Quantum Model Learning Agent

Brian Flynn
QETLabs
School of Physics
University of Bristol
brian.flynn@bristol.ac.uk

Antonio Andreas Gentile
QETLabs
School of Physics
University of Bristol

Raffaele Santagati
Boehringer-Ingelheim Quantum Lab
Vienna, Austria

Nathan Wiebe
Department of Physics
University of Washington

Anthony Laing
QETLabs
School of Physics
University of Bristol

Abstract

Models of quantum mechanical systems capture their interactions, but they are often difficult to retrieve. We describe and implement a Quantum Model Learning Agent (QMLA) to reverse engineer Hamiltonian descriptions from simulated experimental observations and we test its performance. We demonstrate the protocol operating in large model spaces by incorporating a genetic algorithm, devising an objective function inspired by the Elo ratings scheme, which is typically used to rate competitors in games such as chess and football. In a space of over $250,000$ candidate models, the genetic algorithm identifies, in 90% of cases, models with F_1-score > 0.8, when compared with the known target model. By determining the operations which actually occur with respect to those desired, QMLA can be used for the characterisation and calibration of new quantum devices

1 Quantum model learning agent

Recent interest in distilling insight about real quantum systems has resulted in machine learning methodologies such as Quantum Hamiltonian Learning (QHL), for the retrieval of the parameters built into the models assumed to describe the target systems [1, 2, 3, 4, 5, 6]. Meanwhile, model recovery algorithms are gaining traction in the study of solid state systems [7, 8, 9]. QMLA is a framework to reverse engineer models of quantum systems, combining the concepts of individual model training with model search mechanisms.

QMLA was recently introduced and applied in an experimental context [10]; here we extend the protocol to target larger model spaces using simulated target systems, employing a genetic algorithm to search among $250,000$ candidate Hamiltonian models. These represent the generators of the observed dynamics if targeting closed quantum systems, so that the output of QMLA is an approximate but complete model for the system

1.1 Algorithm

For a target quantum system, Q, whose Hamiltonian is given by $\hat{H}_0$, QMLA distills a model $\hat{H}' \approx \hat{H}_0$. Models are characterised by their parameterisation, i.e. their constituent operators, or *terms*, along with corresponding scalar factors. For example, a model consisting of the sum of one-qubit Pauli terms, Eqn. 1,

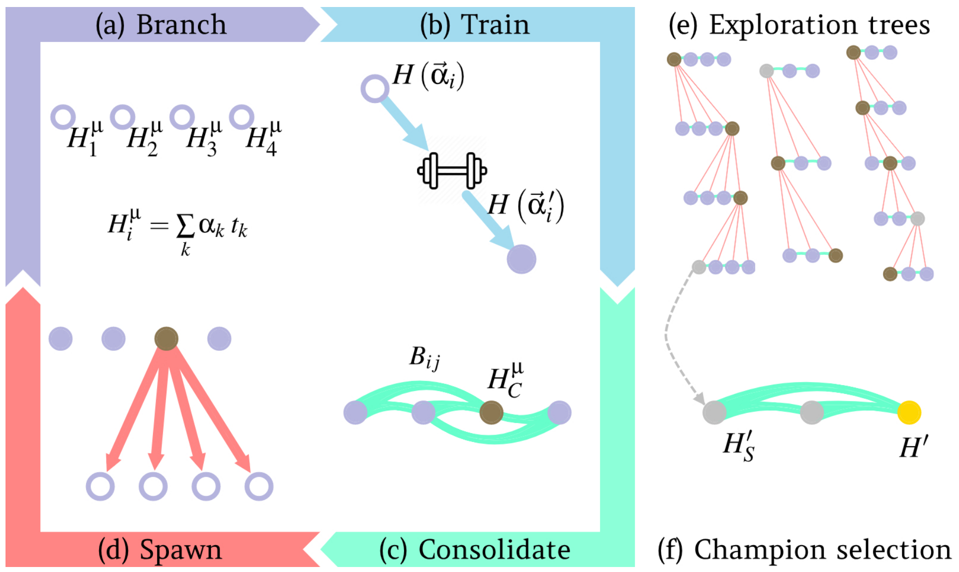

Figure 1: Schematic of QMLA. **(a-d),** model search phase for an Exploration Strategy. **(a)**, Models are placed as (empty, purple) nodes on a branch, μ, where each model is the dot product of parameters $\vec{\alpha_i}$ with terms $\vec{T}$. **(b)**, Each model in the branch is trained according to a subroutine, such as quantum Hamiltonian learning, to optimise $\vec{\alpha}_i$, resulting in the trained $\hat{H}(\vec{\alpha}_i')$ (filled purple node). **(c)**, μ is consolidated, i.e. models are evaluated relative to other models on μ, according to the consolidation mechanism specified by the ES. In this example, pairwise Bayes factors B_{ij} between candidate models $\hat{H}_i, \hat{H}_j$ are computed, resulting in the election of a single branch champion $\hat{H}_C^\mu$ (bronze). **(d)**, A new set of models are *spawned* according to the result of the consolidation stage. These become the nodes of the next branch, iterating back to **a**. **(e-f),** Overview of entire QMLA procedure. **(e)**, The model search phase is captured by an *exploration tree* (ET); Multiple ESs can operate in parallel, e.g. assuming different underlying physics, so there are several ETs. Each ES nominates a champion, $\hat{H}_S'$ (silver). **(f)**, ES champions are consolidated to give the final champion model, $\hat{H}'$ (gold), i.e. the model QMLA's finds as the strongest approximation for the system under study.

$$\hat{H} = \alpha_x \hat{\sigma}^x + \alpha_y \hat{\sigma}^y + \alpha_z \hat{\sigma}^z = \begin{pmatrix} \alpha_x & \alpha_y & \alpha_z \end{pmatrix} \cdot \begin{pmatrix} \hat{\sigma}^x \\ \hat{\sigma}^y \\ \hat{\sigma}^z \end{pmatrix} = \vec{\alpha} \cdot \vec{\mathcal{T}} \tag{1}$$

is characterised by its parameters $\vec{\alpha}$ and terms $\vec{\mathcal{T}}$. QMLA aims primarily to find $\vec{\mathcal{T}}'$, secondarily to find $\vec{\alpha}'$, such that $\hat{H}' = \vec{\alpha}' \cdot \vec{\mathcal{T}}' \approx \hat{H}_0$. In doing so, Q can be completely characterised.

Candidate models $\hat{H}_i$ are *trained* independently: $\hat{H}_i$ undergoes a parameter learning subroutine to optimise $\vec{\alpha}_i$. In this work we use QHL as the parameter learning subroutine [1], although alternative methods can be adopted [11, 5]. The training of individual candidate models involves querying the target system by running customised experiments, and measuring the system in one of its eigenstates, collecting binary projected data processed via interactive quantum likelihood estimation [3][1]. The choice of experimental times for which to evolve the target system is an active area of research, with several heuristic attempts to design informative experiments from which to learn [2, 12]; we use a heuristic algorithm which exploits the most up-to-date knowledge of the candidate's parametrisation [4].

QMLA considers several *branches* μ of candidate models: we can envision individual models as nodes on a branch. Then, at each new branch, models are constructed accounting for the strengths and weaknesses detected on prior branches, so the average approximation of $\hat{H}_0$ should iteratively improve with each branch.

[1]Note: within QHL, we use 500 *experiments* and 2000 *particles* during the training of each model, see [1].

Branches are *consolidated*, meaning that models are evaluated relative to each other, and ordered according to their strength. Models favoured by the consolidation are then used to *spawn* a new set of models, which are placed on the next branch. Bayes factors (BFs) are a useful quantity for distinguishing between models on a pairwise basis: they quantify the relative performance of a given model with respect to a competitor, when both were trained on the same data [13]. Moreover, BFs implicitly favour models of lower paramterisation, i.e. they require strong evidence to favour the model of higher cardinality, automatically granting protection against over-fitting.

The QMLA procedure is depicted in Fig. 1, centred on an iterative *"model search"* phase, which is directed by an Exploration Strategy (ES). The core task of an ES is to specify the spawning mechanism, i.e. how to generate new candidate models based on the information available. In addition, the ES directs the QMLA algorithm at several junctions, such as the resources allocated to learning each model. ESs are bespoke to a given situation; usually the next branch of models is built by exploiting the knowledge gained during the previous branch. For example, the single strongest model found in μ can be retained and spawn new models which merely add one further term, i.e. *greedy* spawning. The ES further specifies how QMLA should consolidate each branch, i.e. how to evaluate the relative strength of each model in μ. The structure of the ES is left entirely to the user; in the next section we outline the logic of the ES used in this work.

The model search phase is subject to termination criteria set by the ES, e.g. when a set number of branches, or some convergence threshold, is reached. QMLA then declares the strongest model considered as the champion with respect to the ES, $\hat{H}'_S$. Finally, QMLA can concurrently run multiple ESs, so the final step of QMLA is to consolidate the set $\{\hat{H}'_S\}$, in order to declare a final *champion model*, $\hat{H}'$.

We emphasise that QMLA is a generic framework with numerous modular aspects, such as the model training subroutine, model consolidation strategy, and model spawning mechanism, all of which are specified by the user-designed ES. In this work we focus on an ES embedding a genetic algorithm, but note that alternative ESs can be used for other use cases, for example the *greedy* model spawning mechanism in [10].

2 Genetic algorithm

We devise a Genetic Algorithm (GA) to implement an adaptive ES search capable of global optimisation across large model spaces. N_m models are proposed in a single *generation* of the GA, which runs for N_g generations, in clear analogy with the branches of QMLA. We outline the usual process of GAs, along with our choice of selection, cross-over and mutation mechanisms, in Appendix A.

First we define the set of *terms*, $\mathcal{T} = \{\hat{t}_i\}$, which may occur in $\hat{H}_0$, e.g. by listing the available couplings in our target system. $\hat{t}_i$ are mapped to binary genes such that chromosomes can be written as binary strings and immediately interpreted as models, e.g. in Table Apdx 1. We study a 4-site lattice with arbitrary connectivity under a Heisenberg–XYZ model, omitting the transverse field for simplicity, i.e. any two sites i, j can be coupled by any of $\{\hat{\sigma}_i^x\hat{\sigma}_j^x,\ \hat{\sigma}_i^z\hat{\sigma}_j^z,\ \hat{\sigma}_i^z\hat{\sigma}_j^z\}$. There are therefore $|\mathcal{T}| = 3 \times \binom{4}{2} = 18$ binary genes, resulting in a model space of $2^{18} \approx 250,000$ candidates.

We can gauge the performance of QMLA's model search by the quality of candidate models produced at each generation: our primary figure of merit is F_1-score, $f \in (0, 1)$, which we use as a proxy for the quality of each candidate model $\hat{H}_i$. f indicates the degree to which $\hat{H}_i$ captures the physics of the target system: $f = 0$ indicates that $\hat{H}_i$ shares no terms with $\hat{H}_0$, while $f = 1$ is found uniquely for $\hat{H}_i = \hat{H}_0$. Note that here we are able to compute f for candidate models because the target $\hat{H}_0$ is simulated, i.e. we know the true terms $\mathcal{T}_0$; this would not be available for a real system with unknown $\hat{H}_0$, but is useful for the analysis of the algorithm itself.

In the absence of a natural objective function, we design a rating system for models, based on the Elo ratings scheme used in chess and sports rankings [14]. Each candidate, $\hat{H}_i$, is assigned a number of *Elo points* R_i; during the consolidation stage of QMLA, direct model comparisons facilitate transfer of Elo points from the inferior to the stronger model, weighted by the evidence of the comparison. This ratings scheme is detailed in Appendix A.1.

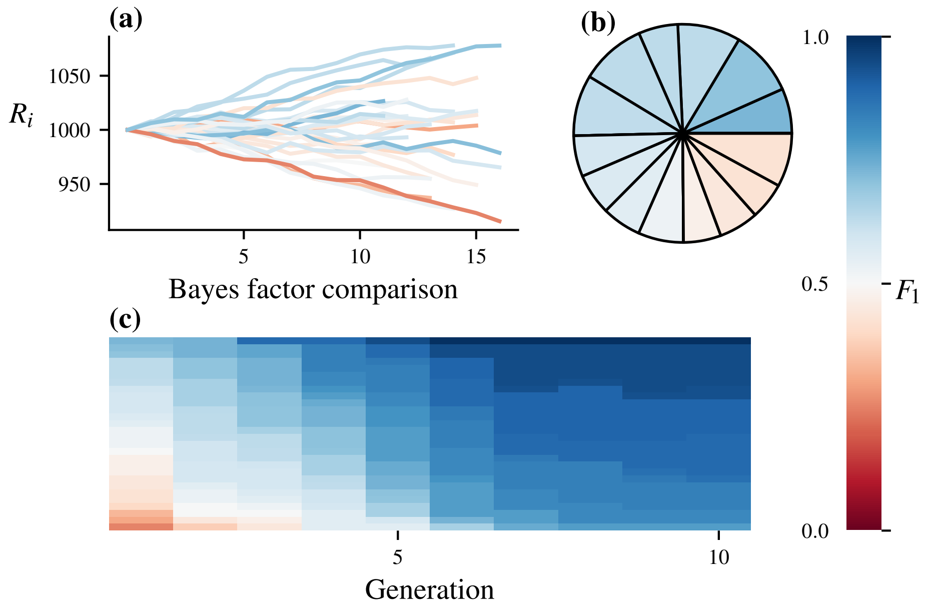

Figure 2: A single instance of QMLA using the genetic exploration strategy, with $N_m = 28$. In all subplots we are showing sets of candidate models, coloured by their F_1-score, to indicate their overlap with the target model. **(a-b), The first generation of the genetic algorithm.** (**a**), The rating, R_i of each candidate model, $\hat{H}_i$, is updated according to its direct comparison with alternatives $\hat{H}_j$. Comparisons occur through pairwise Bayes factors, where the inferior model transfers some of its *Elo points* to the favoured model. Note the Bayes factor is not computed between every pair of models, but instead every model is compared against a random subset ($\sim$ 50%) of contemporaries within the generation. The series of Bayes factor calculations allow us to distinguish statistically strong from weak models: the highest-rated 14 models are considered for parental selection. (**b**), Models' selection probabilities, where parents for the next generation are chosen via a roulette scheme. Each model's probability of being chosen as a parent, during the generation of new models, is based on their final rating from **a**. The models shown in the pie chart are the strongest 14 candidates from the first generation, depicted in **c**. **(c), Progression of the gene pool across $N_g = 10$ generations.** Each tile represent a model; models at generation g are spawned from parents which performed relatively well during generation $g - 1$. Models are sorted by their F_1-score, not the ratings shown in **a**.

In Fig. 2 we first show the inner-proceedings of an individual generation, as well as the results of a single QMLA instance using this genetic algorithm approach as the ES. We then show the combined results of 50 independent instances, in Fig. 3, showing the frequency with which models of different quality (F_1-score) are nominated as $\hat{H}'$, as well as the rates with which each individual $\hat{t}_i \in \mathcal{T}$ is detected. QMLA correctly identifies $\hat{H}_0$ precisely in 32% of instances, with 90% of instances resulting in $\hat{H}'$ with $f \geq 0.8$, while the identification rate of all $\hat{t}_i \in \vec{\mathcal{T}}_0$ are significantly higher than $\hat{t}_i \notin \vec{\mathcal{T}}_0$.

This adaptive ES may prove a useful application of QMLA in the domain of device calibration, in particular to characterise some untrusted quantum simulator. That is, by using the simulator to implement some target $\hat{H}_0$, QMLA can identify which operator is *actually* implemented. For instance, implementation of a four–qubit model relies on high-fidelity two-qubit gates between arbitrary qubit pairs, and QMLA can effectively reconstruct which operations were and were not faithfully computed.

3 Conclusion

We have shown that our proposed protocol, QMLA, can be used to find an approximate model to characterise quantum systems. By facilitating the construction of models according to users'

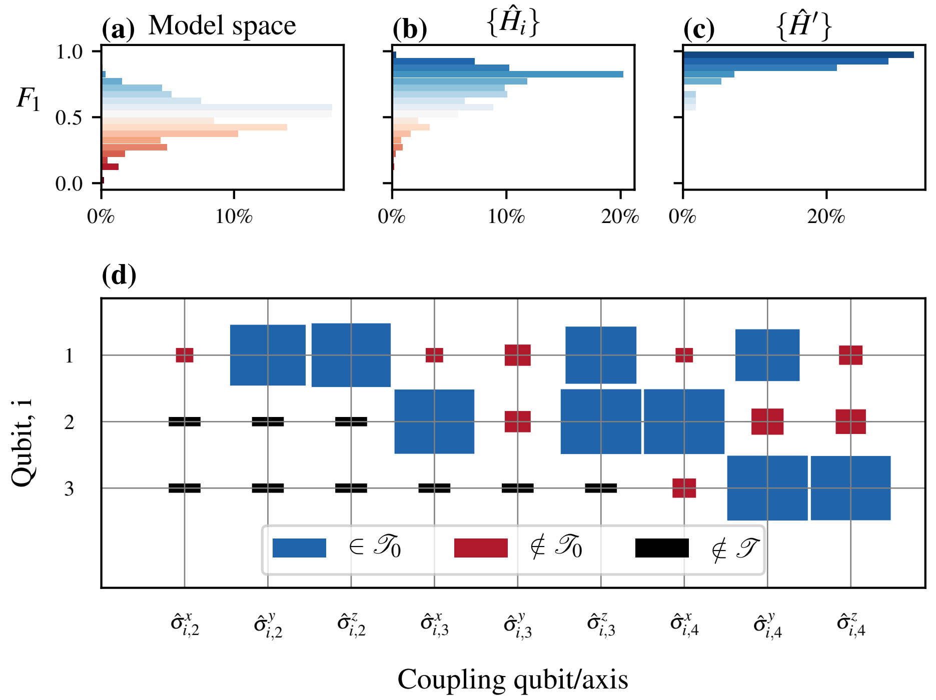

Figure 3: **50 instances of QMLA genetic algorithm.** (**a**), The model space contains $2^{18} \approx 250,000$ candidate models; normally distributed around $f = 0.5 \pm 0.14$. **(b)**, The models explored during the model search of all instances combined, $\{\hat{H}_i\}$, show that QMLA tends towards stronger models overall. **(c)**, Champion models from each instance, showing QMLA finds strong models in general, and in particular finds the true model $\hat{H}_0$ (with $f = 1$) in 32% of cases. **(d)**, Hinton diagram where the size of the blocks show the frequency with which terms are found, while the colour indicates whether that term was in the true model ($\in \mathcal{T}_0$, blue) or not ($\notin \mathcal{T}_0$, red).

specific needs, as well as modular subroutines for training and comparing models, QMLA provides a framework for the characterisation of *grey-box* quantum systems, where users have some knowledge of the range of possible interactions. Essentially, the user may specify the system Q to target, then QMLA – using a trusted (classical or quantum) simulator – determines a model consistent with data measured from Q.

The demonstrated application of this protocol concerns Q as a new, untrusted quantum simulator, whose operation the user wishes to calibrate. QMLA can assess whether the device faithfully implements an encoded $\hat{H}_0$, deducing the operations performed otherwise. This allows for the calibration of devices composed of superconducting qubits, for example, by characterising whether two distant qubits are coupled as intended. Future applications of the framework envision distinguishing between physical regimes, such as whether a target system is best described by Ising, Heisenberg or Hubbard models.

Broader Impact

Near term quantum devices are likely to suffer from fabrication and design imperfections, requiring extensive validation before they can offer computational and metrological advantage. QMLA will benefit the wider development of quantum technologies, by aiding in the calibration, verification and characterisation of these noisy intermediate scale quantum processors and sensors. We intend to provide an open source tool box which can interface with any quantum hardware to retrieve the operators actually implemented, with respect to those desired, allowing for straightforward, automatic calibration of new devices.

	Model	Chromosome					
	$\vec{\mathcal{T}}$	$\hat{\sigma}^x_{(1,2)}$	$\hat{\sigma}^z_{(1,2)}$	$\hat{\sigma}^y_{(2,3)}$	$\hat{\sigma}^x_{(2,3)}$	$\hat{\sigma}^y_{(2,3)}$	$\hat{\sigma}^x_{(2,3)}$
p_A	$(\hat{\sigma}^x_{(1,2)} \quad \hat{\sigma}^z_{(1,2)} \quad \hat{\sigma}^y_{(2,3)})$	1	0	1	0	1	0
p_B	$(\hat{\sigma}^z_{(1,2)} \quad \hat{\sigma}^y_{(2,3)} \quad \hat{\sigma}^z_{(2,3)})$	0	0	1	0	1	1
c_A	$(\hat{\sigma}^x_{(1,2)} \quad \hat{\sigma}^z_{(1,2)} \quad \hat{\sigma}^y_{(2,3)} \quad \hat{\sigma}^z_{(2,3)})$	1	0	1	0	1	1
c_B	$(\hat{\sigma}^z_{(1,2)} \quad \hat{\sigma}^y_{(2,3)})$	0	0	1	0	1	0
c'_B	$(\hat{\sigma}^z_{(1,2)} \quad \hat{\sigma}^x_{(2,3)} \quad \hat{\sigma}^y_{(2,3)})$	0	0	1	$\boxed{1}$	1	0

Table Apdx 1: Mapping between models and chromosomes used by a genetic algorithm. Example shown for a three-qubit system with six possible terms. Model terms are mapped to binary genes: if the gene registers 0 then the corresponding term is not present in the model, and if it registers 1 the term is included. The top two chromosomes are *parents*, $p_A = 101010$ (orange) and $p_B = 001011$ (blue): they are mixed to spawn new models. We use a one–point cross over, shown at their halfway points: the first half of p_A is mixed with the second half of p_B to produce two new children chromosomes, c_A, c_B. Mutation occurs probabilistically: each gene has a 25% chance of being mutated, e.g. a single gene (red) flipping from $0 \rightarrow 1$ to mutate c_B to c'_B. p_A, p_B were selected from the population according to the their success in previous generations. The next generation of the genetic algorithm will then include c_A, c'_B (assuming c_A does not mutate) . To generate N_m models for each generation, $N_m/2$ parent couples are sampled from the previous generation and crossed over.

Appendix

A Genetic algorithm

Here we describe the GA protocol followed, including QMLA-specific aspects. All available Hamiltonian terms compose the set $\vec{\mathcal{T}}$, such that models are uniquely identified by chromosomes, written as bit strings where each bit is a *gene* corresponding to a single term. With $\left|\vec{\mathcal{T}}\right| = n$, there are 2^n valid chromosomes (bit strings), which constitute the *population* $\mathcal{P}$. The QMLA model search phase is then simply the standard GA procedure:

```
1. Randomly select $N_m$ chromosomes from $\mathcal{P}$
   (a) $\{\hat{H}_i\} \leftarrow \mathbb{H}_0$
2. Assign generation $\mu \leftarrow \mathbb{H}_0$
3. For each $\hat{H}_i^\mu \in \mu$
   (a) Train $\hat{H}_i^\mu$ via QHL
   (b) Apply objective function: $g(\hat{H}_i^\mu)$
   (c) Assign selection probability
       i. $s_i \sim g(\hat{H}_i^\mu)$.
       ii. $s_i$ is relative to other models in $\mu$.
4. Spawn new models
   (a) Select two parents from $\mu \leftarrow \hat{p}_A, \hat{p}_B$
```

```
            i. according to $s_i$ of each model
        (b) Cross-over parents to produce children
            i. $C(\hat{p}_A, \hat{p}_B) \leftarrow \hat{c}_A, \hat{c}_B$
        (c) Probabilistically mutate children chromosomes
            i. $M(\hat{c}_A, \hat{c}_B) \leftarrow \hat{c}_A, \hat{c}_B$
    5. Collect newly proposed children chromosomes
        (a) $\{\hat{c}_i\} \leftarrow \mathbb{H}^{\mu+1}$
    6. Determine top model from this generation
        (a) $\hat{H}_C^{\mu} = \max\limits_{g(\hat{H}_i)} \{\hat{H}_i^{\mu}\}$.
    7.
        (a) if $\hat{H}_C^{\mu} == \hat{H}_C^{\mu-4}$
            i. strongest model not improved upon in five generations
            ii. move to 8.
        (b) else if $\mu == xN_g$
            i. move to 8.
        (c) else
            i. $\mu \leftarrow \mu + 1$
            ii. return to 3.
    8. Nominate the strongest model from the final generation
        (a) $\hat{H}' = \max\limits_{g(\hat{H}_i)} \{\hat{H}_i^{\mu}\}$.
```

We set $N_g = 32, N_m = 28$, and use *elitism* rules, whereby the top two models on μ are automatically considered on $\mu + 1$. If the top model is unchanged for five generations, we terminate the model search and delcare that elite model as $\hat{H}'$, otherwise the top model from the final generation is nominated as $\hat{H}'$.

A.1 Objective function

We do not have access to an absolute measure of model quality, and therefore instead devise an objective function based on the information available, and in particular the pairwise comparisons between candidate models, i.e. Bayes factors B_{ij} between $\hat{H}_i, \hat{H}_j$ [13]. Where possible, it is preferable to minimise the number of Bayes factor calculations due to the large overhead, so our objective function aims to rank models with incomplete comparison information. We take inspiration from games such as chess and football, where individual competitors can be assigned an absolute rating based on the matches played, without all individuals having met. In particular, we use a modified Elo rating scheme [14], where we weight the points–transfer between individuals by $log_{10}B_{ij}$. The rating of $\hat{H}_i$, following comparison with $\hat{H}_j$, is given by

$$R_i' = R_i + \log_{10}(B_{ij}) \left(S_i - \frac{1}{1 + 10^{\frac{R_j - R_i}{400}}} \right), \tag{2}$$

where R_k are the initial ratings of the candidates, and S_i is the binary *score* of the BF comparison, from the perspective of $\hat{H}_i$, i.e. $S_i = 1, S_j = 0 \iff B_{ij} > 1$. After a series of pairwise comparisons, models' ratings are mapped to an objective function by subtracting the rating of the poorest performing model on μ:

$$g(\hat{H}_i) = R_i^{\mu} - R_{min}^{\mu}. \tag{3}$$

Finally all $g(\hat{H}_i^{\mu})$ are normalised so that each model's fitness can be interpreted as selection probability for the branch μ, in the following selection mechanism.

A.2 Selection mechanism

Parents are selected from μ through *roulette* selection, easily visualised as pie charts representing each available parent's probability of selection, as in Fig. **??**(a). We truncate to include only the strongest $\frac{N_m}{2}$ models to ensure high quality offspring. Parents produce offspring through a *one-point* crossover, i.e. the first $\frac{1}{k}$ genes from p_A are conjoined with the latter $\frac{k-1}{k}$ of p_B, and we choose $k \in \left(\frac{n}{4}, \frac{3n}{4}\right)$ randomly for each pair of parents. Given that k is not fixed, the same pair of parents can be selected multiple times with different k. Each gene from each child chromosome has the opportunity to mutate, according to a 25% mutation probability. Within μ, models are assigned selection probabilities $\{\hat{H}_i : s_i\}$, reflecting their fitness relative to contemporary models, assessed through the objective function. e.g. if the model space uses a chromosome of length 6, $\left(\hat{H}_i, \hat{H}_j\right)$ can produce offspring via one–point crossovers at positions 3 or 4; see Table. Apdx 1 for an example of crossover at position 3.

A.3 Crossover mechanism

We use a standard one-point crossover as outlined in Table Apdx 1. Given parents p_A, p_B, chosen via the selection mechanism outlined above, the first $\frac{1}{k}$ genes from p_A are conjoined with the latter $\frac{\kappa-1}{\kappa}$ of p_B, where we choose $\kappa \in \left(\frac{n}{4}, \frac{3n}{4}\right)$ randomly for each pair of parents. Given that κ is not fixed, the same pair of parents can be selected multiple times with different κ.

Following crossover, each gene in the childgren chromosomes have a mutation probability of 25%, although it was found that this rate did not drastically effect the outcome of the genetic algorimth, compared with rates of 10%, 15% and 20%.

References

[1] Christopher E Granade et al. "Robust online Hamiltonian learning". In: *New Journal of Physics* 14.10 (Oct. 2012), p. 103013. DOI: `10.1088/1367-2630/14/10/103013`.

[2] C Ferrie, Christopher E Granade, and D G Cory. "How to best sample a periodic probability distribution, or on the accuracy of Hamiltonian finding strategies". In: *Quantum Information Processing* 12.1 (Apr. 2012), pp. 611–623. DOI: `10.1007/s11128-012-0407-6`.

[3] Nathan Wiebe et al. "Quantum Hamiltonian learning using imperfect quantum resources". In: *Physical Review A* 89.4 (2014), p. 042314.

[4] N Wiebe et al. "Hamiltonian Learning and Certification Using Quantum Resources". In: *Physical Review Letters* 112.19 (May 2014), pp. 190501–5. DOI: `10.1103/PhysRevLett.112.190501`.

[5] Eyal Bairey, Itai Arad, and Netanel H Lindner. "Learning a local Hamiltonian from local measurements". In: *Physical review letters* 122.2 (2019), p. 020504.

[6] Tim J Evans, Robin Harper, and Steven T Flammia. "Scalable bayesian hamiltonian learning". In: *arXiv preprint arXiv:1912.07636* (2019).

[7] Gus LW Hart et al. "Evolutionary approach for determining first-principles hamiltonians". In: *Nature materials* 4.5 (2005), pp. 391–394.

[8] Volker Blum et al. "Using genetic algorithms to map first-principles results to model Hamiltonians: Application to the generalized Ising model for alloys". In: *Physical Review B* 72.16 (2005), p. 165113.

[9] Miles Cranmer et al. "Discovering symbolic models from deep learning with inductive biases". In: *Advances in Neural Information Processing Systems* 33 (2020).

[10] Antonio A. Gentile et al. *Learning models of quantum systems from experiments*. 2020. arXiv: `2002.06169 [quant-ph]`.

[11] Sheng-Tao Wang, Dong-Ling Deng, and Lu-Ming Duan. "Hamiltonian tomography for quantum many-body systems with arbitrary couplings". In: *New Journal of Physics* 17.9 (2015), p. 093017.

[12] Raffaele Santagati et al. "Magnetic-field learning using a single electronic spin in diamond with one-photon readout at room temperature". In: *Physical Review X* 9.2 (2019), p. 021019.

[13] Robert E Kass and Adrian E Raftery. "Bayes factors". In: *Journal of the american statistical association* 90.430 (1995), pp. 773–795.

[14] Arpad E Elo. *The rating of chess players, past and present*. Arco Pub., 1978.