A Dynamic Intelligence Test Framework for Evaluating AI Agents

Nader Chmait and **Yuan-Fang Li** and **David L. Dowe** and **David G. Green**[1]

Abstract. In our recent work on the measurement of (collective) intelligence, we used a dynamic intelligence test to measure and compare the performances of artificial agents. In this paper we give a detailed technical description of the testing framework, its design and implementation, showing how it can be used to quantitatively evaluate general purpose, single- and multi-agent artificial intelligence (AI). The source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to artificial agents have been outlined. This will allow evaluating new agent behaviours and also extending the scope of the test. Alternative testing environments are discussed along with other considerations relevant to the robustness of multi-agent performance tests. The intuition is to encourage people in the AI community to quantitatively evaluate new types of heuristics and algorithms individually and collectively using different communication and interaction protocols, and thus pave the way towards a rigorous, formal and unified testing framework for general purpose agents.

1 INTRODUCTION

One of the major research questions at the present state of artificial intelligence is: how smart groups of artificial agents are compared to individual agents? Measuring machine intelligence is a complex topic that has been tackled by a large number of theories and is not yet fully understood as discussed in [24], [22, Section 2] and [21, Chapter 5]. Besides that, the design and study of agent systems has widely expanded in the last few years as agent models are increasingly being applied in a wide range of disciplines in the purpose of modelling complex phenomena.

In our previous work on the measurement of collective intelligence [9, 7, 8] we identified a range of factors that hinder the effectiveness of individual and interactive AI agents. This was achieved by implementing a dynamic intelligence test based on the literature of artificial general intelligence and algorithmic information theory [23, 22, 25]. We have used it to measure and compare the performance of individual, and collectives of, artificial agents across several environmental and interactive settings.

In this paper we give detailed technical description of our dynamic intelligence testing framework. We discuss its design and implementation and show how it can be used to quantitatively evaluate general purpose AI individually, and also collectively using various interaction protocols. As anticipated in our recent work [9], the source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to artificial agents have been outlined. Consequently, it is now possible to evaluate new agent behaviours, and easily extend the scope of the evaluation framework. The intuition is to encourage people in the AI community to evaluate new types of heuristics and algorithms in individual and collective scenarios using different communication and interaction protocols. This will hopefully pave the way towards a rigorous, formal and unified testing framework for general purpose artificial agents.

[1] Faculty of IT, Clayton, Monash University, Australia, email: {nader.chmait,yuanfang.li,david.dowe,david.green}@monash.edu

2 BACKGROUND

Perhaps a good start to understand the history of machine intelligence would be to take a look back at the imitation game [43] proposed by Turing in the 1950s where the idea is to have one or more human judges interrogating a program through an interface, and the program is considered intelligent if it is able to fool the judges into thinking that they are interrogating a human being. While this was once regarded as an intelligence test for machines, it has limitations [34] and is mainly a test of humanness. The machine intelligence quotient (MIQ) using fuzzy integrals was presented in [1] in 2002. However, determining a universal *intelligence quotient* for ranking artificial systems is not very practical and is almost unmeasurable due to the vast non-uniformity in the performances of different types of artificial systems. Several studies [5, 11, 12, 14, 25, 36] have investigated the relevance of compression [11, Sections 2 and 4], pattern recognition, and inductive inference [13, Section 2] to intelligence. Shortly after [5, 11, 12, 25] came the C-test [19] which was one of the first attempts to design an intelligence test consisting of tasks of quantifiable complexities. However, the test was static (non-dynamic) and it did not fully embrace the vast scope implicit in the notion of intelligence. In 2007, Legg and Hutter proposed a general definition of *universal (machine) intelligence* [30], and three years later a test influenced by this new definition, namely the *Anytime Universal Intelligence Test*, was put forward by Hernandez-Orallo and Dowe [22] in order to evaluate intelligence. The test was designed to be administered to different types of cognitive systems (human, animal, artificial), and examples environment classes illustrating the features of the test were also suggested in [22].

To the best of our knowledge, further to single agent intelligence [1, 19, 30, 22], no *formal intelligence tests* were developed in the purpose of *quantifying the intelligence of groups of interactive agents* against isolated (non-interactive) agents - which is one of the motivations behind this work. Yet, before we proceed with the description of our work, one question that might come to a reader's mind is: can't we simply evaluate and compare artificial systems over any given problem or environment from the literature? There are several reasons why we can't do that, most of them were studied and examined in [22]. We briefly summarise some of these principles. Firstly, there is a risk that the choice of the environment used

for evaluation is biased, and that it favours particular types of agents while it is unsuitable for others. Furthermore, the environment should handle any level of intelligence in the sense that dull or brilliant, and slow or fast systems can all be adequately evaluated. The test should return a score after being stopped at any time-period, short or long. Besides, not every evaluation metric is a formal intelligence test or even at a minimum, a reliable performance metric. For instance, the testing environment should be non-ergodic but reward-sensitive with no sequence of actions leading to heaven (always positive) or hell (always negative) scoring situations, and balanced [20] in the sense that it must return a null reward to agents with a random behaviour, etc.

Although the principle advantage of this work is measuring the intelligence of artificial agents, the outcome also has implications for agent-based systems. This is because it provides an opportunity to predict the effectiveness (and expected performance) of existing artificial systems under different collaboration scenarios and problem complexities. In other words, it's one way of looking at (quantifying) the emergence of intelligence in multi-agent systems.

We begin by introducing our methodology for evaluating intelligence using an agent-environment architecture (Section 3). We then re-introduce and elaborate on the Λ^* (Lambda Star) testing environment structure described in [9] (Section 4). We discuss the test implementation, its setup and parameters, in Section 5 and also give examples of how to define and evaluate new agent behaviours over the proposed testing environment. We then discuss in Section 6 some alternative testing environments that might be useful to quantify the performance of artificial agents and raise some arguments and considerations relevant to the robustness of multi-agent performance tests. We conclude in Section 7 with a brief summary.

3 AGENT-ENVIRONMENT FRAMEWORK

A common setting in most approaches to measuring intelligence is to evaluate a subject over a series of problems of different complexities and return a quantitative measure or score reflecting the subject's performance over these problems [22]. In artificial systems, the agent-environment framework [30] is an appropriate representation for this matter. For instance, this framework allows us to model and abstract any type of interactions between agents and environments. It also embraces the *embodiment thesis* [2] by embedding the agents in a flow of observations and events generated by the environment.

Here we define an environment to mean the world where an agent π, or a group of agents $\{\pi_1, \pi_2, \ldots, \pi_n\}$, can interact using a set of observations, actions and rewards [30]. The environment generates observations from the set of observations $\mathcal{O}$, and rewards from $\mathcal{R} \subseteq \mathbb{Q}$, and sends them to all the agents. Then, each agent performs actions from a limited set of actions $\mathcal{A}$ in response. An iteration or step i stands for one sequence of observation-action-reward. An observation at iteration i is denoted by o_i, while the corresponding action and reward for the same iteration are denoted by a_i and r_i respectively. The string $o_1a_1r_1o_2a_2r_2$ is an example sequence of interactions over two iterations between one agent and its environment. An illustration of the agent-environment framework is given in Figure 1. We define the multi-agent-environment framework as an extension of the above, such that $o_{i,j}, a_{i,j}$ and $r_{i,j}$ are respectively the observation, action and reward for agent π_j at iteration i. The order of interactions starts by the environment sending observations to all the agents at the same time. Then, the agents interact and perform corresponding actions, and finally the environment provides them back with rewards. For instance, the first interaction of agents

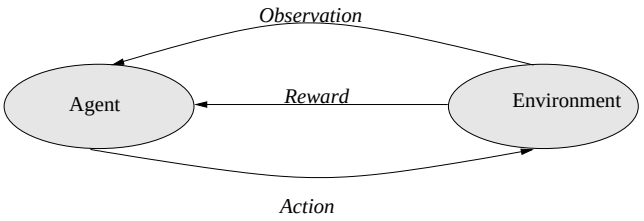

Figure 1: Agent-environment framework [30]

π_1, π_2 in the multi-agent-environment setting, denoted by $o_1a_1r_1$, is equivalent to $o_{1,1}o_{1,2}a_{1,1}a_{1,2}r_{1,1}r_{1,2}$.

4 EVALUATING INTELLIGENCE

In order to assess the performances of AI agents, whether in isolation or collectively, we needed an environment over which we can run formal intelligence tests (of measurable complexities) on artificial agents using the recently described framework. Hence, we developed in our recent work [9] an extension of the Λ environment [28, Sec. 6][22] - one of the environment classes implementing the theory behind the "Anytime Universal Intelligence Test" [22].

One of the reasons for selecting the Anytime Universal Intelligence Test and the Λ environment was because they are derived from formal and mathematical backgrounds that have been practically used to evaluate diverse kinds (including machines) of entities [26, 7, 8, 27]. More importantly, our selection embraces all of the concerns we raised in the introduction regarding the measurement of intelligence, thus providing us with a formal, anytime, dynamic and unbiased setting [22] to quantitatively evaluate the effectiveness of artificial agents.

4.1 The Λ^* (Lambda Star) Environment

We re-introduce the Λ^* (Lambda Star) environment class used in [9] which is an extension of the Λ environment [22, Sec. 6.3][28] that focuses on a restricted - but important - set of tasks in AI.

The general idea is to evaluate an agent that can perform a set of actions, by placing it in a grid of cells with two special objects, *Good* ($\oplus$) and *Evil* ($\ominus$), travelling in the space using movement patterns of measurable complexities. The rewards are defined as a function of the position of the evaluated agent with respect to the positions of $\oplus$ and $\ominus$.

4.1.1 Structure of the test

An environment space is an m-by-n grid-world populated with objects from $\Omega = \{\pi_1, \pi_2, \ldots, \pi_x, \oplus, \ominus\}$, the finite set of objects. The set of evaluated agents $\Pi \subseteq \Omega$ is $\{\pi_1, \pi_2, \ldots, \pi_x\}$. Each element in Ω can have actions from a finite set of actions $\mathcal{A}$ ={*up-left, up, up-right, left, stay, right, down-left, down, down-right*}. All objects can share the same cell at the same time except for $\oplus$ and $\ominus$ where in this case, one of them is randomly chosen to move to the intended cell while the other one keeps its old position. In the context of the agent-environment framework [30], a test episode consisting of a series of ϑ interactions $o_ia_ir_i$ such that $1 \leq i \leq \vartheta$, is modelled as follows:

1. the environment space is first initialised to an m-by-n toroidal grid-world, and populated with a subset of evaluated agents from $\Pi \subseteq \Omega$, and the two special objects $\oplus$ and $\ominus$,

2. the environment sends to each agent a description of its range of 1 *Moore neighbour* cells [17, 48] and their contents, the rewards in these cells, as an observation,
3. the agents (communicate/interact and) respond to the observations by performing an action in $\mathcal{A}$, and the special objects perform the next action in their movement pattern,
4. the environment then returns a reward to each evaluated agent based on its position (distance) with respect to the locations of the special objects,
5. this process is repeated again from point #2 until a test episode is completed, that is when $i = \vartheta$.

The Λ^* environment consists of a toroidal grid space in the sense that moving off one border makes an agent appear on the opposite one. Consequently, the distance between two agents is calculated using the surpassing rule (toroidal distance) such that, in a 5-by-5 grid space for example, the distance between cell $(1, 3)$ and $(5, 3)$ is equal to 1 cell. An illustration of the Λ^* environment is given in Figure 2.

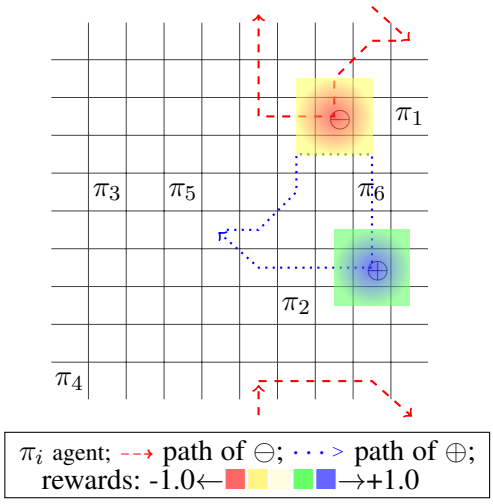

Figure 2: A diagrammatic representation of a sample 10-by-10 Λ^* environment space used in [9] to evaluate the performance of (groups of interactive) artificial agents. The figure shows the objects in Ω, the paths of the two special objects and an illustration of the (positive and negative) rewards in the environment.

4.1.2 Rewarding function

The environment sends a reward to each evaluated agent from the set of rewards $\mathcal{R} \subseteq \mathbb{Q}$ where $-1.0 \leq \mathcal{R} \leq 1.0$. Given an agent π_j, its reward $r_j^i \in \mathcal{R}$ at some test iteration i can be calculated as:

$$r_j^i \leftarrow \frac{1}{d(\pi_j, \oplus) + 1} - \frac{1}{d(\pi_j, \ominus) + 1}$$

where $d(a, b)$ denotes the (toroidal) distance between two objects a and b. Recall that an agent does not have a full representation of the space and only receives observations of its (range of 1 Moore) neighbourhood [17, 48]. Therefore, we constrain the (positive and negative) rewards an agent receives from the environment (as a function of its position with respect to $\oplus$ and $\ominus$ respectively) as follows: the positive reward π_j receives at each iteration is calculated as $1/(d(\pi_j, \oplus) + 1)$ if $d(\pi_j, \oplus) < 2$, or 0 otherwise. Likewise its negative reward at that iterations is $-1/(d(\pi_j, \ominus) + 1)$ if $d(\pi_j, \ominus) < 2$, or 0 otherwise. Its total reward, r_j^i at iteration i, is the sum of its positive and negative rewards received at that iteration.

4.2 Algorithmic Complexity

We regard the Kolmogorov complexity [32][2] of the movement patterns of the special objects as a measure of the algorithmic complexity $K(\mu)$ of the environment μ in which they operate. For instance, a Λ^* environment of high Kolmogorov complexity is sufficiently rich and structured to generate complicated (special object) patterns/sequences of seeming randomness.

The Kolmogorov complexity [32] (Definition 1) of a string x is the length of the shortest program p that outputs x over a reference (Turing) machine U.

Definition 1 Kolmogorov Complexity

$$K_U(x) := \min_{p:\, U(p)=x} l(p)$$

where $l(p)$ denotes the length of p in bits, and $U(p)$ denotes the result of executing p on a Universal Turing machine U.

Assume, for example, that the special object $\oplus$ moves in a 5-by-5 grid space. It has an ordered (and repeating) movement pattern travelling between cells with indices: $7, 3, 4, 9$ and 8 (grayed cells appearing in Figure 3) such that, in a 25-cell grid, indices $1, 2$ and 6 correspond respectively to cells with coordinates $(1, 1)$, $(1, 2)$ and $(2, 1)$ and so on (Figure 3). Also assume that the number of time steps in one test episode $\vartheta = 20$. Following algorithmic information theory, namely Kolmogorov complexity, we consider the algorithmic complexity of the environment $K(\mu)$ in which $\oplus$ operates as the length of the shortest program that outputs the sequence 73498734987349873498 (of length ϑ). We measure the Lempel-Ziv complexity [31] of the movement patterns as an approximation to $K(\mu)$ as suggested in [31, 15].

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 3: A conceptual representation of a 5-by-5 grid space and its cell indices ranging from 1 to 25.

Note that, at one test episode, the movement patterns of the special objects $\oplus$ and $\ominus$ are different but (algorithmically) equally complex making sure the rewards are balanced [20]. The recurrent segment of the movement pattern is at least of length one and at most $\lfloor \vartheta/2 \rfloor$, cyclically repeated until the final iteration (ϑ) of the test.

4.3 Search Space Complexity

We measure the search space complexity $\mathcal{H}(\mu)$ as the amount of *uncertainty* in μ, expressed by Shannon's entropy [38]. Let N be

[2] The concept of Kolmogorov complexity or algorithmic information theory (AIT) is based on independent work of R. J. Solomonoff [39, 40, 41] and A. N. Kolmogorov [29] in the first half of the 1960s, shortly followed by related work by G. J. Chaitin [3, 4]. The relationship between this work and the Minimum Message Length (MML) principle (also from the 1960s) [45] is discussed in [46][44, Chapter 2 and Section 10.1][10, Sections 2 and 6].

the set of all possible states of an environment μ such that a state s_μ, is the set holding the current positions of the special objects $\{\oplus, \ominus\}$ in the m-by-n space. Thus the number of states $|N|$ increases with the increase in the space dimensions m and n, and it is equal to the number of permutation ${}^{\mathrm{m}\times\mathrm{n}}P_2 = \frac{(\mathrm{m}\times\mathrm{n})!}{(\mathrm{m}\times\mathrm{n}-2)!}$. The entropy is maximal at the beginning of the test as, from an agent's perspective, there is complete uncertainty about the current state of μ. Therefore $p(s_\mu)$ follows a uniform distribution and is equal to $1/|N|$. Using $\log_2$ as a base for our calculations, we end up with: $\mathcal{H}(\mu) = -\sum_{s_\mu \in N} p(s_\mu) \log_2 p(s_\mu) = \log_2 |N|$ bits.

Algorithmic and search space complexities could be combined into a higher level complexity measure of the whole environment. This new measure can be very useful to weight test environments used for the measurement of universal intelligence. Nonetheless, having two separate measures of complexity also means that we can quantify the individual influence of each class (or type) of complexity on the performance of agents. This approach appears to be particularly useful for evaluating the factors influencing the performance of agent collectives as these collectives can exhibit different behaviors in response to changes in the measures of each class of environment complexity [9, Section 7].

Overall, we appraise the Λ^* environment, at a minimum, as an accurate measure of the subject's ability of performing over a class of: inductive-inference, compression, and search problems, all of which are particularly related to intelligence [25, 14]. Note, however, that we will use the term *intelligence* to describe the effectiveness or accuracy of an evaluated agent over this test.

4.4 Intelligence Score

The metric of (individual agent) universal intelligence defined in [22, Definition 10] was extended into a collective intelligence metric (Definition 3) returning an average reward accumulation per-agent measure of success (Definition 2) for a group of agents Π, over a selection of Λ^* environments (Section 4.1).

Definition 2 *Given a Λ^* environment μ and a set of (isolated or interactive) agents $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ to be evaluated, the (average per-agent per-iteration) reward $\tilde{R}_{\Pi,\mu,\vartheta}$ of Π over one test episode of ϑ-iterations is calculated as: $\tilde{R}_{\Pi,\mu,\vartheta} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{\vartheta} r_j^i}{n \times \vartheta}$.*

Definition 3 *The (collective) intelligence of a set of agents Π is defined as: $\Upsilon(\Pi) = \frac{1}{\omega} \sum_{\mu \in L} \tilde{R}_{\Pi,\mu,\vartheta}$, where L is a set of ω environments $\{\mu_1, \mu_2, \ldots, \mu_\omega\}$ such that $\forall \mu_t, \mu_q \in L$: $\mathcal{H}(\mu_t) = \mathcal{H}(\mu_q)$, and $\forall \mu_i \in L, K(\mu_i)$ is extracted from a range of (pattern) algorithmic complexities in $]1, K_{max}]$.*

Note the use of the same search space complexity, but different algorithmic complexities, in the intelligence measure defined in Definition 3. The reason behind this is to allow for running controlled experiments to test against the influence each class of complexity has on intelligence separately in a similar manner to [9, Sections 7.3 and 7.6].

5 IMPLEMENTATION DETAILS AND EXPERIMENTAL PROTOCOL

In this section we discuss some important test functionalities and experimental parameters, and give technical description of example agent behaviours showing how they can be practically evaluated over the Λ^* environment.

5.1 Setup and Test Parameters

The intelligence test was implemented in C++, and the source code and scripts to run experiments have been released as open-source [6], with good efforts made to facilitate their re-usability.

Once the test is compiled and run, a new experiment is initiated. The number of test episodes ω, as well as the number of iterations in each episode, for that experiment can be entered into the command-line. Setting ω to 1000 episodes (runs) usually records a very small standard deviation between the test scores[3]. The size of the environment (and thus the search space uncertainty $\mathcal{H}(\mu)$) as well as the number of agents to be evaluated can also be selected prior to each experiment. The robustness of the test scores depends on the size of the environment so it might be desirable to select a larger value of ω for larger environment spaces.

In each episode, agents are administered over different pattern complexities $K(\mu)$ automatically generated by the test, such that $K(\mu) \in [2, 23]$, where a $K(\mu)$ of 23 corresponds to, more or less, complex pattern prediction or recognition problems. Moreover, in each episode, the evaluated agents are automatically re-initialised to different spatial positions in the environment. At the end of each experiment the (intelligence) scores (in the range $[-1.0, 1.0]$) of the evaluated agents and collectives, averaged over all test episodes, are displayed on the screen and also saved to file.

Agents can be evaluated in isolation as well as collectively following the agent environment framework described in Section 3. For instance, the test provides us with three key methods implementing the (multi) agent-environment framework . Let $\tilde{\mu}$ be an instance of the test environment Λ^* and Π a set of agents to be evaluated. The methods $sendObservations(\Pi, k)$ and $sendReward(\Pi, i)$ could be invoked on $\tilde{\mu}$ at each iteration i of the test in order to send observations and rewards respectively to all agents in Π, where $k \in \mathbb{N}^+$ refers to the k^{th}-Moore neighbourhood selected as the evaluated agents' observation range. At each iteration of the test, after receiving an observation, each agent in Π invokes its own method $performAction()$ which returns a discrete action in the range $[1, 9]$, such that an action in $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ maps position-wise to {*up-left, up, up-right, left, stay, right, down-left, down, down-right*}. The selected action is subsequently used to update the agent's position in the environment.

5.2 Defining Agent Behaviours

We have defined an abstract class *Agent* with many declared functionalities that will come out to be essential for implementing and evaluating new agent behaviours over the Λ^* environment.

New isolated (non-interactive) agent behaviours can be introduced as (one of the) subclasses of *Agent*, providing implementations for its abstract methods as necessary. Interactive agent behaviours, on the other hand, are polymorphic classes redefining and extending the behaviour of their isolated agent's superclass.

Homogeneous collectives of interactive agents are aggregations of two or more interactive agents of the same behaviour (class). A simplified UML class diagram illustrating the relationships between isolated and collective agent behaviours is illustrated in Figure 4. Likewise, heterogeneous collectives of agents can be defined as aggregations of two or more interactive agents of different behaviours (classes). Examples of (isolated and collective) agent behaviours are described in the next two subsections.

[3] Usually a standard deviation of less than 0.001 is recorded between identical experiments.

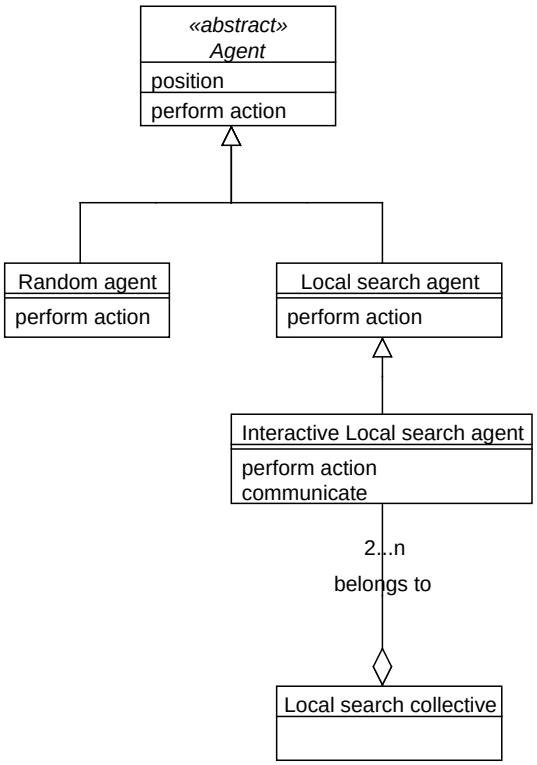

Figure 4: A simplified UML class diagram illustrating the relationships between some isolated and collective agent behaviours.

5.2.1 Isolated agent behaviours

In this subsection, we give a description of some agent behaviours[4] which could be evaluated over the Λ^* environment.

Local search agent: given an agent π_j, we denote by c_j^i and $r(c_j^i)$ the cell where π_j is located at iteration i, and the reward in this cell respectively. Let N_j^i and $R(N_j^i)$ denote respectively the set of neighbour cells of agent π_j (including c_j^i) at iteration i, and the reward values in these cells. $\mathtt{R}(c_j^i, a)$ is a function that returns the reward agent π_j gets after performing action $a \in \mathcal{A}$ when it is in cell c_j^i. The behaviour of a local search agent π_j at iteration i is defined as follows:

$$a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} \mathtt{R}(c_j^i, a).$$

If all actions return an equivalent reward, then a random action in $\mathcal{A}$ is selected.

Q-learning agent: in this reinforcement learning behaviour, the evaluated Q-learning [47] agent learns using a state-action pair quality function, $Q : S \times \mathcal{A} \rightarrow \mathbb{R}$, in order to find the action-selection policy that maximises its rewards. Each test episode of ϑ iterations is equivalent to one training session. Because the testing environment is dynamic, we define a Q-learning state $s_i \in S$ that an agent π_j occupies at iteration i as the pair $\{c_j^i, i\}$ consisting of π_j's current cell position c_j^i and the current iteration i, thus leading to a total number of states $|S| = \mathrm{m} \times \mathrm{n} \times \vartheta$, in a m-by-n environment space, over one test episode. The Q-Learning behavior over one training session is illustrated in Algorithm 1. We use the notations from the previous (Local search agent) paragraph. After training is complete, the evaluated agent simply travels between states by performing the actions with the highest reward values recorded in its Q-table.

[4] Several agent behaviours (isolated and collectives) other than those discussed in this paper have also been implemented and are made available in [6] for both testing and modification.

Algorithm 1 Q-Learning agent behavior over one training session.

Initialize: learning rate α and discount factor γ.
Begin
  for iteration $i \leftarrow 0$ **to** $\vartheta - 1$ **do** ▷ loop over iterations
    $s_i \leftarrow \{c_j^i, i\}$ ▷ set current state
    execute $a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} Q(s_i, a)$ ▷ perform action
    $s_{i+1} \leftarrow \{c_j^{i+1}, i+1\}$ ▷ set new (post-action) state
    $Q(s_i, a_j^i) = Q(s_i, a_j^i) +$ ▷ update Q-table
    $\alpha \left[\mathtt{R}(c_j^i, a_j^i) + \gamma \max_{a \in \mathcal{A}} Q(s_{i+1}, a) - Q(s_i, a_j^i) \right]$
  end for
End

Expert agent: an expert or oracle agent knows the future movements of the special object $\oplus$. At each step i of an episode this agent approaches the subsequent $i+1$ cell destination of $\oplus$ seeking maximum payoff. However, if $\oplus$ has a constant movement pattern (e.g., moves constantly to the right) pushing it away from the oracle, then the oracle will move in the opposite direction in order to intercept $\oplus$ in the upcoming test steps. Once it intercepts $\oplus$, it then continues operating using its normal behaviour.

Random agent: a random agent randomly choses an action from the finite set of actions $\mathcal{A}$ at each iteration until the end of an episode.

The scores of the random and oracle agents could be used as a baseline for the intelligence test scores of artificial agents, where a random agent is used as a lower bound on performance while the expert agent is used as an upper bound.

5.2.2 Agent collectives

The isolated agents could also be evaluated collectively (in groups) using a communication protocol to interact between one another. We propose a simple algorithm for enabling communication between local search agents using stigmergy [16] which is a form of indirect communication. For instance, we let local search agents induce fake rewards in the environment, thus indirectly informing neighbour agents about the proximity of the special objects. Note that fake rewards will not affect the score (real reward payoff) of the agents.

Let $\hat{R}(N_j^i)$ denote the set of fake rewards in the neighbour cells of agent π_j (including c_j^i) at iteration i, and $\hat{\mathtt{R}}(c_j^i, a)$ is a function returning the fake reward agent π_j receives after performing action $a \in \mathcal{A}$ when it is in cell c_j^i at iteration i. Fake rewards are induced in the environment according to Algorithm 2. Each agent proceeds by selecting an action by relying on fake rewards this time instead of the real rewards, as follows: $a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} \hat{\mathtt{R}}(c_j^i, a)$. If all actions

Algorithm 2 Stigmergic or indirect communication: fake reward generation over one iteration i of the test.

Input: Π (set of evaluated agents), $0 < \gamma < 1$ (fake reward discounting factor), a test iteration i.
Initialize: $\forall \pi_j \in \Pi$: $\hat{R}(N_j^i) \leftarrow 0.0$.
Begin
  for j $\leftarrow 1$ **to** $|\Pi|$ **do** ▷ loop over agents
    $r^{max} \leftarrow \max R(N_j^i)$
    $r^{min} \leftarrow \min R(N_j^i)$
    $\hat{r} \leftarrow \gamma \cdot r^{max} + (1 - \gamma) \cdot r^{min}$
    $\hat{R}(N_j^i) \leftarrow R(N_j^i) + \hat{r}$
  end for
End

are equally rewarding, then a random action is selected. Thereupon, we expect local search agents using stigmergy to form non-strategic coalitions after a few iterations of the test as a result of tracing the most elevated fake rewards in the environment.

In the case of Q-learning collectives, the agents in the collective could share and update a common Q-table, and thus all learn and coordinate simultaneously.

5.3 Modularity and Code Re-use

We have provided a large set of functionalities which might come in handy when amending and extending the current scope of the test, and for defining new agent behaviours to be evaluated. These functionalities can be found in the utility class *General* in [6] under the directory `/src/General.cpp`. Moreover, we have used UnitTest++ [33], a lightweight unit testing framework for C++ over Windows, in order to allow for easy defect isolation, assist in validating existing and newly implemented functionality, and encourage code review.

5.4 Experimental Demonstration

An example of an executable test experiment can be found in the *main* method in [6]. Similar types of experiment were conducted in our previous work [9] in order to identify and analyse factors influencing the intelligence of agent collectives. For instance, using the Λ^* environment, one could evaluate several types of multi-agent systems and quantitatively measure how:

- the complexity of the environment (its uncertainty and algorithmic complexity),
- the communication protocol used between the agents,
- the interaction time,
- and the agents' individual intelligence

all reflect (individually but also jointly) on the collective performance of the system. This can be easily achieved by running a series of controlled experiments in which the values (of one or more) of the above factors are altered.

6 ALTERNATIVE ENVIRONMENTS AND FURTHER CONSIDERATIONS

As mentioned in Section 4.1, the Λ^* (Lambda Star) environment focuses on a restricted set of canonical tasks particularly relevant to the intelligence of AI agents. Nonetheless, the generalisation of these canonical tasks does not account for a range of multiagent problems. In particular, the tasks to perform in the Λ^* environment are a nice abstraction of two problems in the literature (among others): searching for a moving target while avoiding injury, and nest selection when there is one and only one best nest. But these tasks do not cover other important multi-agent problems like those that require coordination (e.g., lifting and moving a table).

6.1 Measuring Multi-agent Coordination

Coordination is an important feature in multi-agent systems which has a high influence on their performance. Measuring coordination between interactive agents can be a difficult task. The scope of the Λ^* (Lambda Star) environment does not currently account for the measurement of coordination between agents, but we are considering extensions to assess this. For instance, problems that require coordination could have been evaluated if the payoff received from the *Good* object $\oplus$ (Section 4.1) had only occurred if two or more agents were in its neighborhood.

Another interesting extension to the test setting is to enable the environment to respond to the agent's behaviour and actions. Testing can be performed in an even more heterogeneous setting where the agents don't have the same reward function and/or actions and observations, and to give more attention or weight to the agent's learning (ability), which is an important aspect of intelligence.

Moreover, other properties like (environment) coverage could be evaluated by dispersing agents in the space to monitor what is happening in the environment (e.g., monitoring which neighborhoods are/are not explored by the agents).

6.2 Fitness Landscapes

Lambda Star (Λ^*) is one of many environments which can be used to evaluate artificial agents. A famous problem in AI is to evaluate the performance of artificial agents over fitness landscapes consisting of many local optima but only one global optimum. The landscapes reflect evaluations of some fitness or utilisation function over a set of candidate solutions. Adaptive landscapes can be considered where the underlying fitness evolves or changes over time. We have implemented a performance test based on the abovementioned problem description (by extending the Λ^* environment) and further designed a simulation depicting the behavior of (co-operative) artificial agents exploring a landscape over a period of time. The motivation is to assess the trade-off between exploration and exploitation in a reinforcement learning setting, and investigate the influences of this trade-off on the agents' payoffs in a multiple candidate solution space or environment. A screen-shot from the early stages of that simulation is given in Figure 5.

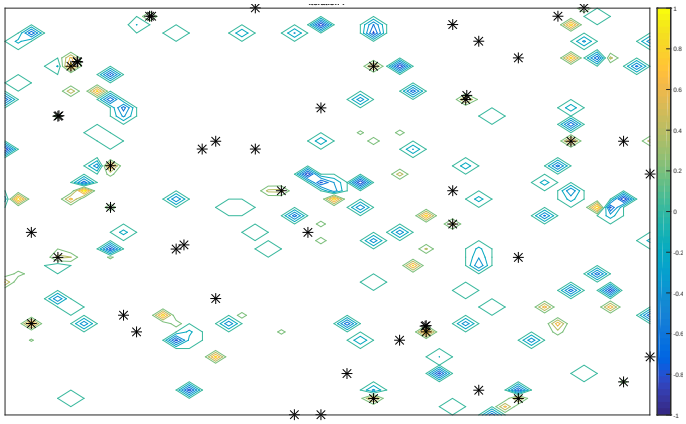

Figure 5: A screen-shot from the early stages of a simulation of a fitness landscape with many local optima but only one global optimum. Colors (and their different intensities showing in the right-hand side color-bar) represent fitness (ranging between $[-1.0, 1.0]$), or the quality of the landscape, at different spatial positions. The black stars represent agents navigating or searching the landscape.

6.3 Further Thoughts on Robust Intelligence Tests

The same way collective intelligence can emerge between artificial agents (due, for example, to the wisdom of the crowd [42], informa-

tion sharing, reduction in entropy, etc.), pluralistic ignorance [37] is also a common phenomenon observed in many social settings which can occur between rational agents. Robust intelligence tests should be able to detect such a phenomenon. For instance, the field of game theory has highlighted several scenarios where cooperation between agents does not leads to an optimal payoff (e.g., the famous prisoner's dilemma [35]). A robust intelligence test should be general enough to reflect and evaluate such scenarios.

Other multi-agent phenomena witnessed in various social settings reflect how agents acting individually might perform adversely to the common good, and thus deplete their available resources as a consequence of their collective behavior. A robust intelligence test should allow for a quantitative assessment of *the tragedy of the commons* [18] phenomena occurring in multi-agent scenarios.

7 CONCLUSIONS

This paper provides a technical description of the design and implementation of the Λ^* environment which can be used to evaluate general purpose AI agents both in isolation and collectively. The high-level evaluation architecture, based on an agent-environment framework, is discussed. The source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to existing and new artificial agents have been outlined and supported by examples.

We have also proposed and discussed some alternative testing environments that might be useful to quantify the performance of artificial agents. We further raised some arguments and considerations (in connection with pluralistic ignorance and the tragedy of the commons phenomena) that are relevant to the robustness of multi-agent performance tests.

Having presented the above, we encourage people in the AI community to evaluate new, more advanced, types of heuristics and algorithms over the Λ^* environment, and also extend its scope to include new functionality. Multi-agent systems can now be assessed using various interaction and communication protocols. This indicates that the performance of agent collectives can be quantitatively evaluated and compared to that of individual agents. This might help answer many open questions in AI regarding the emergence of collective intelligence in artificial systems.

REFERENCES

[1] Zeungnam Bien, Won-Chul Bang, Do-Yoon Kim, and Jeong-Su Han, 'Machine intelligence quotient: its measurements and applications', *Fuzzy Sets and Systems*, **127**(1), 3 – 16, (2002).

[2] Rodney A. Brooks, 'Intelligence without reason', in *Proc. of the 1991 International Joint Conference on Artificial Intelligence*, pp. 569–595, (1991).

[3] Gregory J. Chaitin, 'On the length of programs for computing finite binary sequences', *Journal of the ACM (JACM)*, **13**(4), 547–569, (1966).

[4] Gregory J. Chaitin, 'On the length of programs for computing finite binary sequences: statistical considerations', *Journal of the ACM (JACM)*, **16**(1), 145–159, (1969).

[5] Gregory J. Chaitin, 'Godel's theorem and information', *International Journal of Theoretical Physics*, **21**(12), 941–954, (1982).

[6] Nader Chmait. The Lambda Star intelligence test code-base. `https://github.com/nader-chmait/LambdaStar.git`, 2016.

[7] Nader Chmait, David L. Dowe, David G. Green, and Yuan-Fang Li, 'Observation, communication and intelligence in agent-based systems', in *Proc. 8th Int. Conf. Artificial General Intelligence, Berlin, Germany*, volume 9205 of *Lecture Notes in Artificial Intelligence (LNAI)*, pp. 50–59. Springer, (Jul 2015).

[8] Nader Chmait, David L. Dowe, David G. Green, Yuan-Fang Li, and Javier Insa-Cabrera, 'Measuring universal intelligence in agent-based systems using the anytime intelligence test', Technical Report 2015/279, FIT, Clayton, Monash University, (2015).

[9] Nader Chmait, David L. Dowe, Yuan-Fang Li, David G. Green, and Javier Insa-Cabrera, 'Factors of collective intelligence: How smart are agent collectives?', in *Proc. of 22nd European Conference on Artificial Intelligence (ECAI)*, (2016).

[10] David L. Dowe, 'MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness', in *Handbook of the Philosophy of Science - Volume 7: Philosophy of Statistics*, ed., P. S. Bandyopadhyay and M. R. Forster, pp. 901–982. Elsevier, (2011).

[11] David L. Dowe and Alan R. Hajek, 'A computational extension to the Turing Test', *Proc. 4th Conf. of the Australasian Cognitive Science Society, University of Newcastle, NSW, Australia*, (1997).

[12] David L. Dowe and Alan R. Hajek, 'A computational extension to the Turing Test', *Technical Report #97/322, Dept Computer Science, Monash University, Melbourne, Australia*, (1997).

[13] David L. Dowe and Alan R. Hajek, 'A non-behavioural, computational extension to the Turing Test', in *International conference on computational intelligence & multimedia applications (ICCIMA'98), Gippsland, Australia*, pp. 101–106, (1998).

[14] David L. Dowe, José Hernández-Orallo, and Paramjit K. Das, 'Compression and intelligence: Social environments and communication', in *Proc. 4th Int. Conf. on AGI*, pp. 204–211, Berlin, (2011). Springer.

[15] Scott C. Evans, John E. Hershey, and Gary Saulnier, 'Kolmogorov complexity estimation and analysis', in *6th World Conf. on Systemics, Cybernetics and Informatics*, (2002).

[16] Plerre-P. Grassé, 'La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs', *Insectes sociaux*, **6**(1), 41–80, (1959).

[17] Lawrence Gray, 'A mathematician looks at Wolfram's new kind of science', *Notices of the American Mathematical Society*, **50**(2), 200–211, (2003).

[18] Garrett Hardin, 'The tragedy of the commons', *Science*, **162**(3859), pp. 1243–1248, (1968).

[19] José Hernández-Orallo, 'Beyond the Turing Test', *J. of Logic, Lang. and Inf.*, **9**(4), 447–466, (October 2000).

[20] José Hernández-Orallo. A (hopefully) unbiased universal environment class for measuring intelligence of biological and artificial systems, 2010.

[21] José Hernández-Orallo, *The Measure of All Minds: Evaluating Natural and Artificial Intelligence*, Cambridge University Press, 2016. to appear.

[22] José Hernández-Orallo and David L. Dowe, 'Measuring universal intelligence: Towards an anytime intelligence test', *Artificial Intelligence*, **174**(18), 1508–1539, (December 2010).

[23] José Hernández-Orallo and David L. Dowe, 'On potential cognitive abilities in the machine kingdom.', *Minds and Machines*, **23**(2), 179–210, (2013).

[24] José Hernández-Orallo, Fernando Martinez-Plumed, Ute Schmid, Michael Siebers, and David L. Dowe, 'Computer models solving intelligence test problems: Progress and implications', *Artificial Intelligence*, **230**, 74 – 107, (2016).

[25] José Hernández-Orallo and Neus Minaya-Collado, 'A formal definition of intelligence based on an intensional variant of Kolmogorov complexity', in *Proc. of the Int. Symposium of EIS*, pp. 146–163. ICSC Press, (1998).

[26] Javier Insa-Cabrera, José-Luis Benacloch-Ayuso, and José Hernández-Orallo, 'On measuring social intelligence: Experiments on competition and cooperation', in *Proc. 5th Conf. on AGI*, eds., Joscha Bach, Ben Goertzel, and Matthew Iklé, volume 7716 of *LNCS*, pp. 126–135. Springer, (2012).

[27] Javier Insa-Cabrera, David L. Dowe, Sergio España-Cubillo, M. Victoria Hernandez-Lloreda, and Jose Hernandez-Orallo, 'Comparing humans and AI agents.', in *AGI*, volume 6830 of *LNCS*, pp. 122–132. Springer, (2011).

[28] Javier Insa-Cabrera, José Hernández-Orallo, David L. Dowe, Sergio España, and M Hernández-Lloreda, 'The ANYNT project intelligence test: Lambda-one', in *AISB/IACAP 2012 Symposium "Revisiting Turing and his Test"*, pp. 20–27, (2012).

[29] Andrei N. Kolmogorov, 'Three approaches to the quantitative definition of information', *Problems of information transmission*, **1**(1), 1–7,

(1965).
[30] Shane Legg and Marcus Hutter, 'Universal intelligence: A definition of machine intelligence', *Minds and Machines*, **17**(4), 391–444, (2007).
[31] Abraham Lempel and Jacob Ziv, 'On the Complexity of Finite Sequences', *Information Theory, IEEE Transactions on*, **22**(1), 75–81, (January 1976).
[32] Ming Li and Paul Vitányi, *An introduction to Kolmogorov complexity and its applications (3rd ed.)*, Springer-Verlag New York, Inc., 2008.
[33] Noel Llopis and Charles Nicholson. UnitTest++ testing framework. `https://github.com/unittest-cpp/unittest-cpp`, last accessed, April 2016.
[34] Graham Oppy and David L. Dowe, 'The Turing Test', in *Stanford Encyclopedia of Philosophy*, ed., Edward N. Zalta. Stanford University, (2011).
[35] William Poundstone, *Prisoner's dilemma*, Anchor, 2011.
[36] Pritika Sanghi and David L. Dowe, 'A computer program capable of passing I.Q. tests', in *Proc. of the Joint International Conference on Cognitive Science, 4th ICCS International Conference on Cognitive Science & 7th ASCS Australasian Society for Cognitive Science (ICCS/ASCS-2003)*, ed., P. P. Slezak, pp. 570–575, Sydney, Australia, (13-17 July 2003).
[37] Fatima B. Seeme and David G. Green, 'Pluralistic ignorance: Emergence and hypotheses testing in a multi-agent system', in *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, (2016).
[38] Claude E. Shannon, 'A mathematical theory of communication', *Bell System Technical Journal, The*, **27**(3), 379–423, (July 1948).
[39] Ray J. Solomonoff, 'A preliminary report on a general theory of inductive inference (report ztb-138)', *Cambridge, MA: Zator Co*, **131**, (1960).
[40] Ray J. Solomonoff, 'A formal theory of inductive inference. part I', *Information and control*, **7**(1), 1–22, (1964).
[41] Ray J. Solomonoff, 'A formal theory of inductive inference. part II', *Information and control*, **7**(2), 224–254, (1964).
[42] James Surowiecki, *The Wisdom of Crowds*, Anchor, 2005.
[43] Alan M. Turing, 'Computing machinery and intelligence', *Mind*, **59**, 433–460, (1950).
[44] Christopher S. Wallace, *Statistical and inductive inference by minimum message length*, Springer Science & Business Media, 2005.
[45] Christopher S. Wallace and David M. Boulton, 'An information measure for classification', *The Computer Journal*, **11**(2), 185–194, (1968).
[46] Christopher S. Wallace and David L. Dowe, 'Minimum message length and Kolmogorov complexity', *The Computer Journal*, **42**(4), 270–283, (1999). Special issue on Kolmogorov complexity.
[47] Christopher J. C. H. Watkins and Peter Dayan, 'Technical note: Q-learning', *Mach. Learn.*, **8**(3-4), 279–292, (May 1992).
[48] Eric W. Weisstein. Moore neighborhood, from mathworld - a Wolfram web resource. `http://mathworld.wolfram.com/MooreNeighborhood.html`, 2015. Last accessed: 2015-11-10.