Stochastic Tasks: Difficulty and Levin Search

José Hernández-Orallo

Departament de Sistemes Informàtics i Computació,

Universitat Politècnica de València, Spain.

jorallo@dsic.upv.es

AGI'2015 – The 8th Conference on Artificial General Intelligence,
Berlin, July 22-25

Outline

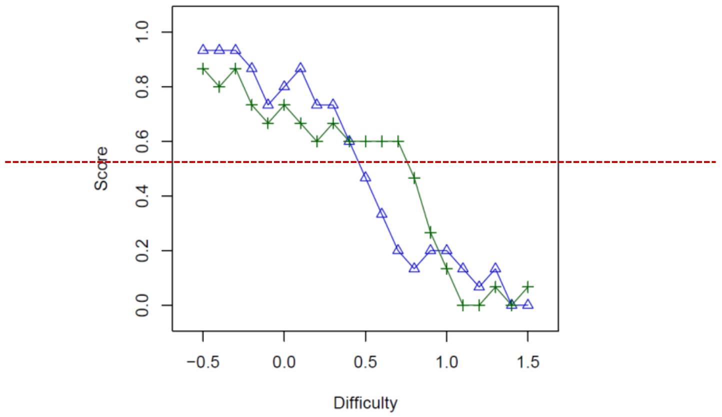
- Tasks, difficulty and evaluation
- Intuitive notion of difficulty
- Asynchronous stochastic tasks
- Difficulty as Levin's Kt
- Task instance difficulty
- Composition and decomposition
- Levin search with stochastic tasks
- Discussion

Tasks, difficulty and evaluation

- One classical approach to evaluation

> "The ability of an individual subject to perform a specified kind of task is the difficulty E at which the probability is ½ that he will do that task" (Thurstone 1937)

  - Basically, we analyse the result of tasks of different difficulty to determine the ability of the subject:
  - Example: response for agents according to task difficulty.



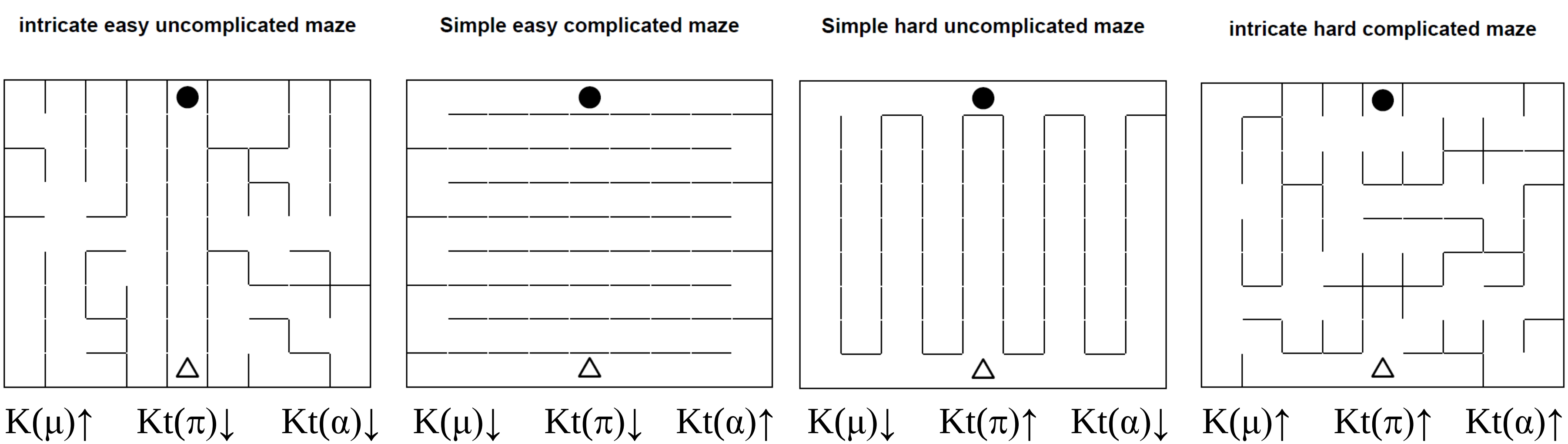
Tasks, difficulty and evaluation

- How is difficulty determined?
  - In psychometrics, difficulty is derived from a population.
  - Item response theory (IRT) is a common approach nowadays.
  - Using an appropriate selection of tasks and ranges of difficulty, we can perform adaptive tests that measure intelligence.
    - Intelligence is understood in terms of task difficulty.

Can we derive difficulty of a task in a formal, computational way?

Intuitive notion of difficulty

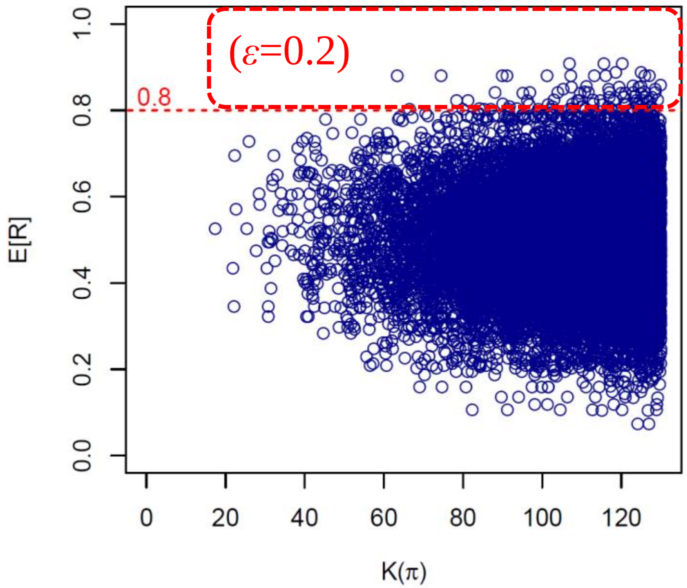
- Many approaches to the notion of difficulty:
  - Looking at the task description and characteristics ("intricate" task).
  - Looking at the solution description and characteristics.
    - By the solution we mean the "policy" that solves the general problem, not the particular execution, series of actions or path for a particular instance ("complicated" run).



Intuitive notion of difficulty

- The notion of difficulty is much better understood with a qualitative notion of solution, rather than a quantitative notion of performance.
- But what is a solution in an interactive task?
  - $\mathbb{R}^{[\mapsto\nu]}(\pi,\mu)$: response achieved by policy π in task μ after ν consecutive episodes or trials.
  - Either it is a goal-oriented task or…
  - we set a threshold.
    - Set of acceptable solutions:

$$\mathcal{A}^{[\epsilon,\mapsto\nu]}(\mu) \triangleq \{\pi \;:\; \mathbb{R}^{[\mapsto\nu]}(\pi,\mu) \geq 1-\epsilon\}$$



Using a threshold is also helpful for the commensurability for several environments.

Asynchronous stochastic tasks

- We can base our notion of difficulty on the length of the policy and the computational steps.

$$\mathbb{LS}^{[\mapsto\nu]}(\pi,\mu) \triangleq L(\pi) + \log \mathbb{S}^{[\mapsto\nu]}(\pi,\mu)$$

- Two terms:
  - $L(\pi)$ is given by the use of a policy description language.
  - How is $\mathbb{S}^{[\mapsto\nu]}(\pi,\mu)$ calculated?
    - For alternating tasks/environments such as a (PO)MDP, the computational steps can be shared among all the transitions or concentrated in a few transitions. This is not realistic.
    - The use of asynchronous tasks (agents can use an instruction "sleep(*t*)") allows for a more realistic calculation of $\mathbb{S}^{[\mapsto\nu]}(\pi,\mu)$.
    - We use a computational model with input and output tapes that can be read and written by the environment at any time (asynchronous).

Asynchronous stochastic tasks

- Why do we want to use stochastic tasks?
  - Many real problems are stochastic.
  - In multi-agent tasks, other agents (opponents or co-operators) are usually stochastic.

- What happens if we consider stochastic tasks/policies?
  - For the stochastic policies we need to use stochastic agents:
    - One possible model is based on probabilistic Turing machines (a Turing machine with access to a true random source) + sleep(*t*) instruction.
  - The computational steps are an expected value.
    - Hence our notation $\mathbb{S}^{[\mapsto\nu]}(\pi,\mu)$.

Difficulty as Levin's Kt

- Kt as an old idea for measuring difficulty as the effort from problem to solution ("gain": $Kt(s|p)$, Hernandez-Orallo 2000).
- Here, brought to asynchronous interactive tasks.
  - "Difficulty as the simplest acceptable policy".

$$Kt^{[\epsilon,\mapsto\nu]}(\mu) \triangleq \min_{\pi\in\mathcal{A}^{[\epsilon,\mapsto\nu]}(\mu)} \mathbb{LS}^{[\mapsto\nu]}(\pi,\mu)$$

- The above formula only considers the "simplest solution".
- Should we look at one or more solutions?
  - Looking at all solutions (weighted by $2^{-\mathbb{LS}}$) may be more accurate and less dependent of the policy description language.
    - But its estimation would be harder (not much to be gained as the difference would be bounded by a small constant).

Task instance difficulty

- A stochastic task is *also* a way of integrating several tasks.
  - It is at least as flexible as an aggregation of tasks using a distribution.
    - The aggregation can reach the threshold 1–ε by succeeding in some instance but not in others (unless ε=0).
  - But how is an instance defined?
    - An instance is given by setting a seed σ for the random tape: μ^{σ}.
  - How can we say that 'sort `gabcdef`' is easier than 'sort `gdaefcb`' without fixing an algorithm or a distribution of algorithms?

Task instance difficulty

- How is task instance difficulty defined relative to a task?
  - Not in terms of computational steps: the division 6/3 looks “easier” than 1252/626. But what about 13528/13528?

> Difficulty of an instance μ^σ is the minimum $\mathbb{LS}$ for any possible tolerance of a policy such that the instance is accepted.

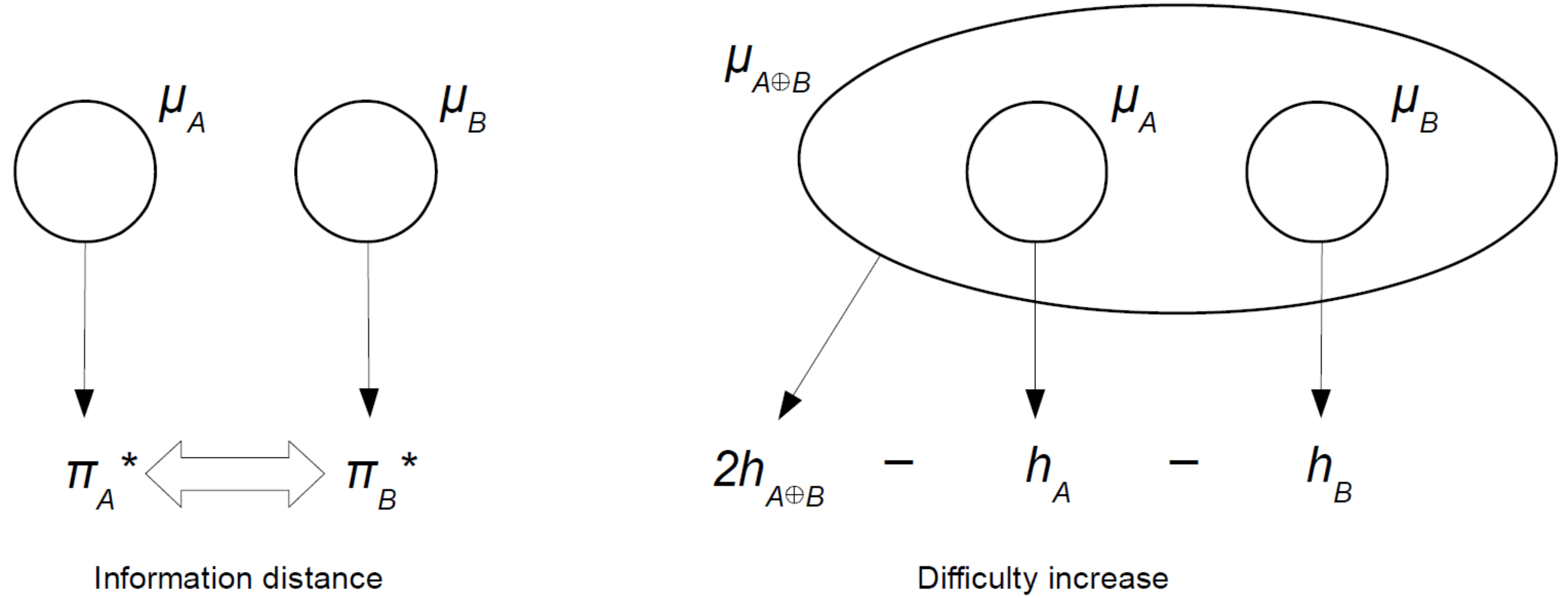
$$Opt_{\mathbb{LS}}^{[\mapsto\nu]}(\mu) \triangleq \left\{ \underset{\pi \in \mathcal{A}^{[\epsilon_0,\mapsto\nu]}(\mu)}{\arg\min} \ \mathbb{LS}^{[\mapsto\nu]}(\pi,\mu) \right\}_{\epsilon_0 \in [0,1]}$$

$$\hbar^{[\epsilon,\mapsto\nu]}(\mu^\sigma|\mu) \triangleq \min_{\pi \in Opt_{\mathbb{LS}}^{[\mapsto\nu]}(\mu) \cap \mathcal{A}^{[\epsilon,\mapsto\nu]}(\mu^\sigma)} \mathbb{LS}^{[\mapsto\nu]}(\pi,\mu)$$

  - Explanation: increase the tolerance until μ^σ is covered by the general policy.
    - That’s the difficulty of the instance.
    - When one constructs a solution, the easiest representative cases are covered first. This is related to consilience and coherence.

Task composition and decomposition

- Composition of stochastic tasks $A \oplus B$ is just defined as a stochastic choice using a (possibly biased) coin.
- From here, we have an alternative way of analysing task similarity.



Levin search with stochastic tasks

- Levin search for prefix Turing machines ensures a solution is found in at most $2^{L(p)} \cdot S(p)$ steps.
  - The logarithm is just Levin's Kt.
  - The unit of Kt can be said to be *logarithm of computational steps*, i.e., a logarithmic scale of the steps required for Levin's search to find a solution.

But Levin search assumes that verifying the solution is almost immediate.

- For stochastic tasks this is no longer the case…

We can never be sure of having found the solution

Levin search with stochastic tasks

- Approach: use a confidence level δ.

$$Pr(r^* - \widehat{r} \leq \epsilon) \geq 1 - \delta$$

- r^*: best possible result for any policy (e.g., 1)
- $\widehat{r}$: average result from the trials.
- We need several runs with the same policy to estimate the above probability with some confidence.
  - With the assumption that all runs take the same number of steps, we have the following verification cost (in steps):

$$\widehat{\mathbb{W}}^{[\epsilon,\delta]}(\pi,\mu) \triangleq \mathbb{S}(\pi,\mu) \cdot \mathbb{B}^{[\epsilon,\delta]}(\pi,\mu)$$

  - With $\mathbb{B}$ being the number of repetitions required to reach confidence δ.

Levin search with stochastic tasks

- Assuming a normal distribution, Levin search can be adapted, and the number of runs is given by:

$$\mathbb{B}^{[\epsilon,\delta]}(\pi,\mu) \triangleq \frac{|z_{\delta/2}|^2 \mathrm{Var}[R(\pi,\mu)]}{(\mathbb{R}(\pi,\mu)+\epsilon-r^*)^2}$$

- And finally, difficulty is obtained through:

$$\log \mathbb{F}^{[\epsilon,\delta]}(\pi,\mu) \triangleq \log(2^{L(\pi)} \cdot \widehat{\mathbb{W}}^{[\epsilon,\delta]}(\pi,\mu)) = L(\pi) + \log \widehat{\mathbb{W}}^{[\epsilon,\delta]}(\pi,\mu)$$

$$\hbar^{[\epsilon,\delta]}(\mu) \triangleq min_{\pi} \log \mathbb{F}^{[\epsilon,\delta]}(\pi,\mu)$$

$\mathbb{B}$ is an additive term so L may still be the most important term.

The point about all this is not whether we get a good approximation of $\mathbb{W}$ with the perhaps unrealistic assumptions, but that to clarify that the search will find those solutions that are well beyond the threshold with low variance first.

Discussion

- Tasks are asynchronous and stochastic.
  - This makes formalisation more unwieldy than common things such as (PO)(M)DPs, but some concepts are still straightforward.
  - Computational steps more meaningful.
- Difficulty as search effort: the (logarithm of) the steps required to find an acceptable solution policy.
- Associated and derived notions.
  - *Instance* difficulty can be defined relative to the task.
  - Difficulty can be used to analyse task composition and similarity.

We only need a formal language to describe the policies.
This can be applied to real, not-fully-specified tasks.

Discussion

- The correspondence of Levin's Kt with Levin search becomes more convoluted for stochastic tasks.
  - Natural phenomenon: solutions that are close to the tolerance level for an acceptable solution with high variance require more time to be verified, and hence are more difficult to find.
- We have used a rough approximation for this effect.
  - Still, the several runs to get confidence that a good solution being found will usually be a small additive factor.
  - The length of the policy will still dominate in the calculation of difficulty.