The Turing Ratio: A Framework for Open-Ended Task Metrics

Hassan Masum **Steffen Christensen** **Franz Oppacher**
Department of Computer Science , Carleton University
{hmasum, schriste, oppacher}@scs.carleton.ca

Abstract

The Turing Test is of limited use for entities differing substantially from human performance levels. We suggest an extension of Turing's idea to a more differentiated measure - the "Turing Ratio" - which provides a framework for comparing human and algorithmic task performance, up to and beyond human performance levels. Games and talent levels derived from pairwise comparisons provide examples of the concept.

We also discuss the related notions of intelligence amplification and task breadth. Intelligence amplification measures total computational efficiency (the computational benefit gained relative to investment, including programmer time, hardware, and so on); we argue that evolutionary computation is a key amplifier of human intelligence. Task breadth is an attempt to weight Turing Ratios by the frequency and importance of the task they measure - doing well at a broad range of tasks is an empirical definition of "intelligence".

Measuring Turing Ratios and considering task breadth, prior knowledge, and time series of the measures may yield long-term insight into both open-ended computational approaches and the underlying task domains being measured.

1 INTRODUCTION

1.1 PROBLEM: MEASURING MIND

The Turing Test was proposed by Alan Turing in [Turing 1950]. By providing an operational benchmark which could plausibly be passed by an entity only if that entity was in some sense intelligent, Turing moved the discussion of artificial intelligence to a new and more productive plane.

However, task performance for open-ended tasks (like Turing's suggestion of unrestricted conversation with a human) has not yet been formalized, in a manner allowing easy comparison across task domains and time scales. As well, no program is close to being ready to take a broad-based intelligence test.

Our goal, then, is to generalize Turing's idea to formalize "task performance". Desirable features include comparing humans and algorithms on the same scale, defining a scale that remains valid across many domains and over long periods of time, and making the definition as quantitative as possible (but no more). We'd like to come up with an objective rating for hard and open-ended tasks.

In the future, boundaries between computational and human intelligence may blur, given sufficient algorithmic advances. Hence we may wish to rank computational performance directly against humans, or against an objective standard; this allows clearer comparisons and time series extrapolations.

1.2 A PARTIAL SOLUTION: THE TURING RATIO

We propose a framework: measure the performance of computational entities on well-defined yet open-ended tasks, and compare these performances with each other and with human performance. Clearly, a key issue in using such a measure is defining a practical algorithm to assess task performance in a particular domain. While perhaps not always possible, we point out several domains for which performance has successfully been defined - competitive games in particular. The observed objective and reproducible measure of task performance provides a test for deciding the level of talent of a participant (insofar as task performance is sufficient to indicate talent).

To the extent that domains can be assigned Turing Ratios, these domains could be ranked by increasing Ratio to quantify the "automatizability" of different kinds of tasks. This will give us clues about the intrinsic domain difficulty - perhaps we will confirm our current intuition about what tasks are difficult, or perhaps there will be surprises in store (e.g. computer chess players). Given ratings of composite tasks that involves many interrelated sub-tasks, we may be able to note a set of "basis tasks" that are common across many composite tasks; such a basis task set would form a more nuanced Turing Test Suite.

With human and computer performance on the same scale, relative economic and cognitive strengths will become clear. Taking into account estimates of Moore's Law and algorithmic advancement, Turing Ratio trends may suggest tasks becoming economically unviable for humans, and point out situations currently overloading human cognitive limits that may be alleviated via algorithmic assistance.

We discuss refinements to this basic idea: the amount of prior knowledge used by a program gives insight into the robustness and domain-specific customization currently necessary for good performance in the given domain. Learning from scratch is more impressive than being thoroughly customized for a domain, and this is a strength of Evolutionary Computation (EC) that is quantifiable in principle. Breadth of the domain in which good performance has been achieved is also important; one practical definition of intelligence is the ability to survive in diverse environments.

Turing Ratios can quantify empirical algorithmic progress, generate objective data for inference about the algorithmic properties of hard tasks, and isolate areas of human strength.

1.3 RELATED WORK

In [Koza 1999], domains are listed in which computers have outperformed humans who tried the same problems. As well, benchmarks for ranking genetic programming results as human-equivalent in a domain are suggested; most are variants of the idea that a reputable publication, patent, or competition result should be regarded as valid evidence of achievement regardless of the source.

Many variants of the Turing Test have been proposed; the extensive discussion in the half century since Turing's seminal paper is reviewed in [French 2000]. As a signal indicating the presence of strong AI, the Turing Test is arguably as good as any other measure yet devised. However, it has the drawback of being relatively binary; there is little provision for partial success, and hence the Test may not be of use until researchers have neared the goal of creating human-equivalent AI. (See [Saygin et al 2000] for further discussion of the Test and its instantiation in the Loebner Prize competition.)

Finally, science fiction provides scenarios in which open-ended "games" and their rankings pervade a society. [Anthony 1980] is the first of a trilogy describing a society where "The Game" formalizes competition in many tasks, and is the sole method for serfs to gain social status: "The Game is violence, or intellect, or art, or chance, alone or with tools or machines or animals--but mainly it is challenge."

2 THE TURING RATIO FRAMEWORK

2.1 ASSUMPTIONS AND SUITABLE TASKS

No single-dimensional measurement can adequately capture all the varieties of "intelligence". Hence, we restrict our domain of discourse for the Turing Ratio (TR) to measurement of task performance - the degree to which task performance implies higher cognitive abilities will clearly be task-dependent. Some key assumptions are necessary:

- Task performance is unambiguously defined and reproducible.
- Task-solving entity is clearly delineated.
- Finally, task performance should be adequately represented with a single-dimensional variable.

Although the single-dimensional assumption may seem restrictive for open-ended tasks, the original Turing Test manages to cleverly reduce conversational performance to a single variable: number of judges who agree that the speaker is human. For multiobjective tasks with stable subobjectives, the subobjectives could be treated independently.

In the current paper we restrict our attention to those domains for which a total ordering can be induced, through either a score metric or a more complex transformation. Extension to multiobjective and partially-ordered cases is deferred for future research, and might benefit from methods for comparing non-dominated solutions from the fields of multicriteria decision-making and multiobjective optimization. See also the later section on task breadth; solving a composite task requiring competence at many subtasks is often a multiobjective problem, as one compares the relative merits of different approaches.

Coming up with tasks that encapsulate other interesting domains is a challenge (the "progress parameterization problem") akin to finding the right fitness function for a complex EC instance. This suggests that task measurements for domains where quantizing performance is non-obvious might be derived from fitness functions for EC programs for those domains, and vice versa.

In addition to the necessary conditions above, we suggest three properties a task should have for the Turing Ratio to be a suitable measure on it:

1. Task measures an important ability.
2. Task admits a series of increasingly better strategies, which are qualitatively different.
3. Reproducible and relevant for decades.

To the degree that a particular task requires intelligent solution methods, a high TR value for that task indicates the presence of intelligence.

2.2 MEASUREMENT TYPES AND STATISTICS

There are four basic types of single-dimensional measurement: nominal, ordinal, interval, and ratio. These correspond to different types of TR measurement tasks:

Type	Definition	Example
Nominal	Qualitative classification into one of several categories, without directly ranking the categories.	A single Turing Test, rating a conversant as human or not (although this boolean case might also be viewed as an ordinal case).
Ordinal	Contains ranking data, but no information on the distance between successive ranks.	Games are a prototypical task.
Interval	Describes task measurements without a known absolute zero point.	Many test scores.
Ratio	Ratio scales give the most information, possessing an absolute zero point.	Computational complexity, where time or space for an algorithm is precisely known.

What happens with repeated task measurements? In the context of TR, statistics confer two main benefits: the reduction of uncertainty, and synthetic generation of more detailed task measurements. The first benefit is widely understood, e.g. multiple Turing Test judges induce a binomial random variable with associated confidence intervals; [Jain 1991] discusses similar "classic" performance measurement.

The second benefit of synthetic derived variables is easily understood in the context of games. Given a particular ordinal task measurement - say a win or loss against a game opponent - we can infer only a limited amount about underlying abilities. However, repeated trials allow an increasingly precise demarcation to be made between various ability levels, generating a synthetic interval or ratio scale - an issue we discuss further in our chess example. For competitive situations like the game example, it's important to note that repeated single-dimensional measurements are limited in the resolution of the derived synthetic variable by the number of measurements, the variability of the task performer, and (to the degree that abilities are not totally ordered) the distribution of opponents.

2.3 ABSOLUTE AND RELATIVE TR

We distinguish between absolute and relative task performance. The intuition is that a well-defined algorithmic task that can be measured in isolation is an absolute task, while a game against opponents is a relative task depending on the current opponent ability distribution. (An analogy is evolution "against" a static environment, versus coevolution against adaptive opponents.)

An Absolute Turing Ratio is therefore defined as a TR independent of when the result was achieved, the performance levels of others on the task, and the competence level of the agent doing the performance evaluation (past a minimum threshold requirement). Note that this is a strong requirement: most open-ended tasks rely directly or indirectly upon fitness comparisons between competitors. (The computational tasks used in algorithmic benchmarks such as combinatorial optimization problems are absolute tasks - but they do not seem to be open-ended, in the sense of admitting an unbounded spectrum of qualitatively more complex solutions.) What would be required is a completely specified task, measuring important abilities, valid for decades, admitting a series of increasingly better and qualitatively different strategies, where the task is performed without reference to other competitors and evaluated algorithmically.

We regard it as an open challenge to specify such tasks. Ideally, they would be "human-complete", in analogy with NP-complete tasks. Turing's task of conversation imitation, for example, is arguably rich enough to be a sufficient indicator for human-level thought (if not necessarily necessary, as Turing himself noted). It is not an Absolute task, though, since the worth of the evaluation depends strongly on the competence and insight of the human judges. Tasks in the Absolute category should be computationally verifiable, given a reasonable amount of computational resources. Note again the strong analogy with the complexity class NP; we don't mind arbitrarily complex solution strategies being used for a task, as long as we can verify the resulting task performance level using an implementable algorithm.

In contrast to an Absolute Turing Ratio, a Relative Turing Ratio result needs to specify the opponent distribution, along with the specific context and environment of the task performance. If the task is widespread and unambiguous enough, it may be sufficient to simply specify the time period in question, e.g. the chess environment circa 2000. Relative Turing Ratios, like most measurements of higher-level abilities in everyday life, are made with reference to the performance of others. (Given an Absolute task, one can usually induce a related Relative one by comparing the performance of different problem-solvers on the Absolute task.)

If Absolute versus Relative is not specified, one can assume Relative. This is due to the current lack of well-specified Absolute task domains - a key area to be formalized.

Usually a given algorithm will vary its performance depending on the resources available to it: time and space are the two typical ones. More generally, having access to an "oracle" that performs a specified black-box function may make a substantial performance difference, e.g. access to the Internet. Thus, TR ratings should specify CPU, memory, and any salient modules or prior information incorporated.

We note that the requirements of resource and opponent specification are consequences of the original assumption that task performance be unambiguous and reproducible. This is essential for systematic comparison between problem domains, solution methods, and task performance over time.

2.4 FORMAL SPECIFICATION

Let's look at two examples of formally specifying TR. First, consider the general specification. Given a program p for a specified task domain D (where p belongs to P, the population of competitors), the computational performance of p in D will be denoted by $\Phi(p, D)$. (Φ returns one of the four variable types defined previously.) We are then interested in measuring:

$$\Phi(p, D, TimeStamp, Resources, P)$$

Resources could be any set of computationally-useful resources; if one or more of these resources can be parameterized by a scalar variable, we can do some informative statistical extrapolation, as we'll see. (P would not need to be specified for an Absolute Turing Ratio result.) One could also include additional information in a TR measurement; the underlying idea is that a well-specified minimal set of information should always be present, so that long-term analysis can usefully be accomplished, including perhaps reparameterization of TR based on a user's judgement.

In the case of n repeated trials, we get:

$$\Phi^n(p, D, Resources, ...)$$

i.e. Φ repeated n times. If Φ is a nominal variable, then Φ^n will be drawn from a binomial or multinomial distribution; in the binomial case, we can use a Beta distribution to infer the probabilities of Φ taking each possible value.

Often, we are interested in comparing two performances (either of distinct programs or of the same program with different resources):

$$\Phi_{Relative}(\Phi(p_1), \Phi(p_2))$$

This should be a function returning a meaningful "ratio": three empirically-common options are simple division (in the ratio-variable case), differences in log performance (so a constant increase in TR implies a constant multiplicative factor increase in Φ), and percentile differences (i.e. relative population ranking).

Finally, we note that the spatial distribution of TR scores could be of interest; this can be modeled by placing competitors in some graph, and measuring TR relative to other scores within some distance.

Now consider a specific program and task. We measure:

- *CompScore* = $\Phi(p, D)$.
- *MaxCompScore* = max $\Phi(p_i, D)$ over all programs p_i in *D*.
- *RefHumanScore* = e.g. max or median $\Phi(Human_i, D)$ over all human performances in *D*.

Then we can define the program's TR, and similarly a domain-wide TR for the best computational performance:

$$TR(p, D) = \Phi_{Relative}(CompScore, RefHumanScore)$$
$$TR(D) = \Phi_{Relative}(MaxCompScore, RefHumanScore)$$

An actual TR measurement would contain more detail - enough to unambiguously specify and reproduce the result, including particularly the domain, resources, and program. The construction of $\Phi_{Relative}$ may also change over time, depending on the opponent distribution and the meaning of a particular score.

We can divide task domains into three general classes by their domain-wide TR:

1. TR(D) << 0 dB (algorithm much worse than human)
2. TR(D) >> 0 dB (algorithm much better than human)
3. TR(D) ~ 0 dB (rough equality)

The interesting domains at any point in time will be those on the ever-expanding "Turing Frontier", where algorithmic task performance is improving rapidly relative to human performance.

3 TURING RATIO MATH AND EXAMPLES

3.1 DERIVING GLOBAL RATINGS FROM PAIRWISE COMPETITIONS

How can we derive ratings objectively? One widely-used techique is through pairwise competitions, e.g. in games; see [Smith 1993] for a discussion of game-player rating systems.

Consider a tournament with N players, P_1 to P_N. Each pair plays a series of games, and we record the series winner. Now make the "winner digraph", with the arc between each pair of nodes pointing from the winner to the loser.

Given this winner digraph, how can we order the players objectively? Simply find a Hamiltonian path - this will form a total ordering of all nodes. But that's NP-complete...so let's make the simplifying assumption for the moment that no cycles exist. (If just a few cycles exist, condense strong components into "class nodes", and then order class nodes.) Then the outdegree sequence of the nodes is (0,1,...,n-1) - so we can find a Hamiltonian path in time $O(|E| + |V|\log|V|)$, by just recursively removing the node with maximal outdegree and adding it to the ordered list of players. This path is our objective ordering.

Generalizing, we can let the results of each pair (P_i, P_j) define a "win probability", as measured by the frequency with which each player wins. Instead of an ordinary digraph, we get a weighted digraph, with an associated win matrix:

$$\mathbf{W}: w_{ij} = \textit{probability of } P_i \textit{ beating } P_j$$

Now we can derive more than just an ordinal ranking of 1st - 2nd - 3rd. Let $\mathbf{p}^W$ be the win measure vector that we seek, which will summarize the relative strengths of different players as described by the observed win matrix. We want $\mathbf{p}^W$ to be the vector from all candidate win measure vectors $\mathbf{p}$ such that $\mathbf{p}_i / (\mathbf{p}_i + \mathbf{p}_j)$ best approximates the observed w_{ij}, i.e. Pr (P_i beats P_j).

Let $\mathbf{W}(\mathbf{p})$ be the candidate win matrix induced by any candidate $\mathbf{p}$, e.g. the matrix formed from $\mathbf{p}$ under the assumption that $\mathbf{p}_i / (\mathbf{p}_i + \mathbf{p}_j)$ gives the exact win probability Pr (P_i beats P_j):

$$\mathbf{p} = \begin{bmatrix} 10 \\ 5 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{W(p)} = \begin{bmatrix} 1/2 & 10/15 & 10/12 & 10/11 \\ 5/15 & 1/2 & 5/7 & 5/6 \\ 2/12 & 2/7 & 1/2 & 2/3 \\ 1/11 & 1/6 & 1/3 & 1/2 \end{bmatrix}$$

Then given an observed $\mathbf{W}$, find the closest $\mathbf{p}^W$:

$$\mathbf{p}^W = \textit{arg min}_{\mathbf{p}} (\| \mathbf{W} - \mathbf{W}(\mathbf{p})\|_2)$$

This $\mathbf{p}^W$ is the objective ranking we seek, found by least squares from $\mathbf{W}$. We have "projected" a weighted digraph (containing N "rankings" per player) onto a single relative rank for each player.

3.2 DEALING WITH LIMITED DATA

While a straightforward and intuitive procedure, there are several real-life statistical issues to consider:

- Most of the time, we don't have exact win probabilities between pairs (P_i, P_j). Rather, we have an observed series of wins and losses. While (Wins / Wins + Losses) should approach $\mathbf{p}_i / (\mathbf{p}_i + \mathbf{p}_j)$ in the limit if player strengths are stationary, we can do better for modest numbers of observed games.
- The number of games played between each pair of players should be taken into account - less games played implies a greater imputed variance to the derived win measure.
- We may not have information on all pairs of players. In fact, many real situations will have a sparse matrix of player evaluations. So we need a way of wisely inferring a uniform measure across players which do not interact directly.

Consider now a case where some pair (P_i, P_j) plays just one game: say P_i wins. Then a first naive estimate of the win probability is

$$Pr(P_i\ wins) = (\#\ P_i\ wins) / (\#\ P_i\ wins + \#\ P_j\ wins)$$

which in this case gives 1 / (1+0) or 100%. This is not a reasonable inference from just a single win.

A workaround often used is to add an initial imaginary tie game to both players. This gives

$$Pr(P_i\ wins) = 1.5 / (1.5 + 0.5) = 3/4$$

However, this is somewhat ad-hoc and we can do better.

The idea is to infer probabilities and confidence intervals from integrating the likelihood PDF over potential win probability values. Then we simply solve for the win probability value that gives us a median value of 0.5 in the CDF (cumulative distribution function).

The details are as follows - we will consider just a single pair of players (P_1, P_2) at first:

n - observed # games played
k - observed # successes of P_1
p - possible probability of P_1 beating P_2 - this is integrated over all possible values to find the true probability p_{True}, as illustrated below
p_{True} - true probability of P_1 beating P_2

Then the procedure is to solve for p_{True} in this equation:

$$\frac{\int_0^{p_{true}} \binom{n}{k} p^k (1-p)^{n-k} dp}{\int_0^1 \binom{n}{k} p^k (1-p)^{n-k} dp} = 0.5$$

(The denominator is a normalization constant, and we solve for the upper limit of integration in the numerator which sets the whole left side equal to a median likelihood.) For example, the case of 1 win and 0 losses described above gives an integral equation which reduces to $p_{True} = 0.5^{1/2}$, or approximately

0.7071. (In fact, W wins and 0 losses gives p_{True} as $0.5^{1/W}$.)

Now, let's review what we have achieved. Recall that in the initial case where we assumed data for exact talent levels between all pairs of players, we were looking for a win measure vector $\mathbf{p}^W$ which minimized the error between the observed win matrix **W** and the synthetic win matrix **W**(**p**) induced from the win measure vector. In our current case, we derive talent levels for each pair of players using the procedure described. These talent levels form the data for the "observed" win matrix **W** - the change is that we have used an estimation procedure to get a good estimate of pairwise talent levels, given a limited number of observed pairwise competitions. (This should be a more accurate procedure than the naive estimation method of simply taking ratios of observed wins and losses, especially for low numbers of observations. Both methods should converge to the true talent levels in the limit.)

(The synthetic win matrix **W**(**p**) is induced from the win measure vector by using the definition of our win measure vector: $\mathbf{p}_i / (\mathbf{p}_i + \mathbf{p}_j)$ should give us the probability that P_i beats P_j, so the entries of **W**(**p**) are filled in by taking this ratio for each possible pair of players. A little more cleverness is required to deal properly with the case where two players have not competed directly; details can be found in the academic literature on talent level inference.)

Finally, to take account of different numbers of games played between different pairs of players, we can add a weight matrix. Each element of this weight matrix stores the number of games played between the corresponding pair of players. The weight matrix then scales the contribution of each pair of players to the matrix difference being minimized.

3.3 RELATIVE TURING RATIOS FROM GLOBAL TALENT LEVELS

Consider two agents A and B playing a stochastic game involving personal talent against one another. Let us label the talent levels of A and B as *a* and *b* - note that these are global talent levels across the entire player population (as found e.g. from the procedure described above). A simple model sets the win expectancy for player *A* as:

$$p_{win}(A) = a/(a + b)$$

If we take the ratio of the ratings of two individuals playing, we can let ρ be a/b, and we get the single-parameter form:

$$p_{win}(A) = \rho / (1 + \rho);\ p_{win}(B) = 1 / (1 + \rho)$$

Intuition and empirical evidence suggests a logarithmic scale would be best to evaluate the wide range of abilities in most domains; one candidate is the decibel scale commonly used to measure signal strength. We therefore define:

$$TR \text{ (in dB)} = 10 \log_{10} (\rho_{measured}) - ReferenceAbility$$

We suggest the convention of letting the median human performance in a domain, where ascertainable, be given the reference level 0 dB. *CompScore* and *MaxCompScore* are then the empirically measured TR values of the current and best program respectively. This Turing Ratio scale represents each person's performance as a real-valued parameter. (The dB unit is easy to understand; however, TR could be quoted in other units.)

Note that deriving the parameter ρ from observed wins and losses can be a complex procedure. We are reducing a two-dimensional matrix of observed wins and losses to a single-dimensional measure; this can only be done in a reasonable way if the original matrix is well-behaved (e.g. there are not too many transitive winner cycles A-B-C-A).

Even if the measure can be found, it may be only approximate due to measurement and performance errors, performance nonstationarity, and so on; methods for dealing with such issues have been developed in e.g. statistical inference and psychological experimentation. Greater variation in performance implies more measurements will be necessary to derive accurate ratings; greater unavoidably-stochastic components to the game imply a decreasing upper bound to the best rating achievable. We note that a practical implementation of the Turing Ratio would need to consider all these issues; [Glickman 1999] outlines one scheme to infer ratings from large competitions.

3.4 A CASE STUDY: CHESS RATINGS

Our TR scale is not without precedent; there is a close relationship with the United States Chess Federation rating system. We outline the similarities and show how chess programs and humans can be effectively rated on the same scale.

The USCF rating system rates a decimal order of magnitude performance difference at 400 rating points [USCF 2002]. Therefore:

$$\textit{USCF rating} = 40(\textit{dB rating}) + \textit{HumanMedian}$$

where *HumanMedian* is median human chess performance, around 600 USCF rating points. We can even use the USCF rules directly to measure performance in any pairwise competitive endeavour where the possible outcomes are Win/Lose/Draw [USCF 2002].

In addition, the Swedish Computer Chess Association (SSDF) has approximate empirical scaling laws for size of the in-memory hash tables used in computer chess programs, and for performance at chess as a function of processor speed. (Note that the SSDF calibrates its point scheme against the World Chess Federation (FIDE) system, which uses a Gaussian distribution scheme with a slightly larger spread then the USCF. In FIDE, there are roughly 410-points in an order-of-magnitude performance difference (derived from a Levenberg-Marquardt fit for A in the function $p_{win} = 1/(10^{-\rho R/A} + 1)$ to the FIDE win expectation curve for rating differences from –100 to 100 [FIDE 2000].)

As quoted in the SSDF FAQ [Grottling 1998], Kathe Spracklen estimated the effect of doubling the in-memory hash tables at 7 ratings points, or a performance factor of $10^{7/410} = 1.04$, about a 4% improvement in relative Turing Ratio. The same FAQ also estimates the effect of doubling processor speed at 70 ratings points, for a performance factor of $10^{70/400} = 1.48$, or a 48% improvement.

We thus have a practical case where both computer and human performance have been compared on the same scale, in a Turing Ratio framework.

How would this apply to deriving a TR for Deep Blue, the chess-playing computer that beat the human world champion? In this case, we'd say that its TR is indeed greater than 1, as measured by wins and losses against human players. In principle, the procedure that we outlined previously for deriving ratings from pairwise comparisons could have been used to get an accurate rating, by playing against a series of (grandmaster level) human players.

Now, if Deep Blue were to continue to improve (so its TR became much greater than 1), a problem would arise - at some point it would probably win all its games, so we wouldn't be able to tell just how far beyond human it was. In this case, there are three practical possibilities:

1. Handicap it with gradually increasing disabilities such as piece handicaps or move restrictions, to bring it back into the human range.
2. Compare it directly to another superhuman algorithm (maybe "Deep Black" if contact is made with a chess-loving alien civilization?).
3. Acknowledge the demonstrated superiority of the tested algorithm in this domain, and look for more interesting domains to test.

3.5 POTENTIAL METRICS AND TASKS

A canonical metric is time to complete task, or complementarily the size of task that can be completed, for e.g. combinatorial and NP-hard problems. More generally, computational complexity theory suggests many metrics measuring aspects of problem hardness. Just as pairwise competition is a canonical domain for Relative TR, computational complexity is a canonical domain for Absolute TR. (Of course, in both cases a wise selection of the task for which ability is being measured is necessary to deduce conclusions of value.)

Games are particularly good domains due to unambiguous task scope and "complex strategies from simple rules". Checkers and chess both have computer players rated higher than any human since the 1990's, with Schaeffer's Chinook victories and Deep Blue respectively. Go programs are still well short of the best humans. Go has a well-defined rating system, and algorithms have slowly been improving in rank; [Bouzy 2001] discusses the algorithmic challenges. (Note that since human reference levels change slowly with respect to computer performance, we might take them to be fixed as a first approximation. Also, we certainly do not claim strategy games are fully open-ended tasks, but relative to their simplicity of definition they do admit a vast series of increasingly creative strategies.)

An important human rating method is tests of competence and "practical substitution": put someone in a situation and see how well they perform. Exams, aptitude tests, and IQ tests - usually relative since they are scaled to student body ability - attempt to estimate likely performance and measure current competence. (They have been widely criticized for measuring only part of "intelligence". Reduction to a single parameter probably cannot summarize a complex mind's performance - the same will be true of computational performance in broad enough domains.)

One might quantify human judgement of performance, e.g. ranking in a music competition, money earned, market or attention share, double-blind human comparison, or votes by an audience. Clearly literature, art and music are among many such tasks with very low TR at present (with exceptions in niches).

Although many computational testbed tasks are intellectual in nature, we believe TR should also be applied to tasks that are embedded in the physical world. Two practical examples of such tasks are driving in real-world conditions, and building a cockroach-killing robot - both these would fulfil very practical needs while demonstrating a wide range of competencies, and will likely admit slowly-increasing performance for decades or longer.

4 BOOTSTRAPPING INTELLIGENCE

What does a given TR value mean? One of our main goals is to define a simple operational measure, from which characteristics of performance at open-ended tasks can be empirically derived. For such analysis, one needs to consider factors in addition to TR value. One key factor is the amount of "bootstrapping" provided by the algorithm; others include normalizing the TR-value achieved by breadth of task scope, computing power available, or human assistance.

4.1 INTELLIGENCE AMPLIFICATION

A game player may not care which method was used to generate an excellent Go-playing program, or whether it achieves its results through a clever billion-move opening book and straightforward many-ply search. However, an algorithm designer may value methods using a minimum of prior knowledge more highly, both for conceptual reasons and for their greater likelihood of rapid future advancement and robustness.

How impressively a program performs depends not just on its performance but also on how much prior customization and problem-specific hardwiring went into it. A program that has millions of moves or answers prestored by human hand may be regarded as correspondingly less impressive than one which adapts or evolves some of its strategies, regardless of the program’s TR-value.

Consider checkers as an example. Schaeffer's hand-tuned Chinook won against the human world champion, as recounted in [Schaeffer 1997]. The 2001 Chinook program has perfect information for all board positions having 8 or fewer pieces (that's 443 billion positions). They crafted it using knowledge-intensive tasks like training Chinook's weighting strategy by analyzing 800 grandmaster games and reweighting their evaluation function to get the same result as the grandmaster on each game. "Learning" was done exclusively offline, by human programming or numerical analysis. The programming team collectively put in tens of thousands of hours, and also encoded all 1000+ games of the play of the world's best human checkers player.

In contrast, Fogel's evolved checkers player Anaconda, described in [Chellapilla & Fogel 2000], performed at an expert but not world-class level. However, Anaconda "learned" how to play checkers. It started only with the rules of the game (so that it could make legal moves) and minimax. It used an evolutionary strategy to tune the weights on a neural network, playing 4 plys of minimax. They trained Anaconda on a 400 MHz Pentium II, in around 60 days (or 250 generations), then set it against human players on the web. After 840 additional generations (or 28 weeks), they played it against Hoyle's Classic Games, which includes 3 AIs that play at expert level. Their algorithm played at six plys and won 6 of 6 games.

We would claim that, their performance differential notwithstanding, Anaconda is at least as significant an accomplishment as Chinook. Further detailed discussion of evolved checkers players and the implications of this approach can be found in [Fogel 2001].

To some extent, these opposite approaches represent a task-centered versus a program-centered point of view. In the former, the main goal is to understand computational performance levels in a certain task, with the specific computational methods (and even to some extent the cost or time investment involved) being a secondary concern. Such a point of view may be suitable for assessing trends in performance levels over time, for understanding the automatizability of a task or cognitive domain, or for understanding the state of the art.

In contrast, a program-centered point of view also considers the "intelligence amplification" aspect: how the program achieved its performance level is as important as what that performance level turns out to be. This point of view is appropriate to those trying to understand the relative adaptability of various computational methods.

At one extreme, a program has a full list of answers given to it; at the other extreme it starts with zero domain knowledge, and succeeds in learning good solutions. In between, there are various degrees of "cooking": exhaustive enumeration of cases, hand-designed primitive functions, intermediate reinforcements, externally supplied fitness functions, and so forth. This distinction is analogous to that between rote and deep learning: in the former all knowledge is given by the teacher, while in the latter the teacher suggests insights and poses problems but the student does most of the problem-solving (and hence develops better metastrategies and metaphors which increase general learning power).

With tongue somewhat in cheek, we suggest the terms "cooked AI" and "raw AI" to name these distinctions. Cooked AI has a high degree of problem-specific heuristics, expert knowledge, and fixed representation; raw AI is more readily generalizable, and autonomously learns. Cooked AI solutions may be good for many domains where one does not mind spending the time and money to customize a solution; raw AI solutions (e.g. EC) tend to learn or bootstrap more during their operational phase, and be cheaper computationally and financially - and hence likely more flexible if the domain changes.

How could intelligence amplification be measured? It would be a way of measuring the distance between a program's starting state of knowledge and the results it eventually achieves. One method might be to measure the increase in Turing Ratio as the program adapts to and learns its computational task:

$$\textit{TR(final high-performance state) - TR(start state)}$$

This difference would objectively measure the improvement in performance of the program. Of course, in many programs this iterative improvement occurs via human programmers providing the improvement, so we look specifically for "autonomous increase in TR", i.e. that due to the program itself once completed and running. By this measure, Chinook shows a much smaller increase in TR than Fogel's checkers player.

A more practical definition may be to scale TR by the cost of different approaches. A heavily customized expert system typically requires the combined efforts of many programmers and knowledge engineers working for years to achieve adequate performance. In contrast, EC approaches have often achieved comparable performance with substantially less effort. One might empirically measure:

$$Cost_{Approach} = Cost_{Programming} + Cost_{Processing}$$

We hypothesize that this measure will tend to be lower for methods with higher intelligence amplification.

(Ultimately, one key advantage of generating theories, algorithms, and heuristics is to increase the efficiency with which the slings and arrows of environmental fortune are handled. From this point of view, high intelligence amplification is simply a long-term measure of computational efficiency. However, efficiency differences between low and high amplification algorithms may well be so large as to represent a series of qualitative differences - so the innocent-looking term "efficiency" hides much subtlety.)

Computational depth may suggest formalizations of intelligence amplification. Intuitively, an object is computationally deep if it has a short generating program but that program takes a long time to produce its output. Perhaps the benefits of interaction with a rich environment could be algorithmically described as access to a rich store of precomputed, difficult-to-compute objects which have proven useful. The basic definition of computational depth using Kolmogorov Complexity is in [Antunes et al 2001], but the ramifications of this idea remain to be explored.

Explicitly measuring prior knowledge and intelligence amplification may become a nonissue in the long run, since approaches which rely on massive pre-knowledge or customization for each problem may be increasingly difficult to program for more human-complete problems - and hence the easier-to-program methods will naturally win out over time.

4.2 TASK BREADTH: THE VIRTUES OF GENERALISTS

Many customized programs work well in a narrow domain, but have little or no computational capability elsewhere. Whether this is an issue depends on the user - a task-centered user wanting only a black-box resource to solve subproblem X or an analysis of the level of best current computer performance at the task will care little about what else the black box can and cannot do, while a program-centered user analyzing the intrinsic merit of the heuristics used by the program may care a great deal. (Note though that even a task-centered user may be wary of low-intelligence-amplification programs due to potential brittleness and high knowledge capture costs.)

It would be nice to be able to define a notion of task breadth in an objective way, so that one could speak of e.g. a program's high TR on a narrow-scope task, versus a lower TR on a larger-scope task. One could then compare different methods and heuristics for their breadth-scaled TR. Parameterization of tasks by breadth is also essential in comparing task areas, and perhaps in forming an "ontology of task space".

To use a mathematical metaphor, one scheme for empirically defining task breadth could be to sum:

$$\sum_{s \in Situations} Frequency(s) * Value(s)$$

for those situations where performing well at the task is of use. If a task is more common or more important in the environment of an agent, being able to do the task well is correspondingly more important. (This simple definition is i) meant as a metaphor to build on, ii) defined from the point of view of a single agent - generalizing to a shared view of task breadth depends on getting approximate agreement for average frequency and value measures.) We note that a similar type of definition may be useful in population biology, where species fitness might be approximated by the number of niches where the species performs well, scaled by the importance of each niche to survival.

A definition of task breadth could give an alternative definition of intelligence amplification as "degree of adaptiveness": the breadth of tasks for which the program performs adequately, given initial adequate performance on a limited task. Perhaps an objective and operational measure of intelligence itself could also be derived, as adaptiveness on a sufficiently-diverse environment set.

If we think about measuring the Turing Ratio of human-complete tasks in the near future, values will likely be close to zero for quite a while. Hence we could measure competence on a very restricted subset (which would give reasonably differentiated values), competence on the full set (which would be very low, e.g. the Loebner prize competition as discussed in [Saygin et al 2000]), or finally competence in some series of successively more difficult tasks between what is achievable now and the full task. This last option might be best, but would require a nested series of tasks with successively higher task breadth; one might expect to see a series of logistic curves on the task series, which would need to be chosen at the right difficulty level to be doable given competence at the previous task (similar to teaching a human, and to how solutions for a tough EC task might be evolved in stages).

4.3 DEGREE OF HUMAN ASSISTANCE

We can distinguish three general cases:

1. Isolated algorithm. Once the task is defined and the program complete, the solution process is autonomous. (The border between this category and the next is fuzzy, e.g. human selection of "interesting output" from evolutionary art programs.)
2. Human-algorithm symbiosis. A human is involved in significant parts of the computational process.
3. Isolated human. The "base case" from which our historical and social experience before the 20th century is derived.

We have been comparing the first and third cases; the second may also be of interest for certain applications, in particular those for which

$$TR(algorithm+human) - TR(algorithm)$$

is large. As with intelligence amplification, for some purposes one might not care if superhuman performance is achieved by an autonomous program or by a human-computer symbiosis; for instance, one could picture a Go grandmaster using a Go AI program to explore possible moves, and beating an unassisted grandmaster of equivalent ability. For other purposes such as program analysis or identification of areas of future rapid progress in performance or cost savings, it would be important to maintain the distinction.

5 USAGE AND APPLICATIONS

5.1 GAMES

Games would be an excellent testbed for the Turing Ratio. They are enjoyable and well-specified domains, almost always with clear conditions for winning and losing. Scoring systems have been developed for many games, and a great deal of knowledge and experience (both human and computational) has been gained in the more popular human games. Many AI systems have been developed; see e.g. the first part of [Bouzy 2001].

Examples of games include backgammon, checkers, chess, Go, real-time strategy games, poker, Scrabble, video games, and playable simulations. Combinatorial games are also of interest, as they have an associated mathematical theory that is tied to computational complexity, which may make it possible to parameterize game complexity (see [Nowakowski 1998], [Berlekamp et al 2001], [Nowakowski 2002]).

Usually, games are multiplayer. One-person "games" like combinatorial optimization problems (or even solitaire games) seem less open-ended. In contrast, multiplayer games specify an "interaction protocol" which (for good games) can be played with increasingly higher sophistication by players that keep learning. Hence multiplayer games encourage a coevolutionary increase in strategy sophistication; see [Funes 2001] for an example where Tron agents were evolved using genetic programming in competition with human players.

Games are increasingly being developed which are relevant to competence testing and policy analysis, as practitioners leverage the tremendous interface elegance developed by the game industry over the last couple of decades; [Sawyer 2002] discusses game-based learning and simulation. Perhaps performance in sophisticated educational games and simulations can become a practical method of measuring TR and human cognitive bounds - and of naturally encouraging the development of strategies and expertise to work around these bounds.

Note that handicapping allows meaningful ranking comparison between a human and a program with superhuman performance.

5.2 RESOURCE-SCALED TR

We can use a "resource-bounded Turing Ratio" by bounding computational resources (time, space, etc), and comparing competing algorithms by their rating under the given resource bounds. This factors out improvement due solely to resource differentials.

Complementarily, TR scales with increasing computational power. For tasks which admit a solution whose time is bounded by a low-order polynomial, the size of the problem solvable scales relatively rapidly with resources, and hence so does TR if it is taken to be the size of the problem solved.

However, the graph of TR achieved versus processor power may have a variety of behaviors, the qualitative categories of which form another way of characterizing difficult domains A low average slope would indicate cases where throwing more hardware at the problem is insufficient to significantly increase performance - an example might be PSPACE-hard problems without good probabilistic approximation schemes. Perhaps in such cases EC methods improve slowly with computational power, and must pass through an inherently sequential “critical path” series of inferences as well as spend time interacting with the real world in order to come up with creative solutions.

Once there exist algorithms with sufficiently high TR for a domain, there may still be a human in the loop for part of the task (including pre or post processing). We can then divide the total time of the symbiotic human-computer team *SymbTime* into human and computer portions, and consider the long-term distribution of these portions. The continuing progression of Moore's Law implies that the cost for a given level of TR performance will decay exponentially for tasks whose improvement scales fast enough with increasing computational power, at least for as long as Moore's Law continues to hold. This in turn suggests the following definitions and ratios:

- $SymbTime = SymbTime_{Comp} + SymbTime_{Human}$
- $HumanTime$ = unaided human time for task
- $HumanRatio = SymbTime_{Human} / HumanTime$
- $CompRatio = SymbTime_{Comp} / HumanTime$

$SymbTime_{Comp}$ accounts for the portion which has been algorithmized, while $SymbTime_{Human}$ is the portion needing humans to perform adequately. *HumanRatio* then represents the limiting reduced time that can be achieved as a fraction of the time required by an unaided human. This is so since the Moore's Law assumption implies

$$Time_{Comp}\ (now+T) = 2^{-cT} * Time_{Comp}\ (now)$$

so *CompRatio* will decrease exponentially. (Although poor user interfaces could result in a *HumanRatio* greater than 1, the cases of greatest interest are cognitively complex tasks with *HumanRatio* close to 0, since they are the tasks which can be usefully outsourced from humans to computers.)

Note that this result also holds for other exponentially-increasing computational properties with which a problem's TR scales well, such as memory. Although exponential growth of a computational property is a temporary phenomenon in a bounded universe, the fact that we are in the middle of several such temporary regimes makes the analysis relevant to the early 21st century. [Hanson JAIR] discusses further consequences of machine intelligence for economic growth.

Ranking domains by *HumanRatio* - and graphing values for domains over time - will give empirical evidence as to the cognitive tasks for which human thought is most suited and required, and the tasks best done by computers. Similarly, one could potentially objectively identify heuristics or approaches that caused large decreases in this ratio for particular domains, and perhaps see some "meta-patterns" in the types of approaches that give the most leverage.

5.3 TIME SERIES

Time series of ratings will give tools for visual and trend analysis. The statistical significance of a Turing Ratio result for a particular domain (e.g. the variance or stationarity of a particular TR rating) may be an issue in some cases; empirically, however, humans do still get ranked in many domains. For many tasks, improvements in TR will probably come in large discrete jumps (e.g. when a clever data structure reduces the algorithm's time complexity by an order of magnitude). We hypothesize that tasks which are more open-ended and creative will have more and smaller jumps, leading perhaps to more predictability in the rate of improvement of such problems (via the Central Limit Theorem).

Similarly, it may be possible to derive taxonomies of domains by the current Turing Ratio. Looking at relative computational strengths between different types of cognitive tasks in some sense tells us how well we understand those tasks, since being able to build an algorithm to perform a task is a strong form of knowledge. As Richard Feynman said: "What I cannot build, I do not understand."

6 DISCUSSION AND CONCLUSIONS

6.1 POTENTIAL OBJECTIONS AND CONTRARY VIEWS

Objection: The point of the Turing test is that the QUALITY of intelligence is sufficiently non-quantitative to require an individual with that quality to judge its presence. Trying to evaluate it numerically is simply missing the point. One might be able to have a creature with limited intelligence evaluate similarly limited intelligence in a Turing test, but one can't simply reduce the test to numbers.

There are two separate points here. One, does the measurement of intelligence require an intelligent tester? Two, are quantitative measurements of such a test of any use?

To the first point, we would say that the jury is still out. Turing's original test does indeed require an intelligent tester; [Dennett 1998] makes several good points in this regard, with respect to both the discrimination power of Turing's conversation task and methods of fooling an unsophisticated tester.

Do all tasks measuring the presence of human-level qualities require an intelligent tester? Perhaps not, especially in the important case where the task itself includes comparisons or competitions with other intelligent competitors. In this case, as long as the task each competitor must perform both admits arbitrarily complex strategies and has well-defined success measures that belong to some tractable computational complexity class, there is no reason in principle why an algorithm cannot judge which participant created the better outcome, and rank them accordingly.

Consider an analogy to the computational complexity class NP - this class contains problems which probably need a large amount of work to do properly, but which have an easy verification process once the hard work of finding the solution has been done. Similarly, a computational agent may play the part of a "verifier of performance", while being unable to itself achieve such performance.

To the second point, the answer is a definite yes. Performance outcomes can indeed be reduced to numbers, or to some not much more complex data structure. Humans do this all the time for many real-world domains: school marks, credit ratings, game scores, and so on. The real trick is in defining a task that is sufficiently general to measure the property that one is really interested in, and yet sufficiently tractable to be measurable given practical resource constraints.

Objection: If you start measuring Turing Ratios for every interesting task that comes along, soon you have a plethora of measurements, each of which is domain-specific to varying degrees. What's the use of having so many distinct measurements, each of which requires some interpretation and domain knowledge to be meaningful to a user?

We don't claim that Turing Ratios end the need for domain-specific understanding to accurately gauge performance - in fact, we strongly emphasize that such understanding is necessary for a nuanced view of what a performance measurement means. However, to the extent that performance measurements across different domains have common features, we argue that standardizing these common features will aid comparison and analysis. As well, the act of putting various measurements into a coherent framework may help create a series of increasingly more powerful performance measurements, forming an open-ended pathway to more widely competent algorithms.

The two issues of creating a descriptive ontology for the set of tasks for which Turing Ratios have been taken, and of choosing a small representative subset that could act as a "Turing Test Suite" measuring the various facets of mind we uncover, are both important open problems.

Task performance is usually not a single-dimensional quantity, as for example with intelligence. In fact, task performance may even be observer-dependent, e.g. composing a piece of music. So how can one claim that a Turing Ratio can be well-defined?

It is true that many or even most open-ended tasks that come to mind have outcomes that are difficult to summarize adequately in a single scalar value. But empirically we can often capture a significant component of performance in such a value, whether the value represents a single conceptual quantity directly or a transformed function of several values. Marks are an example which are highly used in practice.

As for observer dependence of rankings, the same issue applies to human performance in such domains as music. We can, however, usually efficiently separate those who are talented from those who are less so. (Note that to the extent to which observer preferences can be encoded in the task description, the problem goes away.)

You are really discussing a task ratio, not a Turing ratio. Turing asked, "under what conditions can a machine be said to think?" In theory, the idea of evaluating the ability of computers vs each other and humans will result in an empirical set of performance measures in your so-called "ontology of task space", which might then give a derived measure of computer vs human thought in general. But in practice, we don't have the foggiest idea of how to parse this ontology of task space to decide whether some entity can said to be a thinking being. Surely not all human-level performance in this task space requires attributing intelligence to the entity. So which items in this task space are relevant and to what degree?

The objection asks for sufficiency conditions in terms of our ontology of task space, for attribution of general intelligence. The answer is threefold. First, there is a substantial (but by no means complete) degree of support for the idea that some tasks are sufficient in and of themselves to certify human-level intelligence - such as the original Turing Test, or perhaps some of the later variants that deal with various objections to the original Test. If such tasks do in fact exist, then our framework provides a means to generate a task ratio on them, and infer a performance score from multiple (possibly competition-based) measurements.

Second, we agree that general human-equivalent intelligence should also be capable of being inferred from a collection of task space measurements, none of which would individually qualify. We do not have a specific algorithm for how this should be done - it will require a great deal of experimental work (and intellectual argument) to establish. But note that even the original Turing Test implicitly includes testing many different individual task metrics - a sophisticated questioner (required to see past the computational equivalent of smoke and mirrors that might be able to pass a routine or known set of questions) would likely test many different aspects of intelligence, each of which might also be capable of being tested separately. So perhaps the original Turing Test answers the question of "which items in this task space are relevant and to what degree" by leaving them to the intelligent human questioner to implicitly decide, while a Turing Ratio derived from individual task performance measurements would require these assessments to be made explicitly.

Third, we do not wish to unduly restrict notions of "intelligence" to preconceived notions. It may be that sufficient task performance on tasks which would not be sufficient to certify general human-equivalent intelligence is still a valid measure of some useful but more restricted form of intelligence.

While it is clear that one can devise very context-specific measures of performance, these simply wouldn't be interesting unless we believed that performance in one context was predictive of performances in other contexts. To be relevant to the Turing Test topic, it isn't enough to simply list more and more arguments we could add to the test function, or more and more context we could take into account when making comparisons. The key issue is how much and what sorts of correlations to expect between various test scores, so that we can design tests which can see those sorts of correlations if they are there. Are you simply claiming that this is an empirical issue, which we won't know until we get more data?

There are in fact many cases where particular task performance is of great interest. This is particularly the case for tasks which have previously been thought to require "intelligence", but which are then empirically shown to be solvable by algorithms which clearly do not merit the label in its broad sense. Chess is an obvious example, but there are numerous others. And there are likely many more tasks which will come into this category in coming years, those close to what we call the "Turing Frontier" - these might include automated translation, music composition, and driving in busy urban environments, and then again might not. It is of great theoretical and practical interest to track performance in such areas over time, recognizing that they are insufficient to certify human-level intelligence but acknowledging that they do certify some non-trivial degree of "intelligence" to many people - and perhaps also certify algorithmic solutions of economically-useful tasks.

In the case of finding correlations between test scores, we do claim that this is an issue requiring a great deal of empirical study and further research - see also the answer to the previous question. Our notion of "task breadth" is only a modest beginning, and we look forward to more sophisticated notions being developed. The psychological literature on intelligence measurement and types of intelligence would likely provide a basis from which to begin. Note, though, that the idea of specific correlations between task scores may not be well-defined - it presupposes that an "intelligence" (or entity capable of passing a Turing Test) would necessarily have a particular pattern of such correlations, when in fact there may be many varying structures of task performance correlations, corresponding to different kinds of minds.

So far this is all just talk. How could one calculate some actual TR numbers?

We have presented a philosophical framework, some discussion of its implications, and suggestions for practical algorithms. The mathematics of inferring talent levels from pairwise comparisons that we have described above is an eminently practical method of calculating relative TR values - one that is already used in many competitive domains. For absolute TR values, the measurement task is much easier, since by definition we have an algorithm for measuring absolute talent levels and do not have to take ability distributions and changing competitive environments into account; in this case, the problem is to define computational tasks which adequately capture abilities of interest.

In order to translate the idea into hard numbers in a specific domain, two key ingredients are necessary: a domain-specific analysis to determine the exact form that a TR measurement should take, and a series of trusted and reliable measurements. In this sense, it is not a "cookbook procedure" - which accords with intuition, since we know how hard it is to judge good performance in an unfamiliar field (and often rely on environmental cues like accolades, self-confidence, etc). Deriving reliable and useful TR data will require insight and systematic experimentation.

6.2 CONCLUSIONS

We have demonstrated the formalization of algorithmic performance through the Turing Ratio. A mathematical base for the Turing Ratio was explained using notions from statistics and the mathematics of inducing rankings from pairwise comparisons, and conceptually validated through empirical examination of chess program performance. Along with the companion notions of intelligence amplification and task breadth, the Turing Ratio forms a conceptual foundation for measuring long-term progress in machine intelligence, up to and beyond human performance levels.

As [Bentley & Corne 2002] vividly show, evolutionary computation techniques are already producing many results that deserve the label “creative”. To the degree that progress in open-ended and creative tasks can be meaningfully quantified, increasing numbers of empirical performance results will become available for analysis.

Three key challenges now are to further develop the theory, identify suitable applications, and begin to measure Turing Ratio values systematically. It is true that the Turing Ratio as we have defined it is applicable only to the subset of tasks satisfying our assumptions. However, we suggest that this subset consists of tasks which, once identified and explored, may serve as windows into creative and open-ended algorithmic performance.

Acknowledgments

Alan Turing inspired generations of computer scientists with his Turing Test and Turing Machine concepts. We hope he would have appreciated our attempt to extend his ideas.

For their helpful feedback and discussion, we thank Ben Houston, Alexander Tulai, Andrew Vardy, several attendees and anonymous referees at GECCO-2002, and anonymous referees from the *Journal of Evolution and Technology*.

References

[Anthony 1980] Piers Anthony. *Split Infinity.* Ballantine Books, 1980.

[Antunes et al 2001] L Antunes, L Fortnow, and D van Melkebeek. Computational Depth. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, pp 266-273; IEEE, 2001.

[Bentley & Corne 2002] Peter J Bentley and David Corne. *Creative Evolutionary Systems*. Academic Press, 2002.

[Berlekamp et al 2001] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning Ways for your Mathematical Plays (2nd edition, Vol 1).* A K Peters Ltd, 2001.

[Bouzy 2001] Bruno Bouzy. Computer Go: An AI-Oriented Survey. In *Artificial Intelligence* (Vol.132 No.1), pp 39-103.

[Chellapilla & Fogel 2000] Kumar Chellapilla and David B Fogel. Anaconda Defeats Hoyle 6-0: A Case Study Competing an Evolved Checkers Program against Commercially Available Software. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pp 857-863; IEEE Press, 2000.

[Dennett 1998] Daniel C Dennett. Can Machines Think? In *Brainchildren: Essays on Designing Minds,* pp 3-29; MIT Press, 1998.

[FIDE 2000] Fédération Internationale d'Échecs (FIDE). "The Working of the FIDE rating system". In *2000 FIDE Handbook*, section B, 02.10.1a; viewed at http://handbook.fide.com on 14 March 2002.

[Fogel 2001] David B Fogel. *Blondie24: Playing at the Edge of AI.* Morgan Kaufmann, 2001.

[French 2000] Robert M French. The Turing Test: the first fifty years. In *Trends in Cognitive Sciences* (Vol.4 No.3), pp 115-121.

[Funes 2001] Pablo Funes. *Evolution of Complexity in Real-World Domains*. PhD Thesis, Brandeis University, 2001.

[Glickman 1999] Mark Glickman. Parameter estimation in large dynamic paired comparison experiments. In *Applied Statistics* (Vol.48), pp 377-394.

[Grottling 1998] Göran Grottling. *Frequently Asked Questions about the SSDF Rating List.* 1998; viewed at http://home.swipnet.se/~w-36794/ssdf/ssdf-faq.htm on 18 January 2002.

[Hanson JAIR] Robin Hanson. Economic Growth Given Machine Intelligence. To appear in *Journal of Artificial Intelligence Research*.

[Jain 1991] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[Koza 1999] John R Koza. *Genetic Programming III*. Morgan Kaufmann, 1999.

[Masum et al 2002] Hassan Masum, Steffen Christensen, and Franz Oppacher. The Turing Ratio: Metrics for Open-Ended Tasks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*; Morgan Kaufmann, 2002.

[Nowakowski 1998] Richard J Nowakowski (Editor). *Games of No Chance.* Cambridge University Press, 1998.

[Nowakowski 2002] Richard J Nowakowski (Editor). *More Games of No Chance.* Cambridge University Press, 2002.

[Sawyer 2002] Ben Sawyer. Serious Games: Improving Public Policy through Game-Based

Learning and Simulation. Foresight and Governance Project Publication 2002-1, Woodrow Wilson International Center for Scholars; 2002.

[Saygin et al 2000] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. Turing Test: 50 Years Later. In *Minds and Machines (*Vol.10 No.4), pp 463-518.

[Schaeffer 1997] Jonathan Schaeffer. *One Jump Ahead*. Springer-Verlag, 1997.

[Smith 1993] Warren D Smith. Rating Systems for Gameplayers, and Learning. NEC Technical Report 93-104-3-0058-5, 1993.

[Turing 1950] Alan M Turing. Computing Machinery and Intelligence. In *Mind* (Vol.59 No.236), pp 433-460.

[USCF 2002] Mark Glickman. *The United States Chess Federation Rating System*. 2001; viewed at http://math.bu.edu/people/mg/ratings/rs/rs2.html on 14 March 2002.