Doctorate of Philosophy

Schrödinger's Catwalk

Machine learning methods to distill models of quantum systems

Brian Flynn

University of Bristol

April, 2021

ABSTRACT

Quantum technologies exploit quantum mechanical processes to achieve outcomes beyond the reach of classical machinery. One of their most promising applications is quantum simulation, whereby particles, atoms and molecules can be examined thoroughly for the first time, having been beyond the scope of even the most powerful supercomputers.

Models have been useful tools in understanding physical systems: these are mathematical structures encoding physical interactions, which allow us to predict how the system will behave under various conditions. Models of quantum systems are particularly difficult to design and test, owing to the huge computational resources required to represent them accurately. In this thesis, we introduce and develop an algorithm to characterise quantum systems efficiently, by inferring a model consistent with their observed dynamics. The *Quantum Model Learning Agent* (QMLA) is an extensible framework which permits the study of any quantum system of interest, by combining quantum simulation with state of the art machine learning. QMLA iteratively proposes candidate models and trains them against the target system, finally declaring a single model as the best representation for the system of interest.

We describe QMLA and its implementation through open source software, before testing it under a series of physical scenarios. First, we consider idealised theoretical systems in simulation, verifying the core principles of QMLA. Next, we incorporate strategies for generating candidate models by exploiting the information QMLA has gathered to date; by incorporating a genetic algorithm within QMLA, we explore vast spaces of valid candidate models, with QMLA reliably identifying the precise target model. Finally, we apply QMLA to *realistic* quantum systems, including operating on experimental data measured from an electron spin in a nitrogen vacancy centre.

QMLA is shown to be effective in all cases studied in this thesis; however, of greater interest is the platform it provides for examining quantum systems. QMLA can aid engineers in configuring experimental setups, facilitate calibration of near term quantum devices, and ultimately enable complete characterisation of natural quantum structures. This thesis marks the beginning of a new line of research, into automating the understanding of quantum mechanical systems.

DECLARATION OF AUTHORSHIP

This dissertation is submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctorate of Philosophy in the Faculty of Science.

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's *Regulations and Code of Practice for Research Degree Programmes* and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Signed:

_____

Date:

_____

Word count: $\sim 50,000$

ACKNOWLEDGEMENTS

For

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LISTINGS

ACRONYMS

AIC Akaike information criterion, Eq. (1.5)
AICC Akaike information criterion corrected, Eq. (1.6)

BF Bayes factor, **??**
BFEER Bayes factor enhanced Elo ratings, Section 1.2.7
BIC Bayesian information criterion, Eq. (1.9)

EDH experiment design heuristic, **??**
ES exploration strategy, **??**
ET exploration tree, **??**

FN false negatives, **??**
FP false positives, **??**

GA genetic algorithm, **??**
GES genetic exploration strategy

IQLE interactive quantum likelihood estimation, **??**

ML machine learning, **??**

NVC nitrogen-vacancy centre, **??**

OF objective function, **??**

QHL quantum Hamiltonian learning, **??**.
QMLA Quantum Model Learning Agent, **??**.

TLTL total log total likelihood, **??**
TN true negatives, **??**
TP true positives, **??**

GLOSSARY

Q	Quantum system which is the target of QMLA, i.e. the system to be characterised.
$\hat{H}_0$	True model for the target system, Q; i.e. the Hamiltonian model form which QMLA is attempting to retrieve in a given instance.
F_1**-score**	F_1-score, measure of model quality with respect to $\hat{H}_0$, $f \in [0,1]$, Section 1.1.2.
champion	See champion model.
champion model	The model deemed by QMLA as the most suitable for describing the target system.
chromosome	A single candidate, in the space of valid solutions to the posed problem in a genetic algorithm, **??**.
Elo rating	Points transfer system for quantifying relative ability of candidates in a competitive domain, modified for use as an objective function within the genetic algorithm of QMLA, Section 1.2.7
expectation value	Average outcome expected by measuring an observable of a quantum system many times, **??**.
experiment	Experiment performed upon Q when training a model through QHL, **??**.
gene	Individual element within a chromosome.
Hamiltonian	Mathematical structure which captures all interactions to which a quantum system is subject, **??**.
hyperparameter	Variable within an algorithm that determines how the algorithm itself proceeds.
instance	A single implementation of the QMLA algorithm, resulting in a nominated champion model.
likelihood	Value that represents how likely a hypothesis is; usually used in the context of likelihood estiamation, **??**.
model	The mathematical description of some quantum system, **??**.

model space Abstract space containing all descriptions (within defined constraints such as dimension) of the system as models, **??**.

particle Sampled from prior distribution during model training through QHL, each particle gives a parameterisation corresponding to a unique hypothesis about Q, **??**.

probe Input state, $|\psi\rangle$, into which the target system is initialised, before unitary evolution, **??**.

run Collection of QMLA instances, usually targeting the same system with the same initial conditions.

success rate Fraction of instances within a run where QMLA nominates the true model as champion.

true model The model which correctly describes the target system. $\hat{H}_0$ is known for simulated Q but not known precisely for experimental systems.

LIST OF PUBLICATIONS

The work presented in this thesis has appeared[1] publicly in several formats.

Papers

1. *Learning models of quantum systems from experiments.* A.A. Gentile, Brian Flynn, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, R. Santagati and A. Laing. arXiv preprint arXiv:2002.06169 (2020); accepted *Nature Physics* (2021); referred to throughout as [1].
2. *Quantum Model Learning Agent: quantum systems' characterisation through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe and A. Laing. Reporting outcomes of studies on theoretical systems as described in this thesis. In Preparation (2021); referred to throughout as [2].
3. *Quantum Model Learning Agent: Python framework for characterising quantum systems.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe and A. Laing. Technical manuscript detailing software implementation. In Preparation (2021).

Software

4. *QMLA: Python framework for the reverse engineering of Hamiltonian models of quantum systems through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe, S. Paesani, C. E. Granade, and A. Laing. Codebase; open sourced via Github repository; referrred to throughout as [3].
5. *Quantum Model Learning Agent*. Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe, S. Paesani, C. E. Granade, and A. Laing. Documentation for codebase; referrred to throughout as [4].

Conference Proceedings - Talks [2]

6. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Techniques in Machine Learning, Online, 2020.

1 Or will appear in the near future.
2 Note: only conferences proceedings presented by the author are included.

7. *Learning models of quantum systems from experiments.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Bristol Quantum Information Technologies, Online, 2020.
8. *Quantum Model Learning: characterizing quantum systems through machine learning*. Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Techniques in Machine Learning, Daejeon, South Korea, 2019.
9. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Engineering Centre for Doctoral Training Conference, Bristol, UK, 2019. Awarded *Talk prize*.

Conference Proceedings - Posters

10. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Machine Learning for Quantum, IOP Conference, Online, 2021. Awarded *Poster prize*.
11. *Quantum Model Learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Machine Learning for Quantum Technologies, Erlangen, Germany, 2019.
12. *Quantum Model Learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Bristol Quantum Information Technologies, Bristol, UK, 2019.
13. *Quantum Model Learning: characterizing quantum systems through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Engineering Centre for Doctoral Training Conference, Bristol, UK, 2018. Awarded *Poster prize*.

INTRODUCTION

Part I

CONTEXTUAL REVIEW

Part II

ALGORITHMS

Part III

THEORETICAL STUDY

1 GENETIC ALGORITHMS

The Quantum Model Learning Agent (QMLA) framework lends itself easily to the family of metaheuristic optimisation techniques called *evolutionary algorithms*, where individuals, sampled from a population of candidates, are considered as solutions to the given problem. Candidates are batched in *generations*, such that iterative generations aim to efficiently search the available population by mimicing biological evolutionary mechanisms [5]. In particular, we develop an exploration strategy (ES) which incorporates a genetic algorithm (GA) in the construction of models; GAs are a subset of evolutionary algorithms where candidate solutions are expressed as strings of numbers representing some configuration of the system of interest [6]. We describe the concepts of GAs in **??**, so we begin this chapter by describing the adaptations which allow us to build a genetic exploration strategy (GES) within QMLA.

1.1 ADAPTATION TO QMLA FRAMEWORK

Unlike the generic aspects of GAs described in **??**, in the context of QMLA, we must deviate from default mechanisms. The overarching goal of QMLA – to characterise some black box quantum system, Q – proceeds by designing and performing experiments upon Q which enable us to improve the modelling of $\hat{H}_0$. Nothing described so far provides a natural objective function (OF), upon which GAs rely to assess the suitability of candidates relative to contemporary candidates. We can not assume full knowledge of $\hat{H}_0$ while generating candidates models, so we can not simply invoke some loss function with respect to the target model, for example. Instead, we must devise schemes which exploit the knowledge we *do* have about each candidate $\hat{H}_j$, which is the primary challenge in building a GES. We propose and discuss a number of options in Section 1.2. Common to all proposed OFs, however, is that candidates should be trained before evaluation, so that their assessment is based on their actual power in describing Q, rather than some initial parameterisation which may not capture their potential capability. This is a tenet of QMLA: for each candidate $\hat{H}_j(\vec{\alpha}_j)$, we use a subroutine to optimise $\vec{\alpha}_j$; as in earlier applications of QMLA, for this study we rely on quantum Hamiltonian learning (QHL) as the parameter optimisation subroutine.

Ultimately, the conceived role of a GA within QMLA is to generate the sets of models to place on successive branches[1] of an exploration tree (ET) as depicted in **??**. The apparatus within QMLA which facilitates novel model generation techniques is the exploration strategy (ES). Here we will design an ES which acts in cooperation with a GA. The ES specifies that consolidation of a generation μ involves evaluating the *fitness*, g_i, of each candidate, $\hat{H}_i$, via the chosen OF. The GA then maps $\{g_i\}$ of each $\hat{H}_i \in \mu$ to a selection probability, and composes new candidates via

1 Branches in QMLA and generations of the genetic algorithm are equivalent here.

crossover, **??**. Recall from **??**, that we capture the space of available terms as $\mathcal{T}$, i.e. we list – in advance – the feasible terms which may be included in candidate models[2], with $N_t = |\mathcal{T}|$ the number of terms considered. QMLA is then an optimisation algorithm, attempting to find the set $\mathcal{T}'$ which *best* represents the true terms $\mathcal{T}_0$. Note, this does not require identification of the precise true model to be successful, as insight can be gained from approximate models which capture the physics of Q. We introduce metrics for success in Section 1.1.2. We recognise the limitations this structure imposes: we can only identify terms which were conceived in advance; this may restrict QMLA's applicability to entirely unknown systems, where such a primitive set can not even be compiled.

The structure of the overall QMLA algorithm (recall **??**) is unchanged. In a GES:

- Branches: models are still grouped in branches, here called generations.
- Training: models are still trained, again through QHL.
- Consolidation: all models are evaluated according to the OF (to be described in Section 1.2), so branches are consolidated by ranking models according to their fitness.
- Spawning: new models are spawned through the GA by selecting pairs of parents for crossover, with the resultant offspring models probabilistically mutated.

The design of any ES centres on the implementation of the generate_models subroutine – we summarise the GES's method in Algorithm 1. We can restate the informal description of GAs[3], now in the context of QMLA, as

1. Sample N_m models from $\mathcal{P}$ at random
   (a) this is the first generation, μ.
2. Evaluate each model $\hat{H}_j \in \mu$
   (a) train $\hat{H}_j$ through QHL;
   (b) apply the objective function to assign the model's fitness, g_j.
3. Map the fitnesses of each model, $\{g_j\}$, to selection probabilities for each model, $\{s_j\}$
   (a) e.g. by normalising the fitnesses, or by removing some poorly-performing models and then normalising.
4. Generate the next generation of models
   (a) Reset $\mu = \{\}$;
   (b) Select pairs of parents, $\hat{H}_{p_1}, \hat{H}_{p_2}$, from μ
      i. Each model's probability of being chosen is proportional to their s_j;
   (c) Cross over $\hat{H}_{p_1}, \hat{H}_{p_2}$ to produce children models, $\hat{H}_{c_1}, \hat{H}_{c_2}$
      i. mutate $\hat{H}_{c_1}, \hat{H}_{c_2}$ according to some random probabilistic process;

2 Recall that models impose structure on sets of terms: $\hat{H}_j = \vec{\alpha}_j \cdot \vec{T}_j = \sum_{k \in \{j\}} \alpha_k \hat{t}_k$.

3 First stated on **??**.

ii. $\mu \leftarrow \mu \cup \{\hat{H}_{c_i}\}$, only if $\hat{H}_{c_i}$ is not already in μ, to ensure N_m *unique* models are tested at each generation;

(d) until $|\mu| = N_m$, iterate to step (b).

5. Until the N_g^{th} generation is reached, iterate to step 2..
6. The strongest model on the final generation is deemed the approximation to the system, $\hat{H}'$.

1.1.1 *Models as chromosomes*

We first need a mapping from models to chromosomes; this is straightforward given the description of chromosomes as binary strings, exemplified in **??**. We assign a gene to every term in $\mathcal{T}$, so that candidate models are succinctly represented by bit strings of length N_t. We give an example of the mapping between models and chromosomes in Table 1.1. Given that every model is contained in the space of bit strings spanned by N_t bits, we can say that there are a total of 2^{N_t} available models in the model space.

1.1.2 *F_1-score*

We need a metric against which to evaluate models, and indeed the entire QMLA procedure. We can gauge the performance of QMLA's model search by the quality of candidate models produced at each generation, so we introduce a metric to act as proxy for model quality: the F_1-score, denoted f. We define F_1-score formally in this section, but in short, $f \in (0, 1)$ 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 $\hat{H}_i = \hat{H}_0$. We defined F_1-score, as well as a number of metrics in the field of classification in machine learning (ML), in **??**; here we modify those definitions to align with the nomenclature of QMLA.

We emphasise that the goal of this work is to identify the *model* which best describes quantum systems, and not to improve on parameter-learning when given access to particular models, since those already exist to a high standard [7, 8]. Therefore, in this context we can consider the role of QMLA as a classification routine[4], with the goal of classifying whether individual terms $\hat{t}$ from a set of available terms $\mathcal{T} = \{\hat{t}\}$ are helpful in describing data which is generated by $\hat{H}_0$, whose terms constitute $\mathcal{T}_0$. Candidate models $\hat{H}_i$ then have $\mathcal{T}_i$. We can assess $\hat{H}_i$ using standard metrics used regularly in the ML literature, which simply count the number of terms identified correctly and incorrectly:

Model		Chromosome					
	$\vec{T}$	$\hat{\sigma}^x_{(1,2)}$	$\hat{\sigma}^z_{(1,2)}$	$\hat{\sigma}^y_{(2,3)}$	$\hat{\sigma}^x_{(2,3)}$	$\hat{\sigma}^y_{(2,3)}$	$\hat{\sigma}^x_{(2,3)}$
γ_{p_1}	$(\hat{\sigma}^x_{(1,2)}\ \hat{\sigma}^z_{(1,2)}\ \hat{\sigma}^y_{(2,3)})$	1	0	1	0	1	0
γ_{p_2}	$(\hat{\sigma}^z_{(1,2)}\ \hat{\sigma}^y_{(2,3)}\ \hat{\sigma}^z_{(2,3)})$	0	0	1	0	1	1
γ_{c_1}	$(\hat{\sigma}^x_{(1,2)}\ \hat{\sigma}^z_{(1,2)}\ \hat{\sigma}^y_{(2,3)}\ \hat{\sigma}^z_{(2,3)})$	1	0	1	0	1	1
γ_{c_2}	$(\hat{\sigma}^z_{(1,2)}\ \hat{\sigma}^y_{(2,3)})$	0	0	1	0	1	0
γ'_{c_2}	$(\hat{\sigma}^z_{(1,2)}\ \hat{\sigma}^x_{(2,3)}\ \hat{\sigma}^y_{(2,3)})$	0	0	1	1	1	0

Table 1.1: Mapping between QMLA's models and chromosomes used by a genetic algorithm. Example shown for a three-qubit system with six possible terms, $\hat{\sigma}^w_{i,j} = \hat{\sigma}^w_i \hat{\sigma}^w_j$, $w \in \{x, y, z\}$. Model terms are mapped to binary genes: if the gene registers 1 (0) then the corresponding term is (not) present in the model. The top two chromosomes are *parents*, $\gamma_{p_1} = 101010$ (blue) and $\gamma_{p_2} = 001011$ (green): they are mixed to spawn new models. We use a one–point cross over about the midpoint: the first half of γ_{p_1} is mixed with the second half of γ_{p_2}, and vice versa, to produce two new offspring chromosomes, $\{\gamma_{c_1}, \gamma_{c_2}\}$. Mutation occurs probabilistically: each gene has a 25% chance of being mutated, e.g. a single gene (red) flipping from $0 \to 1$ to mutate γ_{c_2} to γ'_{c_2}. The next generation of the genetic algorithm will then include $\{\gamma_{c_1}, \gamma'_{c_2}\}$ (assuming γ_{c_1} 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.

- true positives (TP): number of terms in $\mathcal{T}_0$ which are in $\mathcal{T}_i$
- true negatives (TN): number of terms not in $\mathcal{T}_0$ which are also not in $\mathcal{T}_i$
- false positives (FP): number of terms in $\mathcal{T}_i$ which are not in $\mathcal{T}_0$
- false negatives (FN): number of terms in $\mathcal{T}_0$ which are not in $\mathcal{T}_i$.

These concepts – shown in Fig. 1.1 – allow us to define

- *precision*: how precisely does $\hat{H}_i$ capture $\hat{H}_0$, i.e. if a term is included in $\mathcal{T}_i$ how likely it is to actually be in $\mathcal{T}_0$, Eqn 1.1a;

4 The designation of QMLA as supervised, unsupervised, or otherwise depends upon the ES employed, **??**. The model search of this GES is unsupervised, but we use metrics from literature about classification, since we can assess performance absolutely.

- *sensitivity*: how sensitive is $\hat{H}_i$ to $\hat{H}_0$, i.e. if a term is actually in $\mathcal{T}_0$, how likely $\mathcal{T}_i$ is to include it, Eqn. 1.1b.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.1a}$$

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.1b}$$

Informally, precision prioritises that predicted terms are correct, while sensitivity prioritises that true terms are identified. In practice, it is important to balance these considerations. F_β-score (**??**) is a measure which balances these, with weighting β in favour of sensitivity. In particular, F_1-score considers precision and sensitivity as equally important:

$$F_1 = \frac{2 \times (\text{precision}) \times (\text{sensitivity})}{(\text{precision} + \text{sensitivity})} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} =: f. \tag{1.2}$$

We give an example of these quantities in Fig. 1.1, where $\text{TP} = 3, \text{TN} = 4, \text{FP} = 1, \text{FN} = 2$, giving precision $= 3/4$ and sensitivity $= 3/5$, with a final $f = 0.67$, i.e. f is the average of the indicators of model quality we care about.

We adopt F_1-score as an indication of model *quality* because we are concerned both with precision and sensitivity of the models QMLA predicts as representations of Q. We can use F_1-score to measure the success of the algorithm, by recording f for all models in all generations, allowing us to see whether or not the approximation of the system is improving on average. Of course in realistic cases we can not assume knoweldge of $\mathcal{T}_o$ and therefore cannot compute F_1-score, but it is a useful tool in the development of the GES itself, or in cases where $\hat{H}_0$ is known, such as when the target system is simualated, e.g. in the case of device calibration. Our search for an effective OF can then be guided by seeking the method which most strongly improves the average F_1-score in test-cases. We will not use F_1-score within the algorithm[5], i.e. to inform any steps taken by QMLA, but simply to assess its performance independently.

1.1.2.1 *Distinguishing F_1-score through Bayes factors*

We have so far relied on Bayes factor (BF) as the means by which to distinguish models' ability to explain data from Q. We conjecture that models of higher F_1-score are usually statistically better at predicting dynamics of Q than those of lower F_1-score, and therefore BFs will favour models of higher F_1-score. Verifying this hypothesis will allow us to incorporate statistical tools into the design of OFs; we can perform straightforward tests training models of equally spaced F_1-score, and computing BF between all pairs.

In Fig. 1.2, we show the relationships between F_1-score and BF for various conditions. Firstly, under a standard training regime with full BF comparisons between all pairs, we see that in

5 Except for meta-analysis in Section 1.1.3

Algorithm 1: ES subroutine: generate_models via genetic algorithm

Input: ν // information about models considered to date
Input: τ // truncation rate

Input: $g(\hat{H}_i)$ // objective function that can act on any model $\hat{H}_i$
Input: rank() // function to rank models relative to each other
Input: truncate() // function to truncate set of models
Input: normalise() // function to normalise models' scores
Input: roulette() // function to select models through roulette
Input: crossover() // function to crossover two parents to produce offspring
Input: mutate() // function to mutate offspring probabilistically

Output: $\mathbb{H}$ // set of models

$N_m = |\nu|$ // number of models
for $\hat{H}_i \in \nu$ **do**
  $g_i \leftarrow g(\hat{H}_i)$ // model fitness via objective function
end

$r \leftarrow \text{rank}(\{g_i\})$ // rank models by their fitness
$\mathbb{H}_t \leftarrow \text{truncate}(r, N_m \times \tau)$ // truncate models by rank: only keep $N_m \times \tau$
$s \leftarrow \text{normalise}(\{g_i\})\ \forall \hat{H}_i \in \mathbb{H}_t$ // normalise remaining models' fitnesses
$\mathbb{H} = \{\}$ // new batch of chromosomes/models
while $|\mathbb{H}| < N_m$ **do**
  $p_1, p_2 = \text{roulette}(s)$ // use s to select two parents via roulette selection
  $c_1, c_2 = \text{crossover}(p_1, p_2)$ // produce offspring models
  $c_1, c_2 = \text{mutate}(c_1, c_2)$ // probabilistically mutate
  $\mathbb{H} \leftarrow \mathbb{H} \cup \{c_1, c_2\}$ // add new models to batch
end

return $\mathbb{H}$

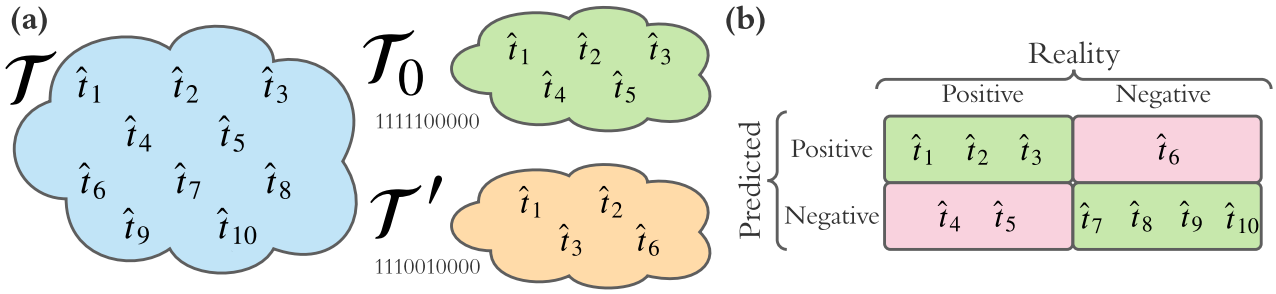

Figure 1.1: Concepts used for classification. **a**, the set of available terms $\mathcal{T}$ containing individual terms $\hat{t}_1$ to $\hat{t}_{10}$. The true model $\hat{H}_0$ is constructed from the set $\mathcal{T}_0$. Suppose a candidate $\hat{H}'$ has the set $\mathcal{T}'$. **b**, the confusion matrix for $\hat{H}'$. Correctly classified terms are true positives and true negatives (green), and incorrectly classified terms are false positives and true negatives (red).

most cases, the model with higher F_1-score is favoured by BF. Some comparisons favour near the diagonals of Fig. 1.2 favour the model with lower F_1-score, although in these cases we argue that the difference in model quality is not overwhelming, since $\left|f_i - f_j\right| \lessapprox 0.1$. In Fig. 1.2b, we run a complete model training subroutine, but compute the BF based on fewer experiments and particles (retaining a fraction $N_p' = 0.2N_p,\ N_e' = 0.2N_e$ for comparisons). This verifies an earlier claim from **??**: although the strength of evidence is weaker given reduced BF resources, the direction of the evidence is usually the same, i.e. the insight is indicative of the true physics, so we can save considerable compute time by trusting these restricted BF calculations. On the other extreme, we see in Fig. 1.2**(c)**, where models are trained with – and BFs based upon – even greater resources than Fig. 1.2**(a)**, we see a similar effect: adding resources strengthens the evidence, but does not fundamentally change the outlook. Finally, in addition to reducing the resources used per BF calculation, we reduce the number of comparisons computed in Fig. 1.2**(d)**, as permitted when rating models according to the OF to be described in Section 1.2.7, or similar measures which can yield fitnesses from reduced data. Essentially we can see that the insight is largely the same from the most and least expensive training/comparison strategies, and by leveraging the available evidence (Fig. 1.2**(d)**), rather than brute-force computing as much evidence as possible (Fig. 1.2**(c)**), we can achieve similar results. Note that the time saving reported between full and partial connectivity between models scales with N_m: here, with $N_m = 10$, the former computes 45 BFs, while the latter computes 17; for $N_m = 60$, as used in full instances/runs presented in this chapter, these rise to 600 and 1770 BFs computations respectively, so the benefit of the latter scheme is amplified.

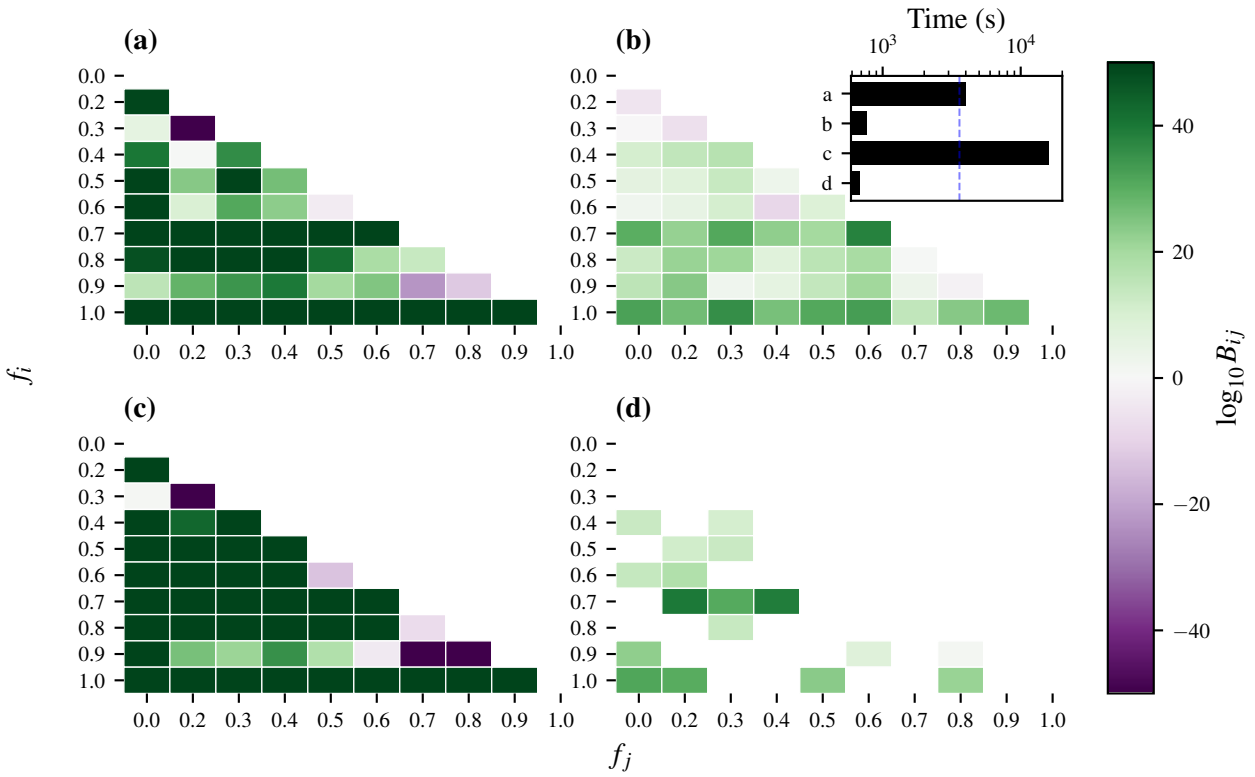

Figure 1.2: Pairwise Bayes factor by F_1-score. Each tile shows $\log_{10} B_{ij}$ against the F_1-score of the two candidates $\hat{H}_i$ (f_i on the y-axis) and $\hat{H}_j$ (f_j on the x-axis). $\log_{10} B_{ij} > 0$ shown in green ($\log_{10} B_{ij} < 0$ shown in purple) indicates statistical evidence that $\hat{H}_i$ ($\hat{H}_j$) is the better model with respect to the observed data. Visualisation is curtailed to $\log_{10} B_{ij} = \pm 50$. **a**, Models are trained with $N_e = 500, N_p = 2500$, and all available data is used in the calculation of BFs. **b**, $N_e = 500, N_p = 2500$ using only a fraction (0.2) of experiments/particles for BF calculations. **c**, $N_e = 1000, N_p = 5000$, using all available data in the calculation of BFs. **d**, $N_e = 500, N_p = 2500$, comparing only a subset of pairs of models through BFs, and using only a fraction (0.2) of experiments/particles for those calculations. This pairwise comparison strategy is used for the OF in Section 1.2.7. **Inset**, timings for each approach in seconds, with $t = 1$hr marked vertically in blue. Implementation details are listed in **??**

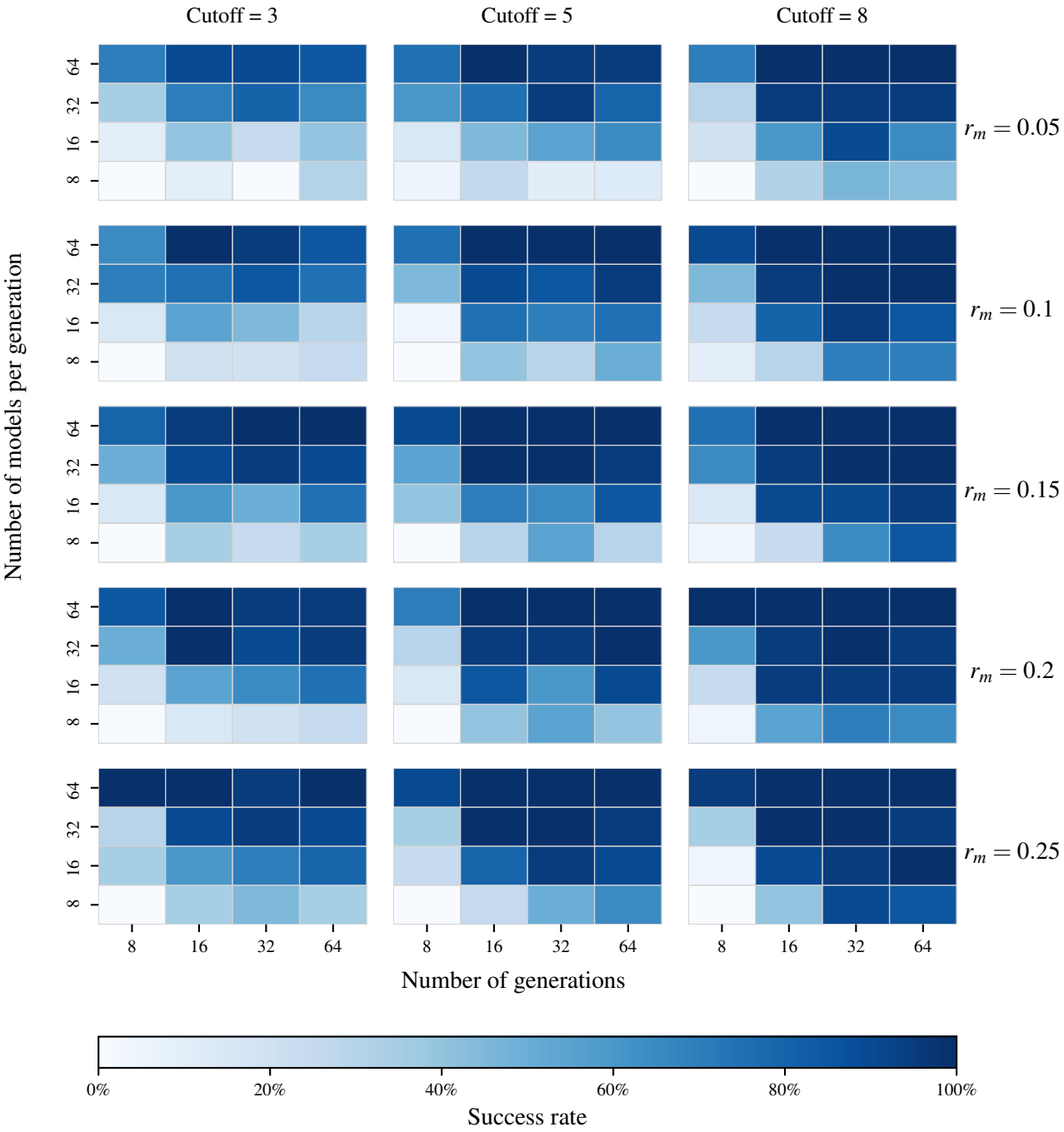

Figure 1.3: Genetic algorithm hyperparameter sweep. Each tile shows the success rate of a run with hyperparameter settings $\{N_m, N_g, r_m, \text{cutoff}\}$, where the success rate is the percentage of 20 instances which found a random $\hat{H}_0$ using F_1-score as objective function. Each subplot shows the success rates for varying numbers of generations, $N_G \in \{8, 16, 32, 64\}$, and numbers of models per generation, $N_m \in \{8, 16, 32, 64\}$. A subplot is generated for ranges of the mutation rate, r_m and the number of generations for which the elite model is unchanged after which the GA is cut off. Implementation details are listed in **??**

1.1.3 *Hyperparameter search*

Firstly we will validate our reasoning that F_1-score is a sensible figure of merit, by directly invoking it as the objective function. That is, we first implement a GA, using the mapping between models and chromosomes outlined above, where we fix the numbers of sites $d = 4$, and assume full connectivity between the sites, with $x-$, $y-$ and $z-$ couplings available, such that there are $N_t = 3 \times \binom{4}{2} = 18$ terms in $\mathcal{T}$, so that the total population is of size 2^{18} chromosomes. We can then sweep over the GA hyperparameters to find a suitable configuration: in Fig. 1.3 we show how the choice of parameters affect the success rate of preciesly identifying the target chromosome, which is chosen at random for each instance, and we run 20 instances of each configuration. The studied hyperparameters[6] are

i. number of generations, N_m;

ii. number of models per generation, N_g;

iii. mutation rate, r_m;

iv. number of generations for which a candidate must reign as the strongest observed, before the search terminates, the *cutoff*.

Naturally, we expect that running for more generations with more models per generation will result in a more effective search in the model space, having examined $N_g N_m$ models. We must also consider, however, that – in realistic cases of QMLA – the total computation time scales dramatically with these parameters, since training and comparing models are expensive subroutines. Our goal is therefore to identify the set of hyperparameters which best searches the model space while demanding the lowest $\{N_g, N_m\}$. We see that, unsurprisingly, the GA performs poorly when run with few resources, but broadly the performances are similiar provided it is run with sufficient resources. We can bound the parameters $r_m \geq 0.1, \text{cutoff} \geq 5, N_m \geq 16, N_g \geq 16$ to ensure a reasonable search through the model space, without having to consider a prohibitive number of models. We must bear in mind, however, that this parameter sweep refers only to the trivial case where the F_1-score is used as the OF, so we do not expect such high success rates in realistic cases.

1.2 OBJECTIVE FUNCTIONS

We have alluded to the central problem in building a GA into QMLA: how to evaluate trained candidate models in the absence of a natural objective function (OF). In Sections 1.2.1 to 1.2.7 we will propose and analyse a number of potential OFs, some of which will underlie later studies in this thesis. We conclude this study by comparing the proposed OFs and selecting one for consideration in the remainder of this chapter; readers interested in the final application may prefer to skip to Section 1.2.8.

6 These and further hyperparameters can be swept using code within the QMLA codebase, in the directory scripts/-genetic_alg_param_sweep.

We will show how each OF computes a fitness, g_i, for candidate models, $\hat{H}_i$. For examples of each, we group together some demonstrative values in Table 1.2. For each $\hat{H}_i$, we may refer to

- $\mathcal{L}_i$, total log total likelihood (TLTL), introduced in **??**;
- k_i, the model's cardinality, i.e. number of terms in its parameterisation;
- $\mathcal{E}_i$, the bespoke set of experiments composed by the experiment design heuristic (**??**) solely for trainng $\hat{H}_i$;
- $n = |\mathcal{E}_i|$, the number of samples (datapoints) used in trainng $\hat{H}_i$.

In Table 1.2, we consider six randomly generated exemplary models – of varying quality with respect to the target, $\hat{H}_0$, listed in Eq. (1.3) – to demonstrate each OF's outcomes.

$$\begin{aligned}
\hat{H}_0 &= \hat{\sigma}^z_{(1,2)}\hat{\sigma}^z_{(1,3)}\hat{\sigma}^z_{(2,3)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(3,5)};\\
\hat{H}_a &= \hat{\sigma}^z_{(1,5)}\hat{\sigma}^z_{(3,4)}\hat{\sigma}^z_{(4,5)};\\
\hat{H}_b &= \hat{\sigma}^z_{(1,4)}\hat{\sigma}^z_{(1,5)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(3,4)};\\
\hat{H}_c &= \hat{\sigma}^z_{(1,2)}\hat{\sigma}^z_{(1,5)}\hat{\sigma}^z_{(2,4)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(4,5)};\\
\hat{H}_d &= \hat{\sigma}^z_{(1,3)}\hat{\sigma}^z_{(1,4)}\hat{\sigma}^z_{(1,5)}\hat{\sigma}^z_{(2,4)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(3,4)}\hat{\sigma}^z_{(3,5)};\\
\hat{H}_e &= \hat{\sigma}^z_{(1,2)}\hat{\sigma}^z_{(1,3)}\hat{\sigma}^z_{(1,5)}\hat{\sigma}^z_{(2,3)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(4,5)};\\
\hat{H}_f &= \hat{\sigma}^z_{(1,2)}\hat{\sigma}^z_{(1,3)}\hat{\sigma}^z_{(2,3)}\hat{\sigma}^z_{(2,4)}\hat{\sigma}^z_{(2,5)}\hat{\sigma}^z_{(3,4)}\hat{\sigma}^z_{(3,5)}.
\end{aligned} \tag{1.3}$$

1.2.1 *Inverse log-likelihood*

$\mathcal{L}_i$, defined in **??**, can be thought of as a measure of the ability of a given model to reproduce a given dataset $\mathcal{D}$ from experiments $\mathcal{E}$. This can be immediately interpreted as an OF, provided each candidate model computes a meaningful total log total likelihood (TLTL), requiring that they are all based on the same set of experiments, $\mathcal{E}_v$, which are designed explicitly for the purpose of model evaluation.

TLTL are negative and the strongest model has lowest $|\mathcal{L}_i|$ (or highest $\mathcal{L}_i$ overall), so the corresponding OF for candidate $\hat{H}_i$ is

$$g_i^L = \frac{-1}{\mathcal{L}_i}. \tag{1.4}$$

In our tests, Eqn. 1.4 is found to be too generous to poor models, assigning them non-negligible probability. Its primary flaw, however, is its reliance on $\mathcal{E}_v$: in order that the TLTL is significant, it must be based on meaningful experiments, the design of which can not be gauranteed in advance, or at least risks introducing strong bias towards some models.

Method		$\hat{H}_a$	$\hat{H}_b$	$\hat{H}_c$	$\hat{H}_d$	$\hat{H}_e$	$\hat{H}_f$
	F_1	0.0	0.2	0.4	0.5	0.7	0.8
	k	3	4	5	7	6	7
	$\overline{l_e}$	0.86 ± 0.29	0.84 ± 0.29	0.77 ± 0.27	0.78 ± 0.29	0.79 ± 0.26	0.79 ± 0.26
	$\mathcal{L}_i$	-143	-152	-131	-150	-125	-124
Inverse log-likelihood	g_i^L	0.00698	0.00659	0.00766	0.00669	0.00803	0.00804
	%	23	0	25	0	26	26
Akaike Info Criterion	AIC	293	311	271	313	261	263
	AICC	293	312	272	314	262	264
	w_i^A	1.81e-07	1.4e-11	0.00724	4.15e-12	1	0.334
	g_i^A	1.17e-05	1.03e-05	1.35e-05	1.01e-05	1.46e-05	1.43e-05
	%	22	0	25	0	27	26
Bayesian Info Criterion	BIC	301	322	284	331	277	281
	w_i^B	5.49e-66	1.26e-70	1.97e-62	1.11e-72	8.43e-61	8.95e-62
	g_i^B	1.11e-05	9.65e-06	1.24e-05	9.11e-06	1.31e-05	1.27e-05
	%	23	0	25	0	27	26
Bayes factor points	g_i^p	0	2	3	2	3	5
	%	0	13	20	13	20	33
Ranking	Ranking	6	4	3	5	2	1
	g_i^R	0	0.1	0.2	0	0.3	0.4
	%	0	10	20	0	30	40
Elo rating	Rating	909	944	1042	1007	1011	1084
	g_i^E	0	35	133	98	102	175
	%	0	0	26	19	20	34
Residuals	mean$\{\tilde{r_p^e}\}$	0.132	0.146	0.114	0.138	0.0858	0.0715
	g_i^r	0.753	0.729	0.785	0.743	0.836	0.862
	%	23	0	24	0	26	27

Table 1.2: Examples of how each objective function (OF), g as described in Section 1.2.1 to Section 1.2.7, assign selection probability (denoted %) to the same set of candidate models, $\{\hat{H}_i\}$ listed in Eq. (1.3), when attempting to learn data from $\hat{H}_0$. Intermediate quantities, e.g. w_i^A, g_i^p are described in the section of the main text describing the corresponding OF. For each model we first summarise its F_1-score (Eq. (1.2)), number of terms k, median likelihood $\overline{l_e}$ (**??**), and TLTL $\mathcal{L}_i$ (**??**), We use $n = 250$ samples, i.e. $\mathcal{L}_i$ is a sum of n likelihoods . The set of models is truncated so that only the strongest four are assigned selection probability.

1.2.2 *Akaike information criterion*

A common metric in the general field of model selection is Akaike information criterion (AIC) [9]. Incorporating TLTL, AIC objectively quantifies how well a given model accounts for data from the target system, and explicitly punishes models which use extraneous parameters by incurring a penalty on k_i. AIC is given by

$$AIC_i = 2k_i - 2\mathcal{L}_i. \tag{1.5}$$

In practice we use a slightly modified form of Eqn. 1.5 which corrects for the number of samples $n = |\mathcal{E}_i|$, called the Akaike information criterion corrected (AICC),

$$AICC_i = AIC_i + 2k_i \frac{k_i + 1}{n - k_i - 1}. \tag{1.6}$$

Model selection from a set of candidates occurs simply by selecting the model with lowest $AICC$. Following [9], by using Eqn. 1.6 as a measure of *relative likelihood* we retrieve selection probability via the *Akaike weights*,

$$w_i^A = \exp\left(\frac{AICC_{\min} - AICC_i}{2}\right), \tag{1.7}$$

where $AICC_{\min} = \min_i\{AICC_i\}$.

Akaike weights impose strong penalties on models which do not explain the data well, but also punish models with more parameters, i.e. potentially overfitting models, effectively searching for the strongest and simplest model simultaneously. The level of punishment for poorly performing models is likely too drastic: very few models will be in a range sufficiently close to $AICC_{\min}$ to receive a meaningful Akaike weight, suppressing diversity in the model population. Indeed, we can see from Table 1.2 that this results in most models being assigned negligible weight, which is not useful for parent selection. Instead we compute a straightforward quantity related to AIC,

$$g_i^A = \left(\frac{1}{AICC_i}\right)^2, \tag{1.8}$$

where we square the inverse AICC to amplify the difference in quality between models, such that stronger models are rewarded.

1.2.3 *Bayesian information criterion*

Related to the concept of AIC (Eqn. 1.5), is that of Bayesian information criterion (BIC),

$$BIC_i = k_i \ \ln(n_i) - 2\mathcal{L}_i, \tag{1.9}$$

where k_i, n_i and $\mathcal{L}_i$ are as defined on Page 15. Analagously to Akaike weights, *Bayes weights* as proposed in §7.7 of [10], are given by

$$w_i^B = \exp\left(-\frac{BIC_i}{2}\right). \tag{1.10}$$

BIC is harsher than AIC in its punishment of models' cardinality k_i, demanding substantial statistical justification for the inclusion of more parameters. Again, this may be overly cumbersome for our use case: with such a relatively small number of parameters, the punishment is disproportionate. As with Akaike weights, rather than using Bayes weights directly, we opt for an OF related to them,

$$g_i^B = \left(\frac{1}{BIC_i}\right)^2. \tag{1.11}$$

1.2.4 *Bayes factor points*

A cornerstone of model selection within QMLA is the calculation of BFs (see **??**). We can compute the pairwise BF between two candidate models, B_{ij}, according to Eqn. **??**. B_{ij} can be based on some evaluation dataset, $\mathcal{E}_v$, but can also be calculated from $\mathcal{E}_i \cup \mathcal{E}_j$: this is a strong advantage since the resulting insight (Eqn. **??**) is based on experiments which were bespoke to both $\hat{H}_i, \hat{H}_j$. As such we can be confident that this insight accurately points us to the stronger of two candidate models.

We can utilise this facility by computing the BF between all pairs of models in a set of N_m candidates $\{\hat{H}_i\}$, i.e. compute $\binom{N_m}{2}$ BFs. Note that this is computationally expensive: in order to train $\hat{H}_i$ on $\mathcal{E}_j$ requires a further $|\mathcal{E}_j|$ experiments, each requiring N_P particles[7], where each particle corresponds to a unitary evolution and therefore the caluclation of a matrix exponential. The size of the model space is then quite a heavy disadvantage: examining N_g generations requires $N_g \times \binom{N_m}{2}$ BF calculations for complete assessment.

In the case where all pairwise BF are performed, we can assign a point to $\hat{H}_i$ for every comparison in which it is deemed superior, according to **??**.

$$g_i^p = \sum_{j \in \mu} b_{ij}, \quad b_{ij} = \begin{cases} 1, & B_{ij} > 1 \\ 0, & \text{otherwise.} \end{cases} \tag{1.12}$$

This is a straightforward mechanism, but is overly blunt because it does not account for the *strength* of the evidence in favour of each model. For example, a dominant model will receive only a slightly higher selection probability than the second strongest, even if the difference between them was $B_{ij} = 10^{100}$. Further, the unfavourable scaling make this an expensive method.

7 Caveat the reduction in overhead outlined in **??**.

1.2.5 *Ranking*

Related to the BF points of the previous section, we can rank models in a generation based on their number of BF points. BF points are assigned as in Eqn. 1.12, but instead of corresponding directly to fitness, we assign models a rank R, i.e. the model with highest g_i^p gets $R = 1$, and the model with n^{th} highest g_i^p gets $R = n$. Note here we truncate μ, meaning we remove the worse-performing models and retain only N'_m models, before calculating R, because computing R using all N_m models results in less distinct selection probabilities.

$$g_i^R = \frac{N'_m - R_i + 1}{\sum_{j=1}^{N'_m} j}, \tag{1.13}$$

where R_i is the ranking of $\hat{H}_i$ and N'_m is the number of models retained after truncation. Eq. (1.13) has a similary effect to Eq. (1.12) but awards higher selection probability to the strongest models. However, it too overlooks the nuanced perspective available through the total statistical evidence gathered by the series of BFs.

1.2.6 *Residuals*

Recall at each experiment, N_P particles are compared against a single experimental datum, d. By definition, d is the binary outcome of the measurement on Q under experimental conditions e. That is, d encodes the answer to the question: after time t under Hamiltonian evolution, did Q project onto the basis we have labelled $|d\rangle$ (usually the same as the input probe state $|\psi\rangle$)?

In practice we often have access to the complete likelihood, i.e. rather than a binary value, we have a number representing the probability that Q will project on to $d = 0$ for a given experiment e, $\Pr_Q(0|e)$. The likelihood – in this case equivalent to the expectation value[8] – for Q is usually given by $\left|\langle\psi| e^{-i\hat{H}_0 t} |\psi\rangle\right|^2$. Likewise, we can simulate this quantity for each particle, $\Pr_p(0|e)$. This allows us to calculate the *residual* between the system and individual particles' likelihoods, r_p^e; we can hence compute the mean residual across all particles in a single experiment, r^e:

$$\begin{aligned} r_p^e &= \left|Pr_Q(0|e) - Pr_p(0|e)\right| \\ r^e &= \underset{p}{\text{mean}}\{r_p^e\} \end{aligned} \tag{1.14}$$

8 For consistency with QInfer [11] – on which QMLA's code base builds – we call the expectation value for the system $\Pr_Q(0)$; the same quantity can be computed for each particle, called $\Pr_p(0)$.

Residuals capture how closely the particle distribution reproduced the dynamics from Q: $r_p^e = 0$ indicates perfect prediction, while $r_p^e = 1$ is completely incorrect. We can therefore maximise the quantity $1 - r$ to find the best model, using the OF

$$g_i^r = |1 - \underset{e \in \mathcal{E}}{\text{mean}}\{r^e\}|^2. \tag{1.15}$$

This OF can be thought of in frequentist terms as similar to the residual sum of squares, although instead of summing the residual squares, we take the average to ensure $0 \leq r \leq 1$. g_i^r encapsulates how well the candidate model reproduces a paticular set of dynamics from the target system, as a proxy for how well that candidate describes the system. This is not always a safe figure of merit: in most cases, we do not expect parameter learning to perfectly optimise $\vec{\alpha}_i$. Reproduced dynamics alone can not capture the prospect that $\hat{H}_i = \hat{H}_0$, but rather inform statistical measures such as BF, that allow us to make qualified statements about the system.

This OF provides a useful test for QMLA's GA: by simulating the case where parameters *are* learned perfectly, such that we know that g_i^r truly represents the ability of $\hat{H}_i$ to mimic $\hat{H}_0$, then this OF gaurantees to promote the strongest models, especially given that $\hat{H}_i = \hat{H}_0 \implies r_p^e = 0 \; \forall \; \{e, p\}$. In realistic cases, however, the non-zero residuals – even for strong $\hat{H}_i$ – may arise from imperfectly learned parameters, rendering the usefulness of this OF uncertain. Finally, it does not account for the cardinality, k_i, of the candidate models, which all ML protocols aim to avoid in general; this could result in favouring severely overfitting models in order to gain marginal improvement in residuals.

1.2.7 *Bayes factor enhanced Elo ratings*

A popular tool for rating individual competitors in sports and games is the *Elo rating* system, e.g. used to rate chess players and soccer teams [12, 13], also finding application in the study of animal hierarchies [14]. Elo ratings allow for evaluating the relative quality of individuals based on incomplete pairwise competitions, e.g. despite two football teams having never played against each other before, it is possible to quantify the difference in quality between those teams, and therefore to predict a result in advance [15]. There is a direct a parallel between these types of competitions and QMLA: we similarly have a pool of individual competitors (models), which we can place in direct competition, and quantify the comparitive outcome through BF, in order to determine the preferred candidate.

Elo ratings are transitive: given some interconnectivity in a generation, we need not compare *every* pair of models in order to make meaningful claims about which are strongest; it is sufficient to perform a subset of comparisons, ensuring each individual undergoes robust competition. We can take advantage of this transitivity to reduce the combinatorial overhead usually associated with computing bespoke BFs between all models (i.e. using their own training data $\mathcal{E}_i$ instead of a generic $\mathcal{E}_v$). In practice, we map N_m models within a generation to vertices on a regular graph of degree $N_m/3$, i.e. each model is connected to $N_m/3$ other models within μ. Models

which share an edge then undergo BF comparison. For example, with $N_m = 60$ this leads to 600 BF calculations, compared with 1770 calculations in the fully connected graph. While every pair of models $(\hat{H}_i, \hat{H}_k)$ are not directly connected, there is always a chain of length $l \leq l_{max}$ edges between them. For $N_m = 60$, we find $l_{max} = 2$, e.g. for $\hat{H}_i, \hat{H}_k$ disconnected, there are comparisons $(\hat{H}_i, \hat{H}_j), (\hat{H}_j \hat{H}_k)\}$.

The Elo rating scheme is a nonlinear points transfer system, as follows: upon creation, $\hat{H}_i$ is assigned a rating R_i; every comparison with a competitor $\hat{H}_j$ results in B_{ij}; R_i is updated according to the known strength of its competitor, R_j, as well as the result B_{ij}. The Elo update ensures that winning models are rewarded for defeating another model, but that the extent of that reward reflects the quality of its opponent. As such, this is a fairer mechanism than BF points, which award a point for every victory irrespective of the opposition: if $\hat{H}_j$ is already known to be a strong or poor model, then ΔR_i proportionally changes the credence we assign to $\hat{H}_i$. It achieves this by first computing the *expected* result of a given comparison with respect to each model, with an exponential response to the difference in their current ratings,

$$E_i = \frac{1}{1 + 10^{\frac{R_j - R_i}{400}}}; \tag{1.16a}$$

$$E_i + E_j = 1, \tag{1.16b}$$

Then, we find the binary *score* from the perspective of each model,

$$\begin{cases} B_{ij} > 1 & \Rightarrow \; S_i = 1; \; S_j = 0 \\ B_{ij} < 1 & \Rightarrow \; S_i = 0; \; S_j = 1 \end{cases} \tag{1.17}$$

which is used to determine the change to each model's rating,

$$\Delta R_i = \eta \times (S_i - E_i) \,. \tag{1.18}$$

An important detail is the choice of η, i.e. the *weight* of the change to the models' ratings. In standard Elo schemes this is a fixed constant, but here – taking inspiration from football ratings where η is the number of goals by which one team beat the other – we weight the change by the strength of our belief in the outcome: $\eta \propto |B_{ij}|$. That is, similarly to the interpretation of Eqn. **??**, we use the evidence in favour of the winning model to transfer points from the loser to the winner, albeit we temper the effect by instead using $\eta = \log_{10}(B_{ij})$, since BF can give very large numbers. In total, then, following the comparison between models $\hat{H}_i, \hat{H}_j$, we can perform the Elo rating update

$$R_i' = R_i + \log_{10}(B_{ij}) \left(S_i - \frac{1}{1 + 10^{\frac{R_j - R_i}{400}}} \right). \tag{1.19}$$

This procedure is easiest to understand by following the example in Table 1.3.

9 Note to achieve $B_{ij} = 10^{100} = e^{\mathcal{L}_i - \mathcal{L}_j} \implies \mathcal{L}_i - \mathcal{L}_j = \ln(10^{100}) \approx 7$.

	Model	R_i	E_i	S_i	B_{ij}	$\log_{10}(B_{ij})$	ΔR_i	R'_i
$\hat{H}_a > \hat{H}_b$	$\hat{H}_a$	1000	0.76	1	1e+100	100	0.24	1024.0
	$\hat{H}_b$	800	0.24	0	1e-100	100	-0.24	776.0
$\hat{H}_b > \hat{H}_a$	$\hat{H}_a$	1000	0.76	0	1e-100	100	-0.76	924.0
	$\hat{H}_b$	800	0.24	1	1e+100	100	0.76	876.0

Table 1.3: Example of Elo rating updates. We have two models, where $\hat{H}_a$ is initially quantified as a stronger candidate than $\hat{H}_b$, i.e. has a higher starting Elo rating, R_i. We demonstrate the effect when there is strong evidence[9] in favour of either model through BF comparison, $B_{ij} \sim 10^{100}$. In the first case, $\hat{H}_a$ defeats $\hat{H}_b$, as firmly expected according to their initial ratings, so the corresponding reward (cost) for $\hat{H}_a$ ($\hat{H}_b$) is relatively small. In the second case, contrary to prediction $\hat{H}_b$ outperforms $\hat{H}_a$, so $\hat{H}_b$ receives a large share of Elo points from $\hat{H}_a$.

Finally, it remains to select the starting rating R_i^0 to assign models upon creation. Although this choice is arbitrary, it can have a strong effect on the progression of the algorithm. Here we impose details specific to the QMLA GES: at each generation we admit the top two models automatically for consideration in the next generation, such that strongest models can stay alive in the population and ultimately win. These are called *elite* models, $\hat{H}_e^1, \hat{H}_e^2$. This poses the strong possibility for a form of generational wealth: if elite models have already existed for several generations, their Elo ratings will be higher than all alternatives by defintion. Instead, we would prefer that newly spawned models can overtake the Elo rating of elite models. To resolve this, at each generation, all models – including $\hat{H}_e^1, \hat{H}_e^2$ – are assigned the same initial rating, $R_i^0 = 1000$.

In order to derive a meaningful selection probability for each candidate, we must first ground the raw Elo rating at each generation μ: we subtract the lowest rating among the entertained models, R^μ_{min}. This serves to ensure the range of remaining R_i represent only by the difference between models as assessed within μ: a very strong model might have much higher R_i than its contemporaries, but that difference was earned exclusively by comparison within μ, so it is deserving of its higher fitness and therefore greater selection probability. We perform this step before truncation[10], so that the models remaining post-truncation all have non-zero fitness. Finally, then, we name this OF the *Bayes factor enhanced Elo ratings (BFEER)*: the fitness of each model $\hat{H}_i^\mu \in \mu$ is attained directly from its rating R_i after undergoing Elo updates based on BFs in the current generation, minus the minimum rating of any model in the same generation R^μ_{min},

$$g_i^E = R_i^\mu - R^\mu_{\text{min}}. \tag{1.20}$$

10 We truncate the N_m models on μ by the truncation rate τ, i.e. only τN_m models are considered as potential parents in the GA. In this chapter we use $\tau = 1/3$.

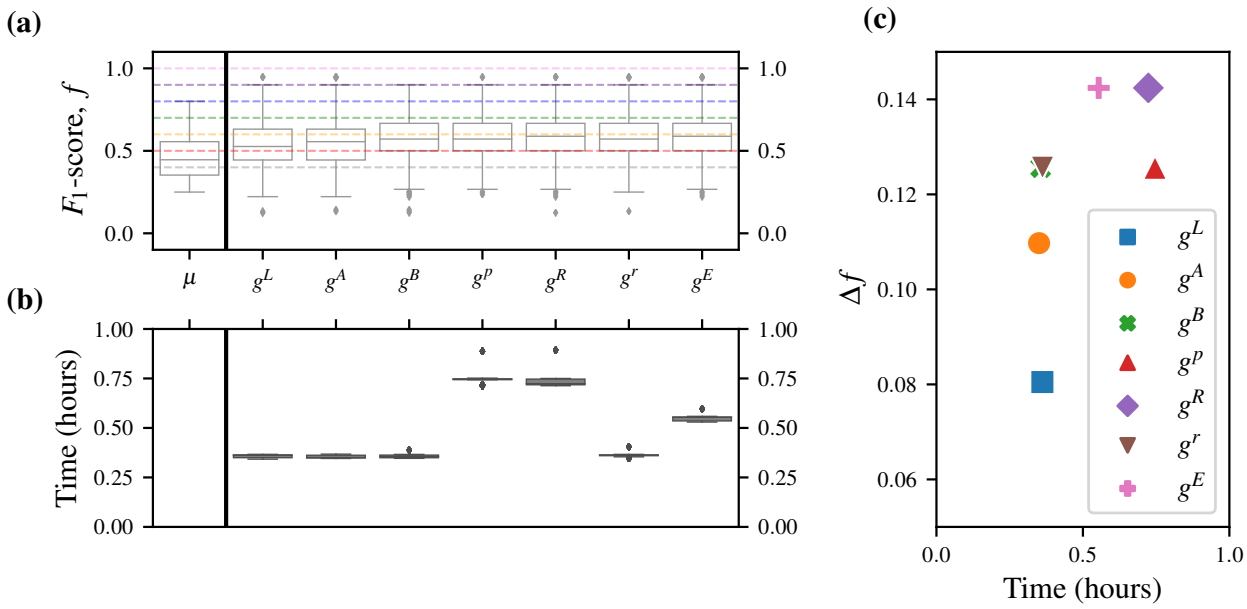

Figure 1.4: Comparison between proposed objective functions (OFs). Each OF trains the same initial generation of $N_m = 28$ models with resources $N_E = 500, N_P = 2000$, and then design a new set of N_m models through a roulette strategy, such that the only difference between OF's output is how they assign selection probability. We run each OF 25 times for the same target system, a 4–qubit Heisenberg–XYZ model. **a**, box-plot of spawned models' F_1-score, f, where the median and inter–quartile ranges are indicated by the boxes, as well as those of the initial generation μ centered on $f_\mu = 0.45$. We mark $f = \{0.4, 0.5, ..., 1.0\}$ for ease of interpretation. **b**, box-plots of the time taken to compute the single generation in each case. **c**, the difference between the median f among the newly proposed models from f_μ, Δf, plotted against the time to achieve the result.

The advantage of this OF is that it gives a meaningful value on the absolute quality of every model, allowing us to determine the strongest, and importantly to find the relative strength between models. Further, it exploits bespoke BFs, i.e. based on the considered models' individually designed $\mathcal{E}_i$, removing the impetus to design $\mathcal{E}_v$ which can evaluate models definitively. One disadvantage is that it does not explicitly punish models based on their cardinality, however this feature is partially embedded by adopting BF for the comparisons, which are known to protect against overfitting [16].

1.2.8 *Objective function selection*

Having proposed a series of possible objective functions, we are now in a position to analyse their appropriateness in the context of QMLA. Recall from Section 1.1.2 that we use F_1-score as

the figure of merit against which individual models are measured; we can compare OFs on the basis of the F_1-score of models they spawn.

First we can remark on the examples listed in Table 1.2. The OFs which rely on the TLTL, i.e. g^L, g^A, g^B, g^r, are effectively tricked by the log-likelihood, which appears reasonably convincing for poor models, e.g. $\hat{H}_a, \hat{H}_c$. This underlines the risk in buidling $\mathcal{E}_v$, which can be biased towards weak models, for example resulting in high selection probability for $\hat{H}_a$ which has $f = 0$, while $\hat{H}_d$, with $f = 0.4$ is discarded. On the other hand, OFs grounded by the BF (g^p, g^R, g^E) invariably promote models of higher F_1-score, justifying the role of statistical evidence used for those calculations. Overall, however, the insights from this complete example are insufficient to make general claims about the performance of each OF, so here we examine their outputs systematically.

Returning to the task of determining our favoured OF, we choose some random target $\hat{H}_0$, and run a single generation of the GES with each OF, allowing us to assess their performance based on the quality of models the GA produces under their respective guidance. We train the same batch of $N_m = 28$ random models in each case, and allow each OF to compute the selection probabilities for those models, and therefore direct the design of the hypothetical next generation of models. We plot the distribution of F_1-score that each OF produces in Fig. 1.4, also accounting for the time taken in each case, i.e. we report the time to train and evaluate the single generation on a 16–core node.

We can see that a strong balance of outcome with resource considerations are achieved by the BFEER strategy, Section 1.2.7, so we will use it for the case study presented in this chapter. We strongly emphasise, however, that the performance of each objective function can vary under alternative conditions, and therefore similar analysis may be warranted for future applications. For instance, if $t_{\max}$ is known to be small, in smaller model spaces, using g^r results in higher success rates. We retain BFEER, however, for generality and novelty, but it is important to recognise that the results listed do not reflect an upper limit of QMLA's performance, but rather reflect the constraints of the system under study; each Q will bring its own unique considerations which can result in significantly stronger or weaker performance under each OF. In particular, we will later use the residual OF, Section 1.2.6, to study a larger model space under assumptions of perfect parameter learning, **??**.

1.3 APPLICATION

Having introduced all the necessary concepts of GAs, mapped them to the QMLA framework and chosen a suitable OF, we can finally use the GES for model search. In summary of this chapter so far, we use the following settings.

- Models are mapped to a unique bit string (chromosome), where each bit represents whether a given model term (gene) is present; chromosomes are of length N_t genes.
- A maximum of N_g generations are run, each with N_m unique models.

- Candidate models are trained using QHL, specifically by using interactive quantum likelihood estimation (IQLE)[11] for parameter estimation.
- Models' fitness are determined by their BFEER, after having been trained by QHL and compared against some set of competing candidate models.
- For generating models on $\mu + 1$, the models on μ are first truncated with truncation rate τ; the remaining τN_m models are assigned selection probability based on their fitness.
- Pairs of models are selected to become parents sequentially using roulette selection. Highly favoured models can parent many offspring models.
- Selected parent models are crossed over via a one-point cross-over, at crossover location $\kappa \in \left(\frac{N_t}{4}, \frac{3N_t}{4}\right)$, and probabilistically mutated with rate $r_m = 0.25$.
- The top two elite models from μ are included on the subsequent generation $\mu + 1$.
- If, after 5 generations, the highest-fitness (elite) model is unchanged, i.e. $\hat{H}_e^{\mu} = \hat{H}_e^{\mu-5}$, we terminate the search and declare that model as the champion, $\hat{H}' = \hat{H}_e^{\mu}$.
- Otherwise, after N_g generations, the highest-fitness model on the final generation is declared the global champion model, $\hat{H}' = \hat{H}_C^{N_g}$.

We will use a four-qubit model space under the fully parameterised Heisenberg formalism, **??**, such that any pair of sites $\langle k, l \rangle$ can be coupled by any of the terms $\{\hat{\sigma}^x_{\langle k,l\rangle}, \hat{\sigma}^y_{\langle k,l\rangle}, \hat{\sigma}^z_{\langle k,l\rangle}\}$, so in total there are $N_t = |\mathcal{T}| = 3 \times \binom{4}{2} = 18$ terms, giving a model space of $2^{18} \approx 250,000$ viable models/chromosomes. For practical reasons[12], we set $N_m = 60$ and $N_g = 16$, although in most cases the elitism clause is triggered so the search terminates long before N_g is reached. The true parameters $\vec{\alpha}_0$ are assigned randomly in the range $(0.25, 0.75)$; within QHL the prior is set as a multivariate normal distribution 0.5 ± 0.125. We choose $\hat{H}_0$ at random to contain half the available terms[13],

$$\hat{H}_0 = \hat{\sigma}^{yz}_{(1,2)} \hat{\sigma}^{z}_{(1,3)} \hat{\sigma}^{y}_{(1,4)} \hat{\sigma}^{xy}_{(2,3)} \hat{\sigma}^{x}_{(2,4)} \hat{\sigma}^{xz}_{(3,4)}. \tag{1.21}$$

1.3.1 *Analysis*

We will analyse the GES from four perspectives: a single model, a single generation, a single QMLA instance, and the overall performance across many instances, i.e. a run.

Recall that BFEER are mediated through random graphs: given N_m models on μ, a given model $\hat{H}_i$ undergoes some $N_i^{BF} < N_m$ BF comparisons. In Fig. 1.5 we show the BF results

11 IQLE assumes complete access to the target system, see **??**. This restricts the present analysis to simulateable, rather than physical, use cases, e.g. device calibration.

12 This is to ensure, with 15 available worker nodes, and accounting for some slowly-learning models, that all N_m models in a generation are trained within $4t_{\text{qhl}}$, where t_{qhl} is the time to train a single model.

13 Note we use a compact model representation, e.g. $\hat{H}_i = \hat{\sigma}^{yz}_{(1,2)} \hat{\sigma}^{z}_{(1,3)} = \hat{\sigma}^{y}_{(1,2)} + \hat{\sigma}^{z}_{(1,2)} + \hat{\sigma}^{z}_{(1,3)}$.

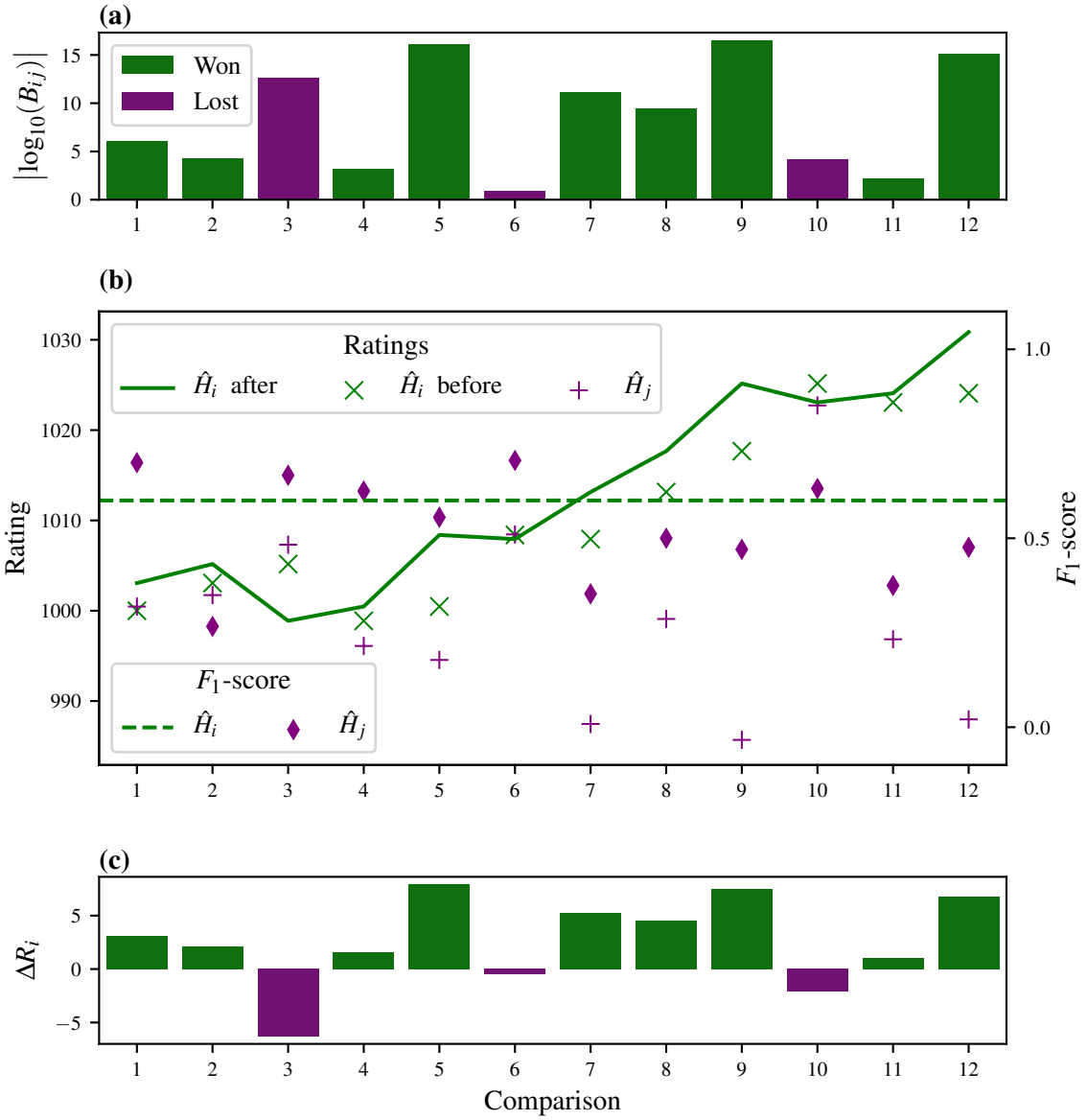

Figure 1.5: Progression of Bayes factor enhanced Elo ratings for a single candidate, $\hat{H}_i$, within a single generation. The x-axis marks successive comparisons for $\hat{H}_i$ against a subset of models from the same generation (see main text). **a**, The BFs between $\hat{H}_i$ and some opponents, $\{\hat{H}_j\}$, from the perspective where $\hat{H}_i$ *wins* given $B_{ij} > 1 \Rightarrow \log_{10} B_{ij} > 0$, and *loses* otherwise. **b**, $\hat{H}_i$'s rating is shown (solid green line) changing according to the BFs comparisons with 12 other models from the same generation. Before each comparison, $\hat{H}_i$'s rating is shown (green cross) as well as the rating of its opponent, $\hat{H}_j$ (purple plus). The F_1-scores are also shown for $\hat{H}_i$ (dashed green line) and $\hat{H}_j$ (purple diamond). **c**, The corresponding change in $\hat{H}_i$'s rating, ΔR_i. Implementation details are listed in **??**

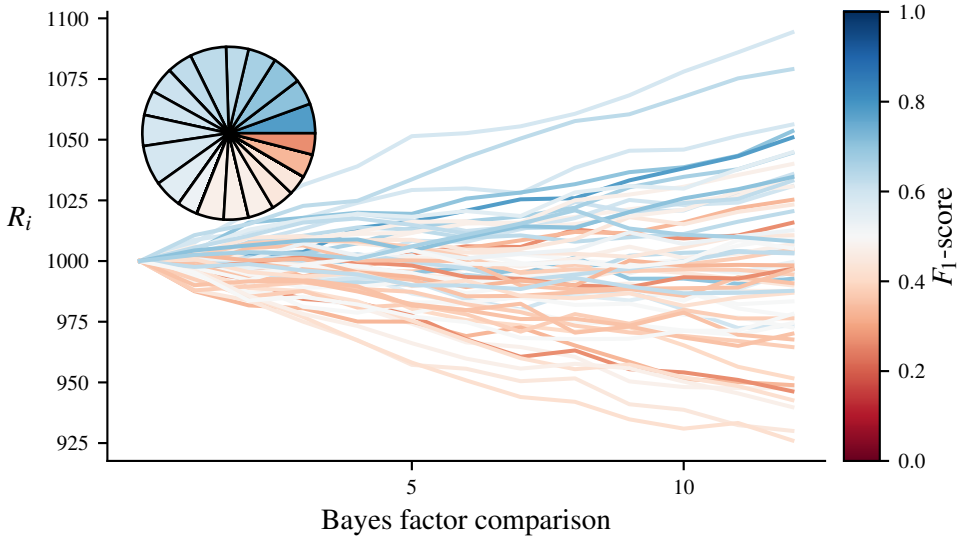

Figure 1.6: Ratings of all $N_m = 60$ models in a single genetic algorithm generation. Each line represents a unique model and is coloured by the F_1-score of that model. The x-axis marks successive comparisons for $\hat{H}_i$ against a subset of models from the same generation (see main text). **Inset**, the selection probabilities resulting from the final ratings of this generation. Only $\tau = 1/3$ of models are assigned selection probability, while the remaining poorer-performing models are discarded. Implementation details are listed in **??**

and effects on the rating of a random model, $\hat{H}_i$, where $N_m = 60$ and $N_i^{BF} = 12$, i.e. $\hat{H}_i$ is directly compared against 20% of contemporary models on μ. We see that $\hat{H}_i$'s rating is effected by whether it wins a given comparison, but also by the strength of evidence provided by the comparison (the BF), and the quality of its opposition $\hat{H}_j$, i.e. the initial rating of $\hat{H}_j$. For example, the sixth comparison finds $\hat{H}_j$ as the superior model, but the evidence is relatively weak and $\hat{H}_i, \hat{H}_j$ began with similar ratings, so R_i is not effected drastically.

We extend the single model analysis of Fig. 1.5 to all N_m models in the first generation in Fig. 1.6. The general trend is that models of higher F_1-score have their ratings increased, at the expense of models of lower F_1-score. After assessing models thus, the set of models is truncated with rate $\tau = 1/3$ to retain only the strongest 20 candidates, which are assigned selection probability, i.e. their chance of being chosen to become a parent during roulette selection, as in **??**. N_m models are required to populate the next generation: the two models with highest R_i – the *elite* models – are automatically granted a position; the remaining positions are filled through the crossover procedure outlined above.

We can likewise consider the quality and ratings of models across generations. In Fig. 1.7**(a)** we see the ratings for models over the first four generations of a QMLA instance: the trend suggested by Fig. 1.6 continues, where models of higher F_1-score tend to achieve higher BFEER.

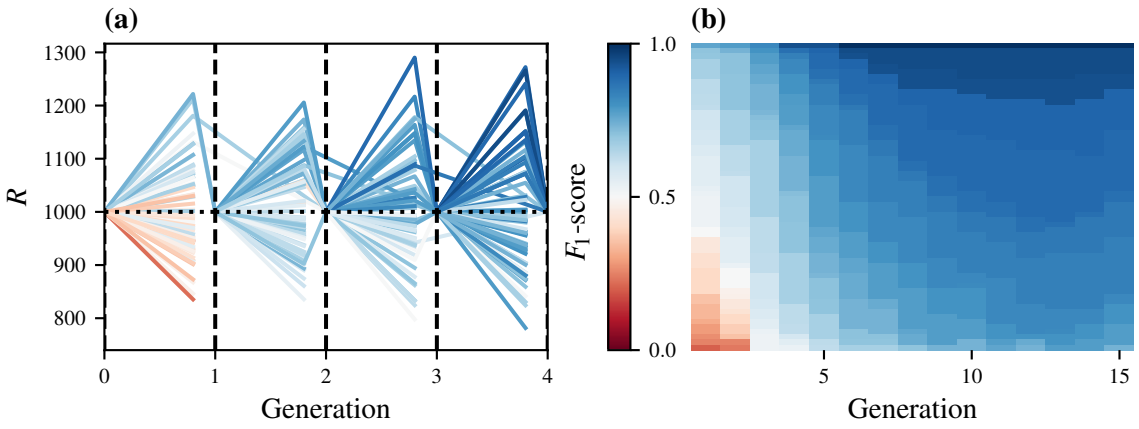

Figure 1.7: A single instance of the QMLA genetic algorithm. **a**, Ratings of all models for the first four generations. Each line in each generation represents a model by its F_1-score through colour. Horizontal dotted lines show the starting rating at that generation. **b**, Gene pool progression for $N_m = 60, N_g = 15$. Each tile at each generation represents a model by its F_1-score through colour. Implementation details are listed in **??**

The gene pool as a whole tends towards a homogeneous set of high-quality models: the final generation consists only of models with $f \geq 0.85$, Fig. 1.7**b**. Consequently, even in instances where the precise model, $\hat{H}_0$, is not identified, the champion model is highly informative, in that it captures many of the same interactions, therefore most-likely providing meaningul insight on the system's physics.

Finally, to understand the performance of the QMLA algorithm overall, we combine 100 independent instances in a run, Fig. 1.8. We see that, while the overall model space can be characterised by a distribution of models with $\bar{f} = 0.5 \pm 0.14$ (Fig. 1.8**(a)**), QMLA quickly moves to the subspace of high-quality models, i.e. the models explored have median $f = 0.76 \pm 0.15$ (Fig. 1.8**b**). This exploration is based on 430 ± 45 chromosomes per instance, i.e. QMLA trains only 0.16% of the 2^{18} permitted models. Ultimately QMLA nominates champion models $\{\hat{H}'\}$ with $f \geq 0.88$ in all instances, and precisely identifies $\hat{H}' = \hat{H}_0$ in 72% of instances, Fig. 1.8**(c)**. Considering the big picture – where the remit of QMLA is to identify the interactions Q undergoes – we show the rate at which each individual term/gene is included in $\hat{H}'$ in Fig. 1.8**(d)**. Crucially, we see that terms which really are within the true Hamiltonian, $\hat{t} \in \mathcal{T}_0$, are found at a higher rate than those without, $\hat{t} \notin \mathcal{T}_0$. This level of analysis can be used to post-validate the outcome of QMLA, i.e. rather than relying on $\hat{H}'$ from a single instance, trusting the terms' individual frequencies as evidence that they are of importance when describing the system of interest.

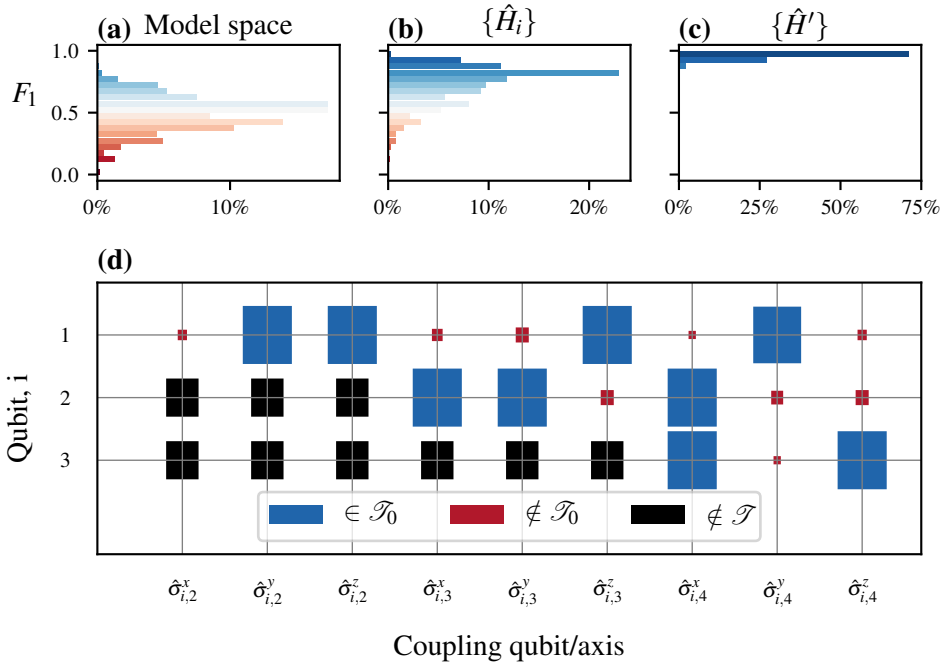

Figure 1.8: A run of the QMLA genetic algorithm (GA), consisting of 100 independent instances. **a**, The model space contains $2^{18} \approx 250,000$ candidate models, normally distributed around $f = 0.5 \pm 0.14$. **b**, The models explored during the model search of all instances combined, $\{\hat{H}_i\}$. QMLA generates $\sim$ 43,000 chromosomes in total across the 100 instances, i.e. each instance trains $\sim$ 430 distinct models; Models generated by QMLA are described overall by a distribution of $f = 0.76 \pm 0.15$, **c**, Champion models from each instance, showing in general QMLA nominates cahmpion models with $f \geq 0.88$ in all instances, and in particular finds the true model $\hat{H}_0$ (with $f = 1$) in 72%. **d**, Hinton diagram showing the rate at which each term is found in the winning model. The size of the blocks show the frequency with which they are found, while the colour indicates whether that term was (not) in the true model in blue (red). Terms represent couplings between two qubits, e.g $\hat{\sigma}^x_{(1,3)}$ couples the first and third qubits along the x-axis. Available terms involve four qubits with full connectivity, resulting in 18 unique terms (terms with black rectangles are not considered by the GA). Implementation details are listed in **??**

1.3.2 *Device calibration*

The use-case presented in this chapter is restrictive, so cannot be considered as a solution to characterising any black box quantum system. Firstly, the set of conceivable terms must be prescribed in advance to facilitate a chromosome mapping; this either limits the range of insight QMLA can achieve to interactions envisaged by the user, or requires a vast set of permissible terms, leading to substantially longer model search phases than shown here. Secondly, models were trained using IQLE in order to learn effectively with relatively few resources. IQLE is only available to train models where we can reliably reverse the evolution of the target system (see **??**), and as such it is only useful when we have complete control over Q, for example where Q is some quantum simulator.

The adaptive GES presented in this chapter may therefore 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. QMLA can effectively reconstruct which operations were and were not faithfully computed, i.e. determine in which operations the device failed to perform the intended calculations, allowing for the calibration of said device. The extension of QMLA to the characterisation of real quantum devices is one of the most promising applications for future research in the scope of the QMLA framework beyond this thesis.

Part IV

EXPERIMENTAL STUDIES

Part V

CONCLUSION

2

OUTLOOK FOR MODEL LEARNING METHODOLOGIES

Optimal control techniques are a crucial component in improving quantum technologies, such that imperfect near-term devices may be leveraged to achieve some meaningful quantum advantages. The developments presented in this thesis contribute to the growing interest in automatic characterisation and verification of quantum systems and devices. Namely, the introduction of the Quantum Model Learning Agent (QMLA) represents an important advancement, whereby quantum systems can be completely characterised starting with little prior knowledge. The majority of this thesis was dedicated to the rigourous testing of QMLA, gradually moving from ideal scenarios in simulation to genuine experimental quantum systems.

We described the implementation of QMLA as an open source software platform in Part II, detailing numerous tunable aspects of the protocol, and their impact on training candidate models in **??**. QMLA facilitates customisation of its core elements and subroutines, such that it is applicable to a wide range of target quantum systems, as described in **??**–**??**. This malleability enables users to easily adapt the framework to their own needs, and formed the basis for the cases studied in the remainder of the thesis: we tested QMLA by devising a series of exploration strategies, each corresponding to a different target quantum system.

In Part III we considered ideal theoretical quantum systems in simulation. Initial tests in **??** showed that QMLA could distinguish between different physical scenarios and internal configurations. In Chapter 1, we explored much larger model spaces by incorporating a genetic algorithm (GA) into QMLA's model design; the GA showed promise for characterising complex quantum systems by successfully identifying the target model. The performance of the GA, however, came at the expense of relying on a restrictive subroutine – used for training individual candidate models – drastically reducing its applicability to realistic systems. However, the restriction is permitted in the scope of characterising *controlled* quantum systems, for example new, untrusted quantum simulators.

We concluded the thesis by considering realistic quantum systems in Part IV. Experimental data from an electron spin in a nitrogen-vacancy centre (NVC) was treated in **??**; this too relied upon tailoring QMLA's procedure with respect to the system under study. A theoretically justified Hamiltonian is proposed by QMLA to describe the decoherence of the electron spin, yielding a highly predictive model in agreement with the system's measured dynamics, albeit exploring a small model space. To overcome concerns that the model search was artifically constrained in the context of realistic systems, **??** exercised QMLA in a vast model space, spanning terms which represent plausible interactions for the same type of system. Here, again, QMLA achieved high success rates, but with caveats on the subroutines assumed for model training, and resorting to simulated data.

In summary, this thesis has provided extensive tests of the QMLA algorithm, but each may be undermined by its individual constraints. In outlook, near-term developments of model learning

methodologies in the context of quantum systems must address these shortcomings, for instance by unifying the strategies described in this thesis. Further, we anticipate immediate application in the study of open quantum systems, by replacing the Hamiltoinian formalism examined here with a Lindbladian representation, permitted within the QMLA apparatus. Through the advancements presented herein, we hope to have provided a solid foundation upon which these constraints may be relaxed, ultimately with a view to providing an automated platform for the complete characterisation of quantum systems. We envision QMLA as a straighforward but powerful utility for quantum engineers in the design of near term quantum devices, expecting continued development of the framework alongside the burgeoning open-source quantum software eco-system.

BIBLIOGRAPHY

[1] Antonio A. Gentile, Brian Flynn, Sebastian Knauer, Nathan Wiebe, Stefano Paesani, Christopher E. Granade, John G. Rarity, Raffaele Santagati, and Anthony Laing. Learning models of quantum systems from experiments, 2020. Accepted: Nature Physics.

[2] Brian Flynn, Antonio A. Gentile, Raffaele Santagati, Nathan Wiebe, and Anthony Laing. Quantum model learning agent: quantum systems' characterisation through machine learning. In preparation, 2021.

[3] Brian Flynn. Codebase: Quantum model learning agent. https://github.com/flynnbr11/QMLA, 2021.

[4] Quantum model learning agent documentation. https://quantum-model-learning-agent.readthedocs.io/en/latest/, Jan 2021. [Online; accessed 12. Jan. 2021].

[5] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[6] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[7] Eyal Bairey, Itai Arad, and Netanel H Lindner. Learning a local hamiltonian from local measurements. *Physical review letters*, 122(2):020504, 2019.

[8] Nathan Wiebe, Christopher Granade, Christopher Ferrie, and David Cory. Quantum hamiltonian learning using imperfect quantum resources. *Physical Review A*, 89(4):042314, 2014.

[9] Burnham KP Anderson DR. Model selection and multimodel inference: a practical information-theoretic approach, 2002.

[10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[11] Christopher Granade, Christopher Ferrie, Steven Casagrande, Ian Hincks, Michal Kononenko, Thomas Alexander, and Yuval Sanders. QInfer: Library for statistical inference in quantum information, 2016.

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

[13] FIFA.com. Who we are - news - 2026 fifa world cup™: Fifa council designates bids for final voting by the fifa congress, Jun 2018.

[14] Christof Neumann, Julie Duboscq, Constance Dubuc, Andri Ginting, Ade Maulana Irwan, Muhammad Agil, Anja Widdig, and Antje Engelhardt. Assessing dominance hierarchies: validation and advantages of progressive evaluation with elo-rating. *Animal Behaviour*, 82(4):911–921, 2011.

[15] Lars Magnus Hvattum and Halvard Arntzen. Using elo ratings for match result prediction in association football. *International Journal of forecasting*, 26(3):460–470, 2010.

[16] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.