

# Combining Fast and Slow Thinking for Human-like and Efficient Navigation in Constrained Environments

M. Bergamaschi Ganapini<sup>1</sup>, M. Campbell<sup>2</sup>, F. Fabiano<sup>3</sup>, L. Horesh<sup>2</sup>, J. Lenchner<sup>2</sup>, A. Loreggia<sup>4</sup>, N. Mattei<sup>5</sup>, F. Rossi<sup>2</sup>, B. Srivastava<sup>6</sup> and K. B. Venable<sup>7</sup>

<sup>1</sup>Union College - USA

<sup>2</sup>IBM Research - USA

<sup>3</sup>University of Parma - Italy

<sup>4</sup>University of Brescia - Italy

<sup>5</sup>Tulane University - USA

<sup>6</sup>University South Carolina - USA

<sup>7</sup>University West Florida, IHMC - USA

## Abstract

Current AI systems lack several important human capabilities, such as adaptability, generalizability, self-control, consistency, common sense, and causal reasoning. We believe that existing cognitive theories of human decision making, such as the thinking fast and slow theory, can provide insights on how to advance AI systems towards some of these capabilities. In this paper, we propose a general architecture that is based on fast/slow solvers and a metacognitive component. We then present experimental results on the behavior of an instance of this architecture, for AI systems that make decisions about navigating in a constrained environment. We show how combining the fast and slow decision modalities, which can be implemented by learning and reasoning components respectively, allows the system to evolve over time and gradually pass from slow to fast thinking with enough experience, and that this greatly helps in decision quality, resource consumption, and efficiency.

## 1. Introduction

AI systems have seen great advancement in recent years, on many applications that pervade our everyday life. However, we are still mostly seeing instances of narrow AI that are typically focused on a very limited set of competencies and goals, e.g., image interpretation, natural language processing, classification, prediction, and many others. Moreover, while these successes can be accredited to improved algorithms and techniques, they are also tightly linked to the availability of huge datasets and computational power [1]. State-of-the-art AI still lacks many capabilities

---


AAAI 2022 FALL SYMPOSIUM SERIES, *Thinking Fast and Slow and Other Cognitive Theories in AI*, November 17-19, 2022, Westin Arlington Gateway in Arlington, Virginia, USA

✉ bergamam@union.edu (M. B. Ganapini); mcam@us.ibm.com (M. Campbell); francesco.fabiano@unipr.it (F. Fabiano); lhoresh@us.ibm.com (L. Horesh); lenchner@us.ibm.com (J. Lenchner); andrea.loreggia@unibs.it (A. Loreggia); andrea.loreggia@unibs.it (N. Mattei); Francesca.Rossi2@ibm.com (F. Rossi); BIPLAV.S@sc.edu (B. Srivastava); bvenable@uwf.edu (K. B. Venable)

🆔 0000-0001-8158-894X (M. Campbell); 0000-0002-1161-0336 (F. Fabiano); 0000-0001-6350-0238 (L. Horesh); 0000-0002-9427-8470 (J. Lenchner); 0000-0002-9846-0157 (A. Loreggia); 0000-0002-3569-4335 (N. Mattei); 0000-0001-8898-219X (F. Rossi); 0000-0002-7292-3838 (B. Srivastava); 0000-0002-1092-9759 (K. B. Venable)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

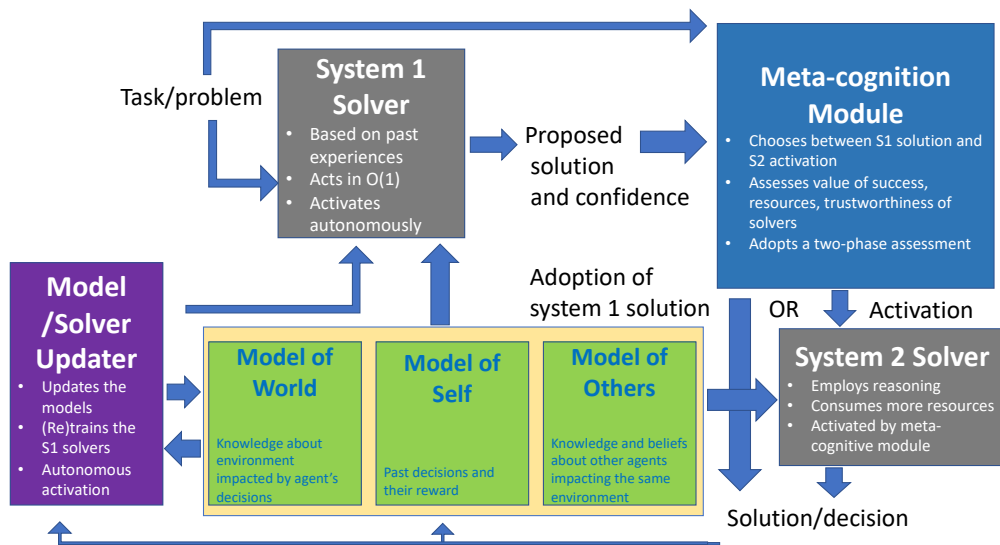
that would naturally be included in a notion of (human) intelligence, such as generalizability, adaptability, robustness, explainability, causal analysis, abstraction, common sense reasoning, ethical reasoning [2, 3], as well as a complex and seamless integration of learning and reasoning supported by both implicit and explicit knowledge [4].

We believe that a better study of the mechanisms that allow humans to have these capabilities can help [5]. We focus especially on D. Kahneman’s theory of thinking fast and slow [6], and we propose a multi-agent AI architecture (called SOFAI, for SLOW and Fast AI) where incoming problems are solved by either System 1 (or “fast”) agents (also called “solvers”), that react by exploiting only past experience, or by System 2 (or “slow”) agents, that are deliberately activated when there is the need to reason and search for optimal solutions beyond what is expected from the System 1 agents. Given the need to choose between these two kinds of solvers, a meta-cognitive agent is employed, performing introspection and arbitration roles, and assessing the need to employ System 2 solvers by considering resource constraints, abilities of the solvers, past experience, and expected reward for a correct solution of the given problem [7, 8]. Many approaches to the design of AI systems have been inspired by the dual-system theory [9, 10, 11, 12, 13, 14, 15], showing that this theory inspires many AI researchers.

In this paper we describe the SOFAI architecture, characterizing the System 1 and System 2 solvers and the role of the meta-cognitive agent, and provide motivations for the adopted design choices. We then focus on a specific instance of the SOFAI architecture, that provides the multi-agent platform for generating trajectories in a grid environment with penalties over states, actions, and state features. In this instance, decisions are at the level of each move from one grid cell to another. We show that the combination of fast and slow decision modalities, which can be implemented as learning and reasoning components, allows the system to create trajectories that are similar to human-like ones, compared to using only one of the modalities. Human-likeness is here exemplified by the trajectories built by a Multi-alternative Decision Field Theory model (MDFT) [16], that has been shown to mimic the way humans decide among several alternatives. In our case, the possible moves in a grid state, take into account non-rational behaviors related to alternatives’ similarity. Moreover, the SOFAI trajectories are shown to generate a better reward and to require a shorter decision time overall. We also illustrate the evolution of the behavior of the SOFAI system over time, showing that, just like in humans, initially the system mostly uses the System 2 decision modality, and then passes to using mostly System 1 when enough experience over moves and trajectories is collected.

## **2. Thinking Fast and Slow in AI**

In this section we give an overview of the SOFAI architecture, additional details are available in the Appendix. SOFAI is a multi-agent architecture (see Figure 1) where incoming problems are initially handled by those System 1 (S1) solvers that possess the required skills to tackle them, analogous to what is done by humans who first react to an external stimulus via their System 1.



**Figure 1:** The SOFAI architecture.

## 2.1. Fast and Slow Solvers

As mentioned, incoming problems trigger System 1 (S1) solvers. We assume such solvers act in constant time, i.e., their running time is not a function of the size of the input problem instance, by relying on the past experience of the system, which is maintained in the model of self. The model of the world contains the knowledge accumulated by the system over the external environment and the expected tasks, while the model of others contains the knowledge and beliefs about other agents who may act in the same environment. The model updater agent acts in the background to keep all models updated as new knowledge of the world, of other agents, or new decisions are generated and evaluated.

Once an S1 solver has solved the problem (for the sake of simplicity, assume a single S1 solver), the proposed solution and the associated confidence level are available to the meta-cognitive (MC) module. At this point the MC agent starts its operations, with the task of choosing between adopting the S1 solver's solution or activating a System 2 (S2) solver. S2 agents use some form of reasoning over the current problem and usually consume more resources (especially time) than S1 agents. Also, they never work on a problem unless they are explicitly invoked by the MC module.

To make its decision, the MC agent assesses the current resource availability, the expected resource consumption of the S2 solver, the expected reward for a correct solution for each of the available solvers, as well as the solution and confidence evaluations coming from the S1 solver. In order to not waste resources at the meta-cognitive level, the MC agent includes two successive assessment phases, the first one faster and more approximate, related to rapid unconscious assessment in humans [17, 18], and the second one (to be used only if needed) more careful and resource-costly, analogous to the conscious introspective process in humans [19]. The next section will provide more details about the internal steps of the MC agent.

This architecture and flow of tasks allows for minimizing time to action when there is no need

for S2 processing since S1 solvers act in constant time. It also allows the MC agent to exploit the proposed action and confidence of S1 when deciding whether to activate S2, which leads to more informed and hopefully better decisions by the MC.

Notice that we do not assume that S2 solvers are always better than S1 solvers, analogously to what happens in human reasoning [20]. Take for example complex arithmetic, which usually requires humans to employ System 2, vs perception tasks, which are typically handled by our System 1. Similarly, in the SOFAI architecture we allow for tasks that might be better handled by S1 solvers, especially once the system has acquired enough experience on those tasks.

## 2.2. The Role of Meta-cognition

We focus on the concept of meta-cognition as initially defined by Flavell [21], Nelson [22], that is, the set of processes and mechanisms that could allow a computational system to both monitor and control its own cognitive activities, processes, and structures. The goal of this form of control is to improve the quality of the system's decisions [23]. Among the existing computational models of meta-cognition [24, 25, 26], we propose a centralized meta-cognitive module that exploits both internal and external data, and arbitrates between S1 and S2 solvers in the process of solving a single task. Notice however that this arbitration is different from an algorithm portfolio selection, which is already successfully used to tackle many problems [27], because of the characterization of S1 and S2 solvers and the way the MC agent controls them.

The MC module exploits information coming from two main sources: 1) the system's internal models of self, world, and others; 2) the S1 solver(s), providing a proposed decision for a task, and their confidence in the proposed decision.

The first meta-cognitive phase (MC1) activates automatically as a new task arrives and a solution for the problem is provided by an S1 solver. MC1 decides between accepting the solution proposed by the S1 solver or activating the second meta-cognitive phase (MC2). MC2 then makes sure that there are enough resources for running S2. If not, MC2 adopts the S1 solver's proposed solution. MC1 also compares the confidence provided by the S1 solver with the risk attitude of the system: if the confidence is high enough, MC1 adopts the S1 solver's solution. Otherwise, it activates the next assessment phase (MC2) to make a more careful decision. The rationale for this phase of the decision process is that we envision that often the system will adopt the solution proposed by the S1 solver, because it is good enough given the expected reward for solving the task, or because there are not enough resources to invoke more complex reasoning.

Contrarily to MC1, MC2 decides between accepting the solution proposed by the S1 solver or activating an S2 solver for the task. To do this, MC2 evaluates the expected reward of using the S2 solver in the current state to solve the given task, using information contained in the model of self about past actions taken by this or other solvers to solve the same task, and the expected cost of running this solver. MC2 then compares the expected reward for the S2 solver with the expected reward of the action proposed by the S1 solver: if the expected additional reward of running the S2 solver, as compared to using the S1 solution, is large enough, then MC2 activates the S2 solver. Otherwise, it adopts the S1 solution.

To evaluate the expected reward of the action proposed by S1, MC2 retrieves from the model of self the expected immediate and future reward for the action in the current state (approximating the forward analysis to avoid a too costly computation), and combines this information with the

confidence the S1 solver has in the action. The rationale for the behavior of MC2 is based on the design decision to avoid costly reasoning processes unless the additional cost is compensated by an even greater additional expected reward for the solution that the S2 solver will identify for this task. This is analogous to what happens in humans [7].

### 3. Instantiating SOFAI on Grid Navigation

In the SOFAI instance that we consider and evaluate in this paper, the decision environment is a  $9 \times 9$  grid and the task is to generate a trajectory from an initial state  $S_0$  to a goal state  $S_G$ , by making moves from one state to an adjacent one in a sequence, while minimizing penalties.

Such penalties are generated by constraints over moves (there are 8 moves for each state), specific states (grid cells), and state features (in our setting, these are colors associated to states). For example, there could be a penalty for moving left, for going to the cell (1,3), and for moving to a blue state. In our specific experimental setting, any move brings a penalty of  $-4$ , each constraint violation gives a penalty of  $-50$ , and reaching the goal state gives a reward of 10.

This decision environment is non-deterministic: there is a 10% chance of failure, meaning that the decision of moving to a certain adjacent state may result in a move to another adjacent state chosen at random. Figure 2 shows an example of our grid decision environment.

Given this decision environment, we instantiate the SOFAI architecture as follows: (1) one S1 solver, that uses information about the past trajectories to decide the next move (see below for details); (2) one S2 solver, that uses MDFT to make the decision about the next move; (3) MC agent: its behavior is described by Algorithm 1; (4) model of the world: the grid environment; (5) model of self: it includes past trajectories and their features (moves, reward, length, time); (6) no model of others.

In Algorithm 1:

- $nTraj(s_x, \{S, ALL\})$  returns the number of times in state  $s_x$  an action computed by solver  $S$  ( $ALL$  means any solver) has been adopted by the system; if they are below  $t_1$  (a natural number), it means that we don't have enough experience yet.
- $partReward(T)$  and  $avgReward(s_x)$  are respectively the partial reward of the trajectory  $T$  and the average partial reward that the agent gets when it usually reaches state  $s_x$ : if we are below  $t_2$  (between 0 and 1), it means that we are performing worse than past experience.
- $c$  is the confidence of the S1 solver: if it is below  $t_3$  (between 0 and 1) it means that our attitude to risk does not tolerate the confidence level.

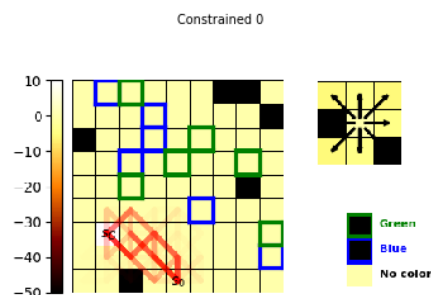


Figure 2: Example of the constrained grid decision scenario. Black squares represents states with penalties. Penalties are generated also when the agent moves left or bottom-right, or when it moves to a blue or a green state. The red lines describe a set of trajectories generated by the agent (all with the same start and end point). The strength of the red color for each move corresponds to the amount of trajectories employing such move.

---

**Algorithm 1** The MC agent

---

Input (Action  $a$ , Confidence  $c$ , State  $s_x$ , Partial Trajectory  $T$ )

```
1: if  $nTraj(s_x, ALL) \leq t_1$  or  $\frac{partReward(T)}{avgReward(s_x)} \leq t_2$  or  
    $c \leq t_3$  then  
2:   if  $nTraj(s_x, S2) \leq t_6$  then  
3:     randomly adopt S1 decision or activate S2 solver  
4:   else  
5:      $expCost_{S2} \leftarrow \frac{expTime_{S2}}{remTime}$   
6:     if  $expCost_{S2} \leq 1$  and  
        $\frac{(expReward_{S2}(s_x) - expReward(s_x, a))}{expCost_{S2}} > t_4$  then  
7:       Set the attention weights in  $W$   
8:       Activate the S2 solver  
9:     else  
10:      Adopt S1 decision  
11:    end if  
12:  end if  
13: else  
14:   Adopt S1 decision  
15: end if
```

---

If any of the tests in line 1 are passed (meaning, the condition is not satisfied), the MC system (MC1) adopts the S1 decision. Otherwise, it performs a more careful evaluation (MC2):

- $t_6$  checks that the S2 solver has enough experience. If not, a random choice between S1 and S2 is made (line 3).
- Otherwise, it checks if it is convenient to activate S2 (line 6), comparing the expected gain in reward normalize by its cost.  $t_4$  gives the tolerance for this gain. If it is convenient, MC activates the S2 solver (line 8), otherwise it adopts S1's decision. In this evaluation,  $expTime_{S2}$  and  $remTime$  are respectively the average amount of time taken by S2 to compute an action and the remaining time to complete the trajectory;  $expReward_{S2}(s_x)$  and  $expReward(s_x, a)$  are the expected reward using S2 in state  $s_x$  and the expected reward of adopting action  $a$  (computed by S1) in state  $s_x$ . The expected reward for an action  $a$  in a state  $s_x$  is:  $E(R|s_x, a) = \sum_{r_i \in R_{s_x, a}} P(r_i|s_x, a) * r_i$ , where  $R_{s_x, a}$  is the set of all the rewards in state  $s_x$  taking the action  $a$  that are stored in the model of self;  $P(r_i|s_x, a)$  is the probability of getting the reward  $r_i$  in state  $s_x$  taking the action  $a$ . As the expected reward depends on the past experience stored in the model of self, it is possible to compute a confidence as follows:  $c(s_x, a) = sigmoid(\frac{(r-0.5)}{(\sigma+1e-10)})$ , where  $\sigma$  is the standard deviation of the rewards in  $s_x$  taking an action  $a$ ,  $r$  is the probability of taking action  $a$  in state  $s_x$ .

The S1 agent, given a state  $s_x$ , chooses the action that maximizes the expected reward based on the past experience. That is:  $\text{argmax}_a (E(R|s_x, a) * c(s_x, a))$ . MC1 and MC2 bear some resemblance to UCB and model-based learning in RL [28]. However, in SOFAI we decompose some of these techniques to make decisions in a more fine grained manner.

The S2 agent, instead, employs the MDFT machinery (see Section A.1.2) to make a decision, where the  $M$  matrix has two columns, containing the Q values of a nominal and constrained RL agents, and the attention weights  $W$  are set in three possible ways: 1) attention to satisfying the constraints only if we have already violated many of them (denoted by 01), 2) attention to reaching the goal state only if the current partial trajectory is too long (denoted by 10), and 3) attention to both goal and constraints (denoted by 02). We will call the three resulting versions

SOFAI 01, 10, and 02.

## 4. Experimental Results

We generated at random 10 grids, and for each grid we randomly chose: initial and final states, 2 constrained actions, 6 constrained states, 12 constrained state features (6 green and 6 blue). For each grid, we run: (1) two reinforcement learning agents: one that tries to avoid the constraint penalties to reach the goal (called RL Constrained), and the other that just tries to reach the goal with no attention to the constraints (called RL Nominal). These agents will provide the baselines; (2) the S1 solver; (3) the S2 solver (that is, MDFT): this agent will be both a component of SOFAI and the provider of human-like trajectories; (4) SOFAI 01, SOFAI 10, and SOFAI 02. Each agent generates 1000 trajectories. We experimented with many combinations of values for the parameters. Here, we report the results for the following configuration:  $t_1 = 200$ ,  $t_2 = 0.8$ ,  $t_3 = 0.4$ ,  $t_4 = 0$ ,  $t_6 = 1$ .

We checked which agent generates trajectories that are more similar to the human ones (exemplified by MDFT). Figure 3 reports the average JS-divergence between the set of trajectories generated by MDFT and the other systems. SOFAI agents perform much better than S1, especially in the 01 configuration.

We then compared the three versions of SOFAI to S1 alone, S2 alone, and the two RL agents, in terms of the length of the generated paths, total reward, and time to generate the trajectories, see Figure 4 and 5. It is easy to see that S1 performs very badly on all three criteria, while the other systems are comparable. Notice that RL Nominal represents a lower bound for the length criteria and an upper bound for the reward, since it gets to the goal with no attention to satisfying the constraints. For both reward and time, SOFAI (which combines S1 and S2) performs better than using only S1 or only S2.

We then passed from the aggregate results over all 1000 trajectories to checking the behavior of SOFAI and the other agents over time, from trajectory 1 to 1000. The goal is to see how SOFAI methods evolve in their behavior and their decisions on how to combine its S1 and S2 agents. Given that SOFAI 01 performs comparably or better than the other two versions, in the following we only show the behavior of this version and will denote it simply as SOFAI.

Figure 6 and 7 shows the length, reward, and time for each of the 1000 trajectories, comparing SOFAI to S1 and to S2. In terms of length and reward, S1 does not perform well at all, while SOFAI and S2 are comparable. However, the time chart shows that SOFAI is much faster than S2 and over time it also becomes faster than S1, even if it uses a combination of S1 and S2. This is due to the fact that S1 alone cannot exploit the experience gathered by S2 within SOFAI, so it generates much worse and longer trajectories, which require much more time. Perhaps the most

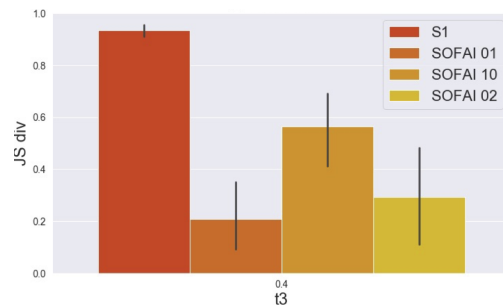
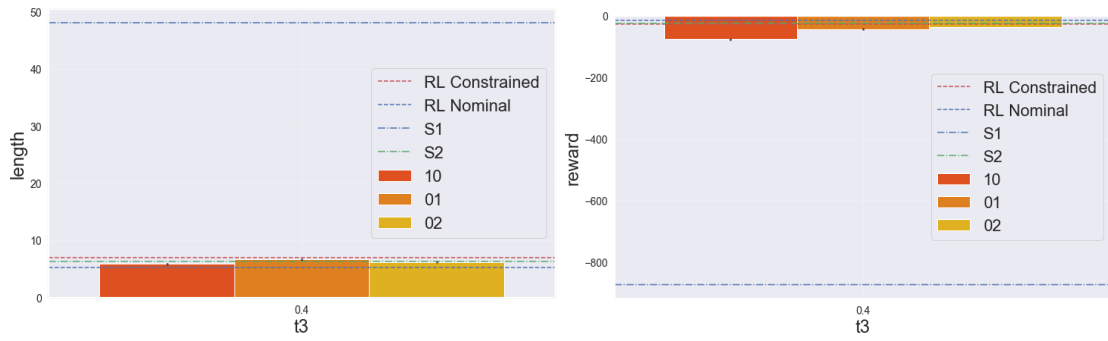
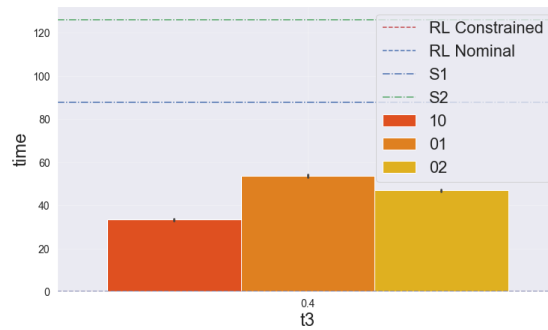


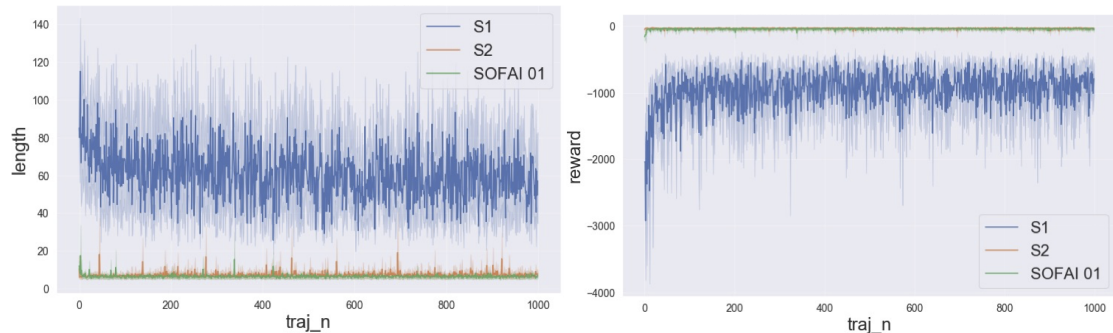
Figure 3: Average JS divergence between the set of trajectories generated by MDFT and the other systems.



**Figure 4:** Average length (left), reward (right), for each trajectory, aggregated over 10 grids and 1000 trajectories.



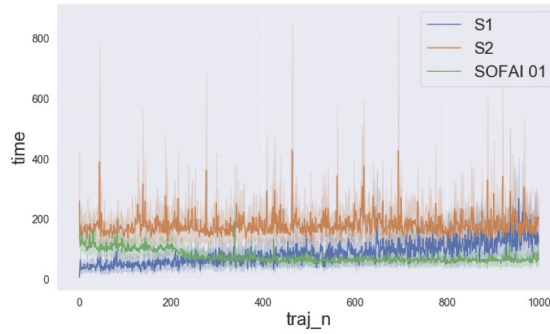
**Figure 5:** Average time for each trajectory, aggregated over 10 grids and 1000 trajectories.



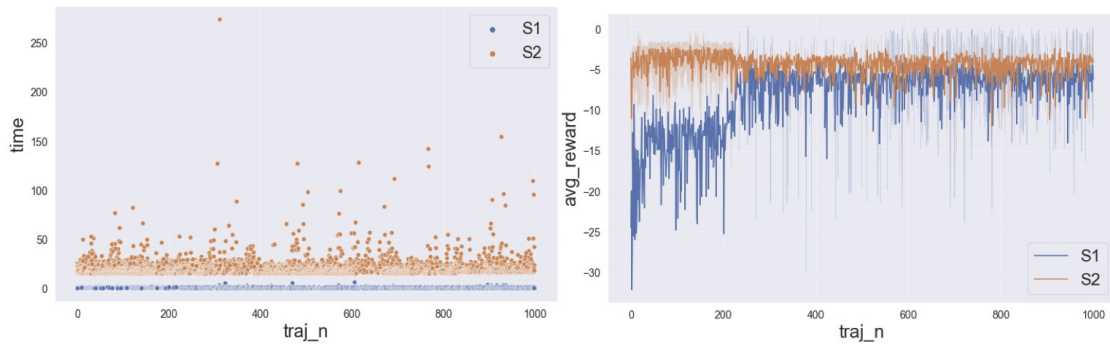
**Figure 6:** Average length (left) and reward (right), for each trajectory aggregated over 10 grids.

interesting is Figure 8. The left figure shows the average time spent by S1 and S2 within SOFAI in taking a single decision (thus a single move in the trajectory): S2 always takes more time than S1, and this is stable over time. The right figure shows the average reward for a single move: S2 is rather stable in generating high quality moves, while S1 at first performs very badly (since there is not enough experience yet) and later generates better moves (but still worse than S2). The question is now: how come S1 improves so much over time? The answer is given by Figure

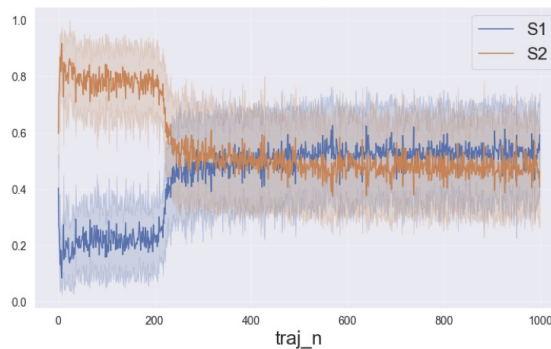




**Figure 7:** Average time to compute each trajectory aggregated over 10 grids.



**Figure 8:** Time to compute a move (left), average reward for a move (right) for each sub-system, over 10 grids.



**Figure 9:** Average fraction of times each sub-system is used over 10 grids.

9 which shows the percentage of usage of S1 and S2 in each trajectory. As we can see, at the beginning SOFAI uses mostly S2, since the lack of experience makes S1 not trustable (that is, the MC algorithm does not lead to the adoption of the S1 decision). After a while, with enough trajectories built by (mostly) S2 and stored in the model of self, SOFAI (more precisely, the MC agent) can trust S1 enough to use it more often when deciding the next move, so much that after about 450 trajectories S1 is used more often than S2. This allows SOFAI to be faster while not

degrading the reward of the generated trajectories. This behavior is similar to what happens in humans (as described in Section A.1.1): we first tackle a non-familiar problem with our System 2, until we have enough experience that it becomes familiar and we pass to using System 1.

## **5. Future Work**

We presented SOFAI, a conceptual architecture inspired by the thinking fast and slow theory of human decision making, and we described its behavior over a grid environment, showing that it is able to combine S1 and S2 decision modalities to generate high quality decisions faster than using just S1 or S2. We plan to generalize our work to allow for several S1 and/or S2 solvers and several problems for the same architecture, thus tackling issues of ontology and similarity.

## References

- [1] G. Marcus, The next decade in AI: Four steps towards robust artificial intelligence, arXiv preprint arXiv:2002.06177 (2020).
- [2] F. Rossi, N. Mattei, Building ethically bounded AI, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI), 2019.
- [3] F. Rossi, A. Loreggia, Preferences and ethical priorities: thinking fast and slow in AI, in: Proceedings of the 18th international conference on autonomous agents and multiagent systems, 2019, pp. 3–4.
- [4] M. L. Littman, et al., Gathering Strength, Gathering Storms: The One Hundred Year Study on Artificial Intelligence (AI100) 2021 Study Panel Report, Stanford University (2021).
- [5] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murgesan, N. Mattei, F. Rossi, B. Srivastava, Thinking fast and slow in AI, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 15042–15046.
- [6] D. Kahneman, Thinking, Fast and Slow, Macmillan, 2011.
- [7] A. Shenhav, M. M. Botvinick, J. D. Cohen, The expected value of control: an integrative theory of anterior cingulate cortex function, *Neuron* 79 (2013) 217–240.
- [8] V. A. Thompson, J. A. P. Turner, G. Pennycook, Intuition, reason, and metacognition, *Cognitive psychology* 63 (2011) 107–140.
- [9] Y. Bengio, The consciousness prior, arXiv preprint arXiv:1709.08568 (2017).
- [10] G. Goel, N. Chen, A. Wierman, Thinking fast and slow: Optimization decomposition across timescales, in: IEEE 56th Conference on Decision and Control (CDC), IEEE, 2017, pp. 1291–1298.
- [11] D. Chen, Y. Bai, W. Zhao, S. Ament, J. M. Gregoire, C. P. Gomes, Deep reasoning networks: Thinking fast and slow, arXiv preprint arXiv:1906.00855 (2019).
- [12] T. Anthony, Z. Tian, D. Barber, Thinking fast and slow with deep learning and tree search, in: Advances in Neural Information Processing Systems, 2017, pp. 5360–5370.
- [13] S. Mittal, A. Joshi, T. Finin, Thinking, fast and slow: Combining vector spaces and knowledge graphs, arXiv preprint arXiv:1708.03310 (2017).
- [14] R. Noothigattu, et al., Teaching AI agents ethical values using reinforcement learning and policy orchestration, *IBM J. Res. Dev.* 63 (2019) 2:1–2:9.
- [15] A. Gulati, S. Soni, S. Rao, Interleaving fast and slow decision making, arXiv preprint arXiv:2010.16244 (2020).
- [16] R. M. Roe, J. R. Busemeyer, J. T. Townsend, Multialternative decision field theory: A dynamic connectionist model of decision making., *Psychological review* 108 (2001) 370.
- [17] R. Ackerman, V. A. Thompson, Meta-reasoning: Monitoring and control of thinking and reasoning, *Trends in Cognitive Sciences* 21 (2017) 607–617.
- [18] J. Proust, The philosophy of metacognition: Mental agency and self-awareness, OUP Oxford, 2013.
- [19] P. Carruthers, Explicit nonconceptual metacognition, *Philosophical Studies* 178 (2021) 2337–2356.
- [20] G. Gigerenzer, H. Brighton, Homo heuristicus: Why biased minds make better inferences, *Topics in Cognitive Science* 1 (2009) 107–143.
- [21] J. H. Flavell, Metacognition and cognitive monitoring: A new area of cognitive–

- developmental inquiry., *American psychologist* 34 (1979) 906.
- [22] T. O. Nelson, *Metamemory: A theoretical framework and new findings*, in: *Psychology of learning and motivation*, volume 26, Elsevier, 1990, pp. 125–173.
  - [23] M. T. Cox, A. Raja, *Metareasoning: Thinking about thinking*, MIT Press, 2011.
  - [24] M. T. Cox, *Metacognition in computation: A selected research review*, *Artificial intelligence* 169 (2005) 104–141.
  - [25] J. D. Kralik, et al., *Metacognition for a common model of cognition*, *Procedia computer science* 145 (2018) 730–739.
  - [26] I. Posner, *Robots thinking fast and slow: On dual process theory and metacognition in embodied AI* (2020).
  - [27] P. Kerschke, H. H. Hoos, F. Neumann, H. Trautmann, *Automated algorithm selection: Survey and perspectives*, *Evolutionary computation* 27 (2019) 3–45.
  - [28] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, A Bradford Book, Cambridge, MA, USA, 2018.
  - [29] D. Kim, G. Y. Park, P. John, S. W. Lee, et al., *Task complexity interacts with state-space uncertainty in the arbitration between model-based and model-free learning*, *Nature communications* 10 (2019) 1–14.
  - [30] J. R. Busemeyer, J. T. Townsend, *Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment.*, *Psychological review* 100 (1993) 432.
  - [31] J. M. Hotaling, J. R. Busemeyer, J. Li, *Theoretical developments in decision field theory: Comment on tsetsos, usher, and chater (2010).*, *Psychological Review* (2010).

## A. Appendix

### A.1. Background

We introduce the main ideas of the thinking fast and slow theory. We also describe the main features of the Multi-alternative Decision Field Theory (MDFT) [16], that we will use in the experiments (Section 3 and 4) to generate human-like trajectories in the grid environment.

#### A.1.1. Thinking Fast and Slow in Humans

According to Kahneman's theory, described in his book "Thinking, Fast and Slow" [6], human's decisions are supported and guided by the cooperation of two kinds of capabilities, that for the sake of simplicity are called *systems*: System 1 ("thinking fast") provides tools for intuitive, imprecise, fast, and often unconscious decisions, while System 2 ("thinking slow") handles more complex situations where logical and rational thinking is needed to reach a complex decision.

System 1 is guided mainly by intuition rather than deliberation. It gives fast answers to simple questions. Such answers are sometimes wrong, mainly because of unconscious bias or because they rely on heuristics or other short cuts [20], and usually do not provide explanations. However, System 1 is able to build models of the world that, although inaccurate and imprecise, can fill knowledge gaps through causal inference, allowing us to respond reasonably well to the many stimuli of our everyday life.

When the problem is too complex for System 1, System 2 kicks in and solves it with access to additional computational resources, full attention, and sophisticated logical reasoning. A typical example of a problem handled by System 2 is solving a complex arithmetic calculation, or a multi-criteria optimization problem. To do this, humans need to be able to recognize that a problem goes beyond a threshold of cognitive ease and therefore see the need to activate a more global and accurate reasoning machinery [6]. Hence, introspection and meta-cognition is essential in this process.

When a problem is new and difficult to solve, it is handled by System 2 [29]. However, certain problems, over time as more experience is acquired, pass on to System 1. The procedures System 2 adopts to find solutions to such problems become part of the experience that System 1 can later use with little effort. Thus, over time, some problems, initially solvable only by resorting to System 2 reasoning tools, can become manageable by System 1. A typical example is reading text in our own native language. However, this does not happen with all tasks. An example of a problem that never passes to System 1 is finding the correct solution to complex arithmetic questions.

#### A.1.2. Multi-Alternative Decision Field Theory

Multi-alternative Decision Field Theory (MDFT) [16] models human preferential choice as an iterative cumulative process. In MDFT, an agent is confronted with multiple options and equipped with an initial personal evaluation for them along different criteria, called attributes. For example, a student who needs to choose a main course among those offered by the cafeteria will have in mind an initial evaluation of the options in terms of how tasty and healthy they look. More formally, MDFT comprises of:

**Personal Evaluation:** Given set of options  $O = \{o_1, \dots, o_k\}$  and set of attributes  $A = \{A_1, \dots, A_J\}$ , the subjective value of option  $o_i$  on attribute  $A_j$  is denoted by  $m_{ij}$  and stored in matrix  $\mathbf{M}$ . In our example, let us assume that the cafeteria options are *Salad* ( $S$ ), *Burrito* ( $B$ ) and *Vegetable pasta* ( $V$ ). Matrix  $\mathbf{M}$ , containing the student's preferences, could be defined as shown in Figure 10 (left), where rows correspond to the options ( $S, B, V$ ) and the columns to the attributes *Taste* and *Health*.

$$\mathbf{M} = \begin{bmatrix} 1 & 5 \\ 5 & 1 \\ 2 & 3 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & -1/2 & -1/2 \\ -1/2 & 1 & -1/2 \\ -1/2 & -1/2 & 1 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} +0.9000 & 0.0000 & -0.0405 \\ 0.0000 & +0.9000 & -0.0047 \\ -0.0405 & -0.0047 & +0.9000 \end{bmatrix}$$

**Figure 10:** Evaluation ( $\mathbf{M}$ ), Contrast ( $\mathbf{C}$ ), and Feedback ( $\mathbf{S}$ ) matrix.

**Attention Weights:** Attention weights are used to express the attention allocated to each attribute at a particular time  $t$  during the deliberation. We denote them by vector  $\mathbf{W}(t)$  where  $W_j(t)$  represents the attention to attribute  $j$  at time  $t$ . We adopt the common simplifying assumption that, at each point in time, the decision maker attends to only one attribute [16]. Thus,  $W_j(t) \in \{0, 1\}$  and  $\sum_j W_j(t) = 1, \forall t, j$ . In our example, we have two attributes, so at any point in time  $t$  we will have  $\mathbf{W}(t) = [1, 0]$ , or  $\mathbf{W}(t) = [0, 1]$ , representing that the student is attending to, respectively, *Taste* or *Health*. The attention weights change across time according to a stationary stochastic process with probability distribution  $\mathbf{w}$ , where  $w_j$  is the probability of attending to attribute  $A_j$ . In our example, defining  $w_1 = 0.55$  and  $w_2 = 0.45$  would mean that at each point in time, the student will be attending *Taste* with probability 0.55 and *Health* with probability 0.45.

**Contrast Matrix:** Contrast matrix  $\mathbf{C}$  is used to compute the advantage (or disadvantage) of an option with respect to the other options. In the MDFT literature [30, 16, 31],  $\mathbf{C}$  is defined by contrasting the initial evaluation of one alternative against the average of the evaluations of the others, as shown for the case with three options in Figure 10 (center).

At any moment in time, each alternative in the choice set is associated with a **valence** value. The valence for option  $o_i$  at time  $t$ , denoted  $v_i(t)$ , represents its momentary advantage (or disadvantage) when compared with other options on some attribute under consideration. The valence vector for  $k$  options  $o_1, \dots, o_k$  at time  $t$ , denoted by column vector  $\mathbf{V}(t) = [v_1(t), \dots, v_k(t)]^T$ , is formed by  $\mathbf{V}(t) = \mathbf{C} \times \mathbf{M} \times \mathbf{W}(t)$ . In our example, the valence vector at any time point in which  $\mathbf{W}(t) = [1, 0]$ , is  $\mathbf{V}(t) = [1 - 7/2, 5 - 3/2, 2 - 6/2]^T$ .

Preferences for each option are accumulated across the iterations of the deliberation process until a decision is made. This is done by using **Feedback Matrix**  $\mathbf{S}$ , which defines how the accumulated preferences affect the preferences computed at the next iteration. This interaction depends on how similar the options are in terms of their initial evaluation expressed in  $\mathbf{M}$ . Intuitively, the new preference of an option is affected positively and strongly by the preference it had accumulated so far, while it is inhibited by the preference of similar options. This lateral inhibition decreases as the dissimilarity between options increases. Figure 10 (right) shows  $\mathbf{S}$  for our example [31].

At any moment in time, the preference of each alternative is calculated by  $\mathbf{P}(t+1) = \mathbf{S} \times \mathbf{P}(t) + \mathbf{V}(t+1)$  where  $\mathbf{S} \times \mathbf{P}(t)$  is the contribution of the past preferences and  $\mathbf{V}(t+1)$  is the valence computed at that iteration. Starting with  $\mathbf{P}(0) = 0$ , preferences are then accumulated for either a fixed number of iterations (and the option with the highest preference is selected) or until the preference of an option reaches a given threshold. In the first case, MDFT models decision

making with a *specified* deliberation time, while, in the latter, it models cases where deliberation time is *unspecified* and choice is dictated by the accumulated preference magnitude. In general, different runs of the same MDFT model may return different choices due to the attention weights' distribution. In this way, MDFT induces choice distributions over set of options and is capable of capturing well know behavioral effects such as the compromise, similarity, and attraction effects that have been observed in humans and that violate rationality principles [30].