluating policy optimisation agents using algorithmic intelligence quotient test. Nowaczyk, S., Biecek, P., Chung, N. C., Vallati, M., Skruch, P., Jaworek-Korjakowska, J., Parkinson, S. and Nikitas, A. (ed.), In *Proceedings of the 3rd International Workshop on Explainable and Interpretable Machine Learning (XI-ML 23)*, vol. 1947 from *Communications in Computer and Information Science*, pp. 435–451, Cham. Springer.

Watkins, C. (1989). *Learning from Delayed Rewards*. Dissertation, University of Cambridge, Kings College, Cambridge.