

Epistemic Planning in a Fast and Slow Setting

Francesco Fabiano^{1,*}, Marianna Bergamaschi Ganapini², Lior Horesh³,
Andrea Loreggia⁴, Keerthiram Murugesan³, Vishal Pallagani⁵, Francesca Rossi³ and
Biplav Srivastava⁵

¹*Department of Mathematical, Physical and Computer Sciences, University of Parma, Parma, Italy*

²*Philosophy Department, Union College, New York, USA*

³*Thomas J. Watson Research Center, Yorktown Heights, New York, USA*

⁴*Department of Information Engineering, University of Brescia, Brescia, Italy*

⁵*Artificial Intelligence Institute, University of South Carolina, Columbia, United States*

Abstract

AI applications are by now pervading our everyday life. Nonetheless, most of these systems lack many capabilities that, we humans, naturally consider to be included in a notion of “intelligence”. In this paper we present a multi-agent system, inspired by the cognitive theory known as thinking fast and slow by D. Kahneman, to solve Multi-agent Epistemic Planning (MEP) problems. This is an instance of a general AI architecture, referred to as SOFAI (for Slow and Fast AI). This paradigm exploits multiple solving approaches (referred to as fast and slow solvers) and a metacognition module to arbitrate between them and enhance the reasoning process, that, in this specific case, is concerned with planning in epistemic settings. The behavior of this system is then compared to a state-of-the-art MEP solver, showing that the newly introduced system presents better results in terms of generality, solving a much wider set of problems with an acceptable trade-off between solving times and solution accuracy.

Keywords

Fast and Slow AI, Epistemic Planning, Metacognitive Reasoning

1. Introduction

The AI community has formalized and developed several techniques that permit to model agents which can solve intricate problems in autonomy. These techniques range from the use of various formal logics to the creation of neural-based structures. The former is an area of study that tries to define rational behavior for our systems while the latter, imitating the physiology of our brain, aims to emulate (to some degree) the human behavior.

In this work we present an architecture that aims to allow multiple techniques to be exploited and we show its behavior on solving multi-agent epistemic planning problems (MEPs). This

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

*Corresponding author.

✉ francesco.fabiano@unipr.it (F. Fabiano); bergamam@union.edu (M. B. Ganapini); lhoresh@us.ibm.com (L. Horesh); andrea.loreggia@unibs.it (A. Loreggia); keerthiram.murugesan@ibm.com (K. Murugesan); vishalp@mailbox.sc.edu (V. Pallagani); francesca.rossi2@ibm.com (F. Rossi); biplav.s@sc.edu (B. Srivastava)
🆔 0000-0002-1161-0336 (F. Fabiano); 0000-0001-6652-9853 (M. B. Ganapini); 0000-0001-6350-0238 (L. Horesh); 0000-0002-9846-0157 (A. Loreggia); 0000-0001-8898-219X (F. Rossi); 0000-0002-7292-3838 (B. Srivastava)



© 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)

architecture is inspired by the well-know cognitive theory Thinking Fast and Slow by Kahneman [1], since it includes both “fast” and “slow” solvers and a metacognitive module to arbiter between the two. Slow (or System 2, System-2) solvers are solving problems by reasoning and (usually) exploiting symbolic techniques, while fast (or System 1, System-1) solvers just employ past experience to identify the solution to a given problem. A metacognitive module provides a centralized governance and chooses the best solver for the problem at hand [2].

In this work, we consider the planning domain and we employ an instance of this architecture that includes an existing epistemic planner as the slow solver and three case-based plan selectors as fast solvers. Experimental results on a widely used planning problem domain show that the behavior of our architecture is better than using the existing planner alone, both in terms of solved instance and average solving time.

Summarizing, the main contributions of this paper are:

- The definition of an architecture, inspired by the thinking fast and slow theory, to tackle epistemic planning problems;
- The characterization of fast and slow planners/solvers for the architecture, as well as the metacognitive module;
- Experimental results on an epistemic planning domain, showing the behavior of the architecture when using each of three different fast planners.

The paper is structured as follow. After this brief introduction, we introduce the background knowledge on the thinking fast and slow theory and multi-agent epistemic planning. We then describe how to unify the main concepts of the thinking fast and slow theory in the epistemic planning environment and we give a complete characterization of the architecture, with special emphasis on the metacognitive module and the two kinds of solvers. We follow with a description of the experimental setting, tables and graphs providing the experimental results showing the behavior of our system on planning problems, and a discussion of such results. We conclude the paper summarizing the main contribution and hinting at ongoing work.

2. Background

2.1. Thinking Fast and Slow

Thanks to improved algorithms, techniques, computational power, and dedicated hardware [3], the various automated reasoning tools are becoming more and more efficient and reliable in dealing with their areas of interest. However, all of these tools still lack capabilities that, we humans, naturally consider to be included in a notion of “intelligence” as, for example, generalizability, robustness, and abstraction. For these reasons, a growing segment of the AI community is attempting to address these limitations and is trying to create systems that display more “human-like qualities”. One of the central strategies to tackle this problem, adopted by various research groups [4, 5, 6, 7, 8], envisions tools, usually referred to as cognitive architectures [9], that exploit a combination of both the aforementioned approaches. In particular, in this paper,

we explore multi-agent epistemic planning in the context of one of these architecture that stems from a modern cognitive theory, *i.e.*, the well-known *Thinking Fast and Slow* paradigm proposed by D. Kahneman [1].

Kahneman’s theory states that humans’ reasoning capabilities are categorized into two diverse “Systems”, called System-1 (S1) and System-2 (S2). In particular, S1 identifies the intuitive and imprecise decision-making processes (“thinking fast”), while S2 provides tools to tackle all those situations where complex decisions need to be taken following logical and rational thinking (“thinking slow”). Other than problem difficulty, S1 and S2 discern which problem they should tackle based on the experience accumulated on the problem itself. That is, when a *new* non-trivial problem has to be solved, it is handled by S2. However, certain problems that initially can be solved only by S2, can later be solved by S1 after having accumulated a certain amount of experience. The reason is that the procedures used by S2 to find solutions to such problems also accumulate examples that S1 can later use readily with little effort. We note that S1 and S2 are not systems in the multi-agent sense, but rather they encapsulate two wide classes of information processing.

2.2. Multi-agent Epistemic Planning

Automated planning is a branch of artificial intelligence where the objective is to find plans, that is, sequences of actions, that can lead the agent to achieve desired goals. Epistemic planning is a specific form of planning where the agent must additionally deal with the epistemic notions of knowledge and beliefs. In the Multi-agent Epistemic Planning (MEP) problem, there are at least two planning agents. In what follows, we will only give an overview of MEP, referring the readers to Fagin et al. [10], Baral et al. [11], Van Ditmarsch et al. [12], Bolander [13] for details.

The idea of representing knowledge and beliefs has always been central in many research areas such as logics, philosophy, and computer science, *e.g.*, see Hintikka [14]. In this setting, we are concerned with finding the best series of action that modifies the information flows to enable the agents to reach goals that (might) refer to physical objects or agents’ knowledge/beliefs [15]. We will use the term “knowledge” to encapsulate *both* the notions of an agent’s knowledge and their beliefs. These concepts are distinct in *Dynamic Epistemic Logic* (DEL), which is the underlying basis of the MEP problem, but for simplicity in our high level introduction we will treat them as one. More details on the actual differences on these concept can be found in Fagin et al. [10].

While completely presenting DEL and the multi-agent epistemic setting background is not in the scope of this paper, let us quickly introduce some fundamental concepts that would allow to provide an intuitive meaning to this area of study. In particular, in a MEP problem we are concerned with the knowledge of the agents about the world or about others’ knowledge. This information is expressed through *belief formulae*, *i.e.*, formulae that can express: *i*) physical properties of the worlds; *ii*) knowledge of some agent (or group of agents) about these properties; and *iii*) nested knowledge about others’ knowledge.

The semantics of DEL formulae is traditionally expressed using *pointed Kripke structures* [16]. The epistemic action language that we used in our work implements three *types of action* and three *observability relations* following the standard proposed by Baral et al. [11]. In particular, each agent is assumed to be able to perform one of the following types of action: 1) *World-altering*

action: used to modify certain properties (*i.e.*, fluents) of the world; 2) *Sensing* action: used by an agent to refine her beliefs about the world; and 3) *Announcement* action: used by an agent to affect the beliefs of other agents. Moreover, each agent is associated to one of the following observability relations during an action execution: 1) *Fully-observant*: the agent is aware of the action execution and also knows the effect of the action; 2) *Partially-observant*: the agent is aware of the action execution without knowing the effects of the action; and 3) *Oblivious*: the agent is not even aware of the action execution. Each type of action defines a diverse transition function and alters an epistemic state in different ways.

Let us note that the amount of information carried within a single epistemic state (*i.e.*, a Kripke structure) and the high degree of liberty derived by complex transition functions (*e.g.* Baral et al. [11], Fabiano et al. [17, 18]) make planning in the multi-agent epistemic planning a very heavy-resource process that often brings unfeasibility to the planning task itself [19].

3. Thinking Fast and Slow in MEP

Two of the prominent lines of work in AI, *i.e.*, data-driven approaches and symbolic reasoning, seem to embody (even if loosely) the two Systems presented above. In particular, data-driven approaches shares with S1 the ability to build (possibly imprecise and biased) models from past experience, often represented by sets of data. For example, perception activities, such as seeing, that in humans are handled by S1, are currently addressed with machine learning techniques in AI. Similarly, S2's capability to solve complex problems using a knowledge-based approach is somewhat emulated by AI techniques based on logic, search, and planning, that make use of explicit and well-structured knowledge. While the parallelism data-driven-S1 and symbolic-S2 represent a starting point in developing an automated fast and slow AI, we should not assume these two techniques to be the exclusive representative of the respective System.

In this paper, we transpose the concepts derived by the thinking fast and slow paradigm into the MEP setting. We will start by presenting a general definitions for S1 and S2 solvers and then describe the actual implementations of S1 and S2 reasoners in the epistemic setting. We will make use of three *models* to represent key modules of our architecture, that from now on we will call Plan-SOFAI. In particular, the *model of self* is used to store the experience of the architecture, 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. Finally, the *model updater* 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.

The general characterization of a S1 solver, triggered immediately when the problem is presented to the Plan-SOFAI, does not require many factors.

- These solvers are assumed to rely on the past experience of the Plan-SOFAI itself.
- Moreover, we assume that the running time for S1 approaches to be independent of the input and, instead, to depend on the experience accumulated by the overall architecture, in the *model of self*.

- Finally, we consider a S1 solver to be an entity that relies on “intuition” (with a slight abuse of notation).

Considering these characteristics, the next question that naturally arises is *can MEP ever be considered as a S1 task, considering that epistemic planners, in literature, always rely on look-ahead strategies?* We considered some ideas that could help us develop a S1 epistemic planner. Among those, only a few were not using search methods (intensively) but rather mostly relied on experience. Finally, we identified a feasible, yet functional, way to exploit experience in the epistemic planning setting. The idea is to make use of *pre-computed plans*; that is, S1 can be used to determine which of the plans already generated by past experiences is the one that “fits the best” the current problem. Of course, determining if an already computed plan is a good choice or not for the current problem is a difficult research question on its own. Since the focus of this work is to devise a fast and slow architecture for epistemic planning rather than optimizing its internal components, we decided to use a simple, yet effective, criterion to select the best fitting plan. In particular, S1 selects, among past solutions for the same domain, the pre-computed plan that is the closest in term of *Levenshtein Distance* and *Jaccard Similarity* [20], as we will see in more detail later.

Our Plan-SOFAI is a S1-by-default architecture: whenever a new problem is presented, a S1 solver with the necessary skills to solve the problem starts working on it, generating a solution and a confidence level. This allows to minimize the resource consumption making use of the much faster S1 solving process when there is no need for S2—that is when the solution proposed by S1 is “good enough”. Nevertheless, as for the human brain, S1 may encounter problems that it cannot solve, either due to its lack of experience or the inherent intricacy of the problem itself. These situations require, then, the use of more thought-out resolution processes, generally provided by S2 approaches. Notice that we do not assume S2 solvers to be always better than S1 solvers: given enough experience, some tasks could be better solved by S1 solvers. This behavior also happens in human reasoning [21]. In the particular case of MEP, we can consider as S2 solving procedures the tools that employ traditional planning strategies. These can be, for example, the planner RP-MEP [22] or EFP 2.0 [17]. While these two solvers adopt different strategies to solve a Multi-agent Epistemic Planning problem, they both explore the search space and do not rely on experience. In particular, as we will see later, in this work we will make use of EFP 2.0 by Fabiano et al. [17].

4. The Fast and Slow MEP Architecture

4.1. The SOFAI Architecture

As main contribution of this paper, we present an architecture inspired by cognitive theories to solve the multi-agent epistemic planning problem. In particular, our tool is heavily based on a recent architecture called SOFAI [8] that is, in turn, inspired by the dual-system proposed by Kahneman [1]. Following the ideas of Kahneman, the architecture is equipped with two types of Systems dedicated to computing a solution to an incoming task, and a third agent in charge of orchestrating the overall reasoning task.

In this architecture, incoming problems are initially handled by S1 solvers that have the

required skills to tackle them. S1 solvers compute a solution relying on the past experience collected by the architecture. The computation is not affected by the size of the input problem and thus S1 solvers provide a solution in constant time. Past experience is maintained in the model of the self. A model updater agent is in charge of keeping all models updated as new something happens (e.g., new decisions are generated). Let us note that, in our case, the updater agent is in charge of storing each new solution found by S2.

The solution computed by the S1 solver (from now on for the sake of simplicity, let us assume it is just one S1 solver) and the corresponding confidence level is made available to the metacognitive (**MC**) module which can now choose between the proposed solution or engaging a S2 solver. An S2 agent is typically a reasoning model that is able to deal with the current problem, this kind of solver are more demanding in terms of time and other types of resources. For this reason, only **MC** agent can decide to activate an S2 solver.

MC agent assessment is based on several checks that are devoted to establishing whether it is worth adopting the solution proposed by S1 or engaging S2 for a more accurate solution. This allows for minimizing time to action when there is no need for S2 processing.

4.2. The Metacognitive Module

For our MEP solver, following the SOFAI architecture Ganapini et al. [8], we also defined a *metacognition process*. This means that we want our Plan-SOFAI to be equipped with a set of mechanisms that would allow it 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]. Metacognition models have been largely studied [24, 25, 9, 26] in the past years. Among the various proposed modalities, we envisioned our Plan-SOFAI to have a centralized metacognitive module that exploits both internal and external data and arbitrates between S1 and S2 solvers. Let us note that this module is structurally and conceptually different from an algorithm portfolio selection [27, 28].

We propose a metacognitive module that itself follows the *thinking fast and slow* paradigm. This means that our **MC** module is comprised of two main phases: the first one takes intuitive decisions without considering many factors, while the second one is in charge of carefully selecting the best solving strategy, considering all the available elements whenever the first phase did not manage to return an adequate solution. We will refer to the former with **MC-1** and to the latter with **MC-2**. **MC-1** is in charge of deciding whether to accept the solution proposed by the S1 solver or to activate **MC-2**. **MC-1** takes this decision considering the *confidence*, among others few factors, of the S1 solver: if the confidence, which usually depends on the amount of experience, is high enough, **MC-1** adopts the S1 solver’s solution.

If **MC-1** decides that the solution of the S1 solver is not “good enough”, it engages **MC-2**. Intuitively, this module needs to evaluate whether to accept the solution proposed by the S1 solver or which S2 solver to activate for the task. To do this, **MC-2** compares the expected reward for the S2 solver with the expected reward of the S1 one: if the expected additional reward of running the S2 solver, compared to the S1 one, is large enough, then **MC-2** activates the S2 solver. **MC-2**, following the human reasoning model [29], is designed to avoid costly reasoning processes unless the additional cost is compensated by an even greater expected reward for the solution that the S2 solver will devise.

5. Metacognition at Work

In what follows, we provide a “concrete” view of the S1/S2 framework for the multi-agent epistemic planning setting. To do so, we will make use of Algorithms 1–3.

Before going into detail, let us briefly comment on the input and on the parameters of these algorithms. The process requires the domain description (D), a particular instance (I) that we want to solve on such domain, and the time limit (TL) within which the instance needs to be solved. The parameters, instead, represent some internal values that capture some sort of “inclination” of the architecture towards employing S1. In particular, we have that:

- i) the acceptable correctness (A) represents the minimum ratio of solved goals, w.r.t. the total number of them, that defines an acceptable solution. Let us note that this measure can also be changed to depend on other factors or to account for goals importance, for example. Its default value is 0.5;
- ii) $T1$ represents the minimal amount of experience required by the Plan-SOFAI to consider a solution proposed by S1. Its default value is set to 20;
- iii) $T2$ represents the minimum number of S1 usages after which it will consider S1 accountable for its mistakes. This threshold allows the architecture to initially try to employ S1 more freely, to augment its experience. Conversely, after the minimum number $T2$ of solutions, the metacognition actually uses the previous performances of S1 to check for S1 accountability. Its default value is set to 20;
- iv) $T3$ is a value between 0 and 1 and it is used to represent the *risk-aversion* of the architecture: the higher the value the more incline the Plan-SOFAI is to use S2. The default value is set to 0.9;
- v) ϵ is a factor that is used to scale the probability that S1 solution may actually be employed even if it was not considered convenient. This is added to increase the number of S1 usages, and consequently its experience, in those situations where the low confidence of S1 itself may limit it too aggressively. Let us note that the solution proposed by S1 needs to be validated before being accepted in any case (Line 2 of Algorithm 2). Its default value is 0.1;
- vi) (M) represents the experience of the Plan-SOFAI. Every time a solution for a problem is found, this is stored in the memory with a series of useful information, *e.g.*, the correctness value, the employed system (*i.e.*, S1 or S2), the difficulty of the instance, the required time, and so on.

Algorithm 1 Fast and Slow MEP Architecture

Input: Domain (D), Instance (I), Time Limit (TL)

Parameter: Acceptable Correctness (A), $T1$, $T2$, $T3$, ϵ , Memory (M),

Output: Plan (S), Correctness (C)

```
1: Let  $p$  be the solution of  $I$  found by  $S1$ 
2: Let  $cx$  be the confidence of  $S1$  on  $p$ .
3: if  $|M.solved\_instances(D)| \geq T1$  then
4:   if  $|M.solved\_instances(D, S1)| < T2$  then
5:     Let  $K = 0$ 
6:   else
7:     Let  $avg\_corr = 0$ 
8:     for all  $i \in M.solved\_instances(D, S1)$  do
9:        $avg\_corr + = \frac{|i.solved\_goals()|}{|i.tot\_goals()|}$ 
10:    end for
11:    Let  $K = 1 - avg\_corr$ 
12:  end if
13:  if  $cx \times (1 - K) \geq T3$  then
14:    return  $\langle S, C \rangle = try\_S1(p, D, I, TL)$ 
15:  end if
16: end if
17: Let  $diff = I.compute\_difficulty()$ 
18: Let  $est\_t = M.avg\_t\_from\_diff(diff)$ 
19: Let  $rem\_t = TL - elapsed\_t$ 
20: Let  $est\_cost = \frac{est\_t}{rem\_t}$ 
21: if  $est\_cost > 1$  then
22:   return  $\langle S, C \rangle = try\_S1(p, D, I, TL)$ 
23: else
24:   Let  $prob = (1 - T3) \times \epsilon$ 
25:   if  $prob \geq generate\_random\_number(0, 1)$  then
26:     return  $\langle S, C \rangle = try\_S1(p, D, I, TL)$ 
27:   else
28:      $C = \frac{|I.solved\_goals(p)|}{|I.tot\_goals()|}$ 
29:     if  $C \geq A$  then
30:       if  $(1 - (est\_cost \times (1 - T3))) \geq C \times (1 - K)$  then
31:         return  $\langle S, C \rangle = solve\_with\_S2(p, D, I, rem\_t)$ 
32:       else
33:         return  $\langle S = p, C \rangle$ 
34:       end if
35:     else
36:       return  $\langle S, C \rangle = solve\_with\_S2(null, D, I, rem\_t)$ 
37:     end if
38:   end if
39: end if
```

We are now ready to describe in more detail Algorithms 1–3. Let us start by presenting Algorithms 1. In particular, we can identify **MC-1** in Lines 1–16, and **MC-2** in Lines 17–39. As already mentioned, to better emulate the thinking fast and slow paradigm, we assume *System-1* to automatically start and provide a solution at the beginning of **MC-1**. That is why we start the metacognitive process by storing the results of such process in the variables p and cx , that represent the solution found by *S1* and the confidence that *S1* has about this solution appropriateness, respectively. The metacognitive process then proceeds to check whether the experience accumulated by the architecture is enough to consider *S1* reliable (Line 3). If the architecture has enough experience, the metacognition considers the confidence of *S1*, adjusted to take into account the previous solutions proposed by *S1* itself (Lines 4–12), and determines whether *S1*'s confidence is within the tolerated risk, identified by $T3$ (Line 13). If the confidence of *S1* is enough, then the *Plan-SOFAI* tries to employ *S1*'s solution (line 14).

If, at any point, *S1*'s solution is considered not appropriate by the metacognitive process—because it violates some checks—then **MC-2** starts. This part of the procedure begins by determining a value that represents the difficulty of the problem instance (derived by various factors such as the number of agents, possible actions, fluents, and so on) at Line 17. This measure is then used to determine the average solving time for a given difficulty and to estimate the cost of solving the given problem (Line 18–20). If the cost exceeds 1 then there is not enough time to call *S2* and *Plan-SOFAI* tries to employ *S1*. The system can also adopt *S1* with a probability that is related with the risk aversion $T3$ and a parameter ϵ , this is done in Lines 24–26, to improve the exploration skill of the architecture itself. The *Plan-SOFAI* evaluates the solution proposed by *S1* and, if it is acceptable (Line 29), whether the extra time required by *S2* counterbalanced the cost (Line 30). If the solution is not acceptable or the increase in correctness is big enough *S2* is called (Line 36 and 31, respectively), otherwise the solution proposed by *S1* is used (Line 33).

Algorithm 2 *try_S1* function

Input: Plan (p), Domain (D), Instance (I), Time Limit (TL)

Parameter: Acceptable Correctness (A)

Output: Plan (S), Correctness (C)

```

1:  $C = \frac{|I.solved\_goals(p)|}{|I.tot\_goals()|}$ 
2: if  $C \geq A$  then
3:   return  $\langle S = p, C \rangle$ 
4: else
5:   return  $\langle S, C \rangle = solve\_with\_S2(null, D, I, TL)$ 
6: end if

```

Algorithms 2 and 3 are instead used to try and adopt the solution proposed by *S1* and to try and solve the problem with *S2*, respectively. In particular, Algorithm 2 takes the solution proposed by *S1* and checks whether it has an acceptable degree of correctness. If it does then the solution is employed, otherwise Algorithm 3 is called. This function simply calls the planner EFP 2.0 on the instance of the problem to solve and, if it terminates before the available time ends, it returns the plan found by the *S2* planner with confidence equal to 1. If EFP 2.0 cannot

Algorithm 3 solve_with_S2 function

Input: Plan (p), Domain (D), Instance (I) Time Limit (TL)

Output: Plan (S)

```
1: if EFP 2.0 ( $D,I$ ) terminates within  $TL$  then
2:   return  $\langle S = \text{EFP 2.0.get\_plan}(), C = 1 \rangle$ 
3: else if  $p \neq \text{null}$  then
4:   return  $\langle S = p, C = \frac{|I.\text{solved\_goals}(p)|}{|I.\text{tot\_goals}()} \rangle$ 
5: else
6:   OPT-OUT
7: end if
```

find the solution within the time limit then the solution from S1, if acceptable, is adopted; otherwise Plan-SOFAI returns no solution and terminates.

5.1. MEP S1 and S2 solvers

While in the previous paragraph, we described how our architecture decides which solving approach is the most appropriate, here we will provide an high level overview of solving processes themselves. In particular, we designed our S1 solver to solely rely on past experience, accumulated by using either S1 itself or S2. The S1 solver analyzes the memory and looks, through the various solved instances, which one is the closest to the problem that is being tackled. Once the closest instance is identified, S1 returns the plan associated to it as a solution and the distance value as measure for confidence.

This distance can be calculated in two different ways, generating effectively two different S1 solvers¹. The first measure of distance considers the problems as a set of formulae, *i.e.*, the ones that comprise the initial and goal states, and adopts the well-known Jaccard Similarity [20], that is the ratio between the union and the intersection of the two sets we are considering, as a metric for finding similarity between the input problem and the instances existing in the case library. The second metric is calculated by transforming the two instances into two distinct strings, once again comprised of all the initial and goal states information (separated by the special characters “|”), that are then compared using the Levenshtein distance [30] to determine the actual distance measure. Let us note that since we are considering only instances of the same domain, there is no need to incorporate other information. Nonetheless, if we would like to compare also instances of different domains, the domains’ description could easily be added to the representative of each instance.

While MEP S1 solvers did not exist in literature and, therefore, we needed to implement ad-hoc solutions, the same is not true for S2 planners. In fact, as mentioned we employed the state-of-the-art comprehensive epistemic planner EFP 2.0 as our S2 solver. Given that explaining how this planner works is beyond the scope of this paper, we can safely assume this approach to be a black-box that returns the best solution possible to a MEP problem, if exists. Nonetheless, we refer the interested readers to Fabiano et al. [17] for a detailed explanation on the internal

¹We compare the performances of these two S1 solvers later in the paper.

mechanisms of EFP 2.0.

6. Experimental Results and Discussion

6.1. Experimental Setup

In this section, we compare the new multi-agent epistemic planning architecture introduced as main contribution of this paper with EFP 2.0 [17] that, to the best of our knowledge, is the state-of-the-art comprehensive multi-agent epistemic solver. All the experiments were performed on 3.00GHz Intel Core i7-5500U machine with 16GB of memory.

As benchmark, we used a small variation of the standard epistemic planning domain known as **Coin in the Box (CB)** [31, 32, 11]. In this domain $n \geq 2$ agents are in one of two rooms, one of which contains a box with a coin inside. In the initial configuration, everybody knows that: *i*) none of the agents know whether the coin lies heads or tails up; *ii*) the box is locked; *iii*) only one agent has the key that opens the box; *iv*) each agent might be attentive or not w.r.t. to certain actions execution; Moreover, we know that each agent can execute one of the following actions: *i*) move: an agent can move to the other room; *ii*) open: an agent, if it has the key, can open the box; *iii*) peek: to learn whether the coin lies heads or tails up, an agent can peek into the box, but this requires the box to be open; *iv*) announce: this will result in all the listening (*i.e.*, attentive) agents to believe that the coin lies heads or tails up depending on the announced value; *v*) distract/signal another agent: these actions will make an attentive agent no more attentive or vice-versa, respectively. The goals usually consist in some agents knowing whether the coin lies heads or tails up while other agents know that it knows, or are ignorant about this.

The experiments consists of a set of 240 different instances of the **CB** domain that differ on the initial configurations, the goals, and in the number of acting agents. Regarding the various input and parameters of the architecture (used in Algorithms 1, 2, 3) we imposed: (i) a Time Limit (*TL*) of 90 seconds to solve each instance; (ii) an Acceptable Correctness (*A*) of 0.5, meaning that at least half of the goals must be satisfied for a S1 solution to be considered; (iii) the various thresholds (*i.e.*, *T1*, *T2*, *T3*) and ϵ to have they default values; and (iv) the Memory (*M*) to be empty at the beginning of the overall solving procedure.

6.2. Results

As baseline for our experiments, we used the solver EFP 2.0 [17]. As mentioned, we let the solver tackle all the 240 instances with a time-out of 90 second per instance. The main idea is that the results of the execution of EFP 2.0 represents the current capabilities of the MEP community and are comparable to a solely S2-based architecture. While this approach is guaranteed to find the best solution, if this exists (as it makes use of Breadth-First Search), it is not flexible enough to adapt to situations where the resources, namely time, are limited.

We then compared the results of EFP 2.0 with four different configurations of the architecture presented in this paper (Table 1). To avoid unnecessary clutter, let us identify the various configurations, with the following abbreviations:

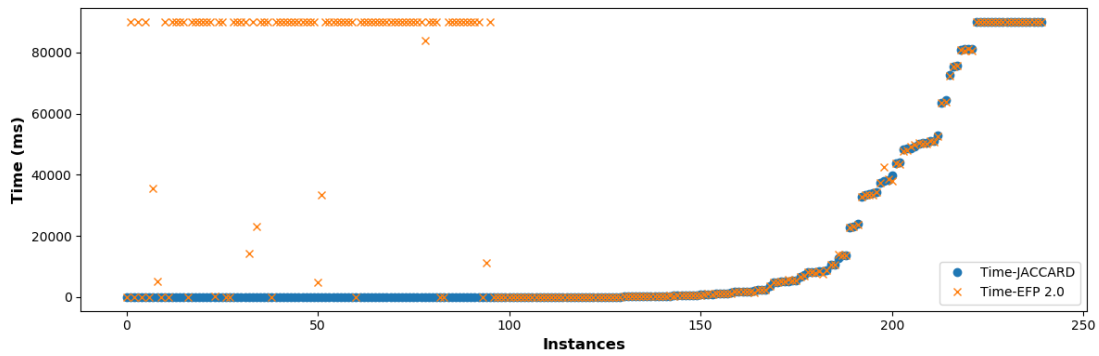


Figure 1: Time comparison between our Jac and EFP 2.0.

- **Jac:** the configuration of the architecture where the S1 makes use of the Jaccard similarity as notion of distance;
- **Lev:** the configuration where the metric for distance between problem is defined through Levenshtein distance;
- **Mix:** the configuration where S1 chooses the most similar instance in the memory, w.r.t. the given problem, by selecting the highest (normalized) score between the ones calculated with both the Jaccard similarity and the Levenshtein distance;
- **Rng:** the configuration where the S1 randomly picks one of the instances in memory as “most similar” without even considering a notion of distance. This approach was inserted as a baseline to outperform with a well-thought S1.

	EFP 2.0	Jac	Lev	Mix	Rng
Solved	149	222 (+49%)	207 (+39%)	197 (+32%)	181 (+21%)
Time (avg)	42.737	14.495	19.608	20.001	27.755
Corr (avg)	0.62	0.81	0.89	0.67	0.66
S1 calls	0	86	74	115	69

Table 1

Comparison between EFP 2.0 and the diverse configurations of our architecture on various parameters. The row Solved express the number of instances, out of the 240, that have been solved with correctness at least superior to 0.5. Time is expressed in seconds and the correctness (Corr) being the ratio between solved goals and their total number is a measure between 0 and 1. All the results also consider the instances that could not be solved, to these we assigned 90 seconds as time and 0 as correctness.

Table 1 shows that **Jac** is the best all-round configuration, with the most number of solved instances, the lowest average solving times and the highest average correctness. In fact, all configurations solved more problems (ranging from 21-49% more) and took less time on an average to find the plan. To provide further analysis we present, in Figure 1, a plot that shows

the resulting times of this configuration against EFP 2.0. In this, the instances (x-axis) are sorted w.r.t. the time employed by `Jac` to solve them. As we can see, all the solving time of `Jac` outperforms EFP 2.0 (equal or lower height on the y-axis). Let us note that the instances which are placed exactly on the top of the plot, *i.e.*, on the 90 seconds line, are the ones that have timed-out.

6.3. Discussion

Table 1 and Figure 1 highlight some interesting results about our architecture. In particular, thanks to its ability to “adapt” to resources constraints, our architecture can provide a flexible tool to solve instances even for those settings where the inherent complexity of the problems (*i.e.*, multi-agent epistemic planning) brings, most of the time, unfeasibility to the solving process. Our proposal can be seen as a way to exploit the accumulated experience, in a human fashion, to quickly solve known problems that otherwise would require a lot of efforts.

Moreover, the trade-off offered by allowing not-fully correct, but sound, plans is also a way to produce a solution that, even if partial, can be more useful than no solution at all. Let us note that some of the solved problem by `S1`, and not by EFP 2.0, do not have a solution that can satisfy all the goals. While the planning process of EFP 2.0 cannot detect these situations given the undecidability of MEP [19], our architecture simply reports a plan to reach some of the subgoals, providing once again a better alternative to no plan at all. Nonetheless, the parametric nature of the architecture allows also to tweak it so that it only accept fully correct plans. This can be done by setting the value of the acceptable correctness (A) to 1. With this small change the architecture will only return plans that reach the goal state while still taking advantage of the `S1` capabilities of the architecture (albeit `S1` solutions would be adopted less times). The same goes for the other internal parameters that easily allow the user to modify the architecture so that it is more prone to accept less accurate solution in favor of saving time, and vice-versa.

Another interesting result that we can deduce from Table 1 is that the employment of `Mix` is worse than both `Jac` and `Lev`. While, at first glance, this might seem a contradictory result it actually shows that the balance between the usage of `S1` and `S2` need to be preserved by the architecture. In fact, as `Mix` uses the combination of `Jac` and `Lev`, the number of time that a solution of `S1` is employed is higher than the two approaches (115 against 86 and 74). This makes it so that the architecture has fewer opportunities to increase its experience that can be re-used later to solve different problems. While it is not easy to find the best trade-off for limiting the employment of `S1`, it is our intention to define ways to automatically tune the internal parameters of the architecture so that it can exploits at maximum its capabilities.

7. Conclusions and Ongoing Work

In this work, we presented an architecture to solve multi-agent epistemic planning problems that is heavily inspired by the well-known cognitive theory Thinking Fast and Slow by Kahneman [1]. This tool builds on the SOFAI architecture [8], which makes use of a metacognitive process to arbitrate the solving processes, and two solvers referred to as `System-1` and `System-2`. While `S2` is directly derived from the literature, that is the comprehensive MEP solver known as EFP 2.0 [17], the `S1` solver (and its variations) have been designed ad-hoc for the proposed

architecture to exploit past experience by selecting a plan for a given planning instance. The SOFAI-inspired approach showed very promising results outperforming the state-of-the-art epistemic planner EFP 2.0 in terms of both solved instance and average solving times, on a set of 240 different planning problems. Another advantage of the proposed architecture is that it can be used to incorporate new solving techniques developed by the MEP community. In fact, our tool can be easily modified to employ different S1 or S2, or multiple versions of them. We are currently working on a version of the architecture that allows for multiple S2, in particular both EFP 2.0 and RP-MEP [22].

References

- [1] D. Kahneman, *Thinking, Fast and Slow*, Macmillan, 2011.
- [2] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murugesan, N. Mattei, F. Rossi, B. Srivastava, Thinking fast and slow in AI, in: *Proceedings of the 35th AAAI conference*, 2021, pp. 15042–15046.
- [3] G. Marcus, The next decade in ai: Four steps towards robust artificial intelligence, 2020. [arXiv:2002.06177](https://arxiv.org/abs/2002.06177).
- [4] A. Newell, SOAR as a unified theory of cognition: Issues and explanations, *Behavioral and Brain Sciences* 15 (1992) 464–492.
- [5] D. Friedlander, S. Franklin, Lida and a theory of mind, *Frontiers in Artificial Intelligence and Applications* 171 (2008) 137.
- [6] J. G. Trafton, L. M. Hiatt, A. M. Harrison, F. P. Tamborello, S. S. Khemlani, A. C. Schultz, ACT-R/E: An embodied cognitive architecture for human-robot interaction, *Journal of Human-Robot Interaction* 2 (2013) 30–55.
- [7] G. Goel, N. Chen, A. Wierman, Thinking fast and slow: Optimization decomposition across timescales, in: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 1291–1298.
- [8] M. B. Ganapini, M. Campbell, F. Fabiano, L. Horesh, J. Lenchner, A. Loreggia, N. Mattei, F. Rossi, B. Srivastava, K. B. Venable, Thinking fast and slow in AI: the role of metacognition, *CoRR abs/2110.01834* (2021). URL: <https://arxiv.org/abs/2110.01834>. [arXiv:2110.01834](https://arxiv.org/abs/2110.01834).
- [9] I. Kotseruba, J. K. Tsotsos, 40 years of cognitive architectures: core cognitive abilities and practical applications, *Artificial Intelligence Review* 53 (2020) 17–94. doi:10.1007/s10462-018-9646-y.
- [10] R. Fagin, Y. Moses, J. Y. Halpern, M. Y. Vardi, *Reasoning About Knowledge*, MIT press, 1995.
- [11] C. Baral, G. Gelfond, E. Pontelli, T. C. Son, An action language for multi-agent domains, *Artificial Intelligence* 302 (2022) 103601. URL: <https://www.sciencedirect.com/science/article/pii/S0004370221001521>. doi:<https://doi.org/10.1016/j.artint.2021.103601>.
- [12] H. Van Ditmarsch, W. van Der Hoek, B. Kooi, *Dynamic Epistemic Logic*, volume 337, Springer Netherlands, 2007. doi:10.1007/978-1-4020-5839-4.
- [13] T. Bolander, A gentle introduction to epistemic planning: The del approach, *arXiv preprint arXiv:1703.02192* (2017).

- [14] J. Hintikka, *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Ithaca: Cornell University Press, 1962.
- [15] T. Bolander, M. B. Andersen, Epistemic planning for single-and multi-agent systems, *Journal of Applied Non-Classical Logics* 21 (2011) 9–34. doi:10.1016/0010-0277(83)90004-5.
- [16] S. A. Kripke, Semantical analysis of modal logic i normal modal propositional calculi, *Mathematical Logic Quarterly* 9 (1963) 67–96. doi:10.1002/malq.19630090502.
- [17] F. Fabiano, A. Burigana, A. Dovier, E. Pontelli, EFP 2.0: A multi-agent epistemic solver with multiple e-state representations, in: *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling*, Nancy, France, October 26-30, 2020, AAAI Press, 2020, pp. 101–109. URL: <https://aaai.org/ojs/index.php/ICAPS/article/view/6650>.
- [18] F. Fabiano, A. Burigana, A. Dovier, E. Pontelli, T. C. Son, Multi-agent epistemic planning with inconsistent beliefs, trust and lies, in: *PRICAI 2021*, Hanoi, Vietnam, November 8-12, 2021, *Proceedings, Part I*, volume 13031 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 586–597. doi:10.1007/978-3-030-89188-6_44.
- [19] T. Bolander, M. Jensen, F. Schwarzentruher, Complexity results in epistemic planning, in: *IJCAI International Joint Conference on Artificial Intelligence*, volume 2015-January, 2015, pp. 2791–2797.
- [20] K. Rinarta, W. Suryasa, L. G. S. Kartika, Comparative analysis of string similarity on dynamic query suggestions, in: *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 2018, pp. 399–404. doi:10.1109/EECCIS.2018.8692996.
- [21] G. Gigerenzer, H. Brighton, Homo heuristicus: why biased minds make better inferences, *Top Cogn Sci* 1 (2009) 107–143. doi:10.1111/j.1756-8765.2008.01006.x.
- [22] C. J. Muise, V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce, L. Sonenberg, Planning over multi-agent epistemic states: A classical planning approach, in: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, January 25-30, 2015, Austin, Texas, USA, AAAI Press, 2015, pp. 3327–3334. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9974>.
- [23] M. T. Cox, A. Raja, *Metareasoning: Thinking about Thinking*, The MIT Press, 2011.
- [24] M. T. Cox, Metacognition in computation: A selected research review, *Artificial Intelligence* 169 (2005) 104–141. URL: <https://www.sciencedirect.com/science/article/pii/S0004370205001530>. doi:<https://doi.org/10.1016/j.artint.2005.10.009>, special Review Issue.
- [25] J. D. Kralik, J. H. Lee, P. S. Rosenbloom, P. C. Jackson, S. L. Epstein, O. J. Romero, R. Sanz, O. Larue, H. R. Schmidtke, S. W. Lee, K. McGreggor, Metacognition for a common model of cognition, *Procedia Computer Science* 145 (2018) 730–739. URL: <https://www.sciencedirect.com/science/article/pii/S1877050918323329>. doi:<https://doi.org/10.1016/j.procs.2018.11.046>, postproceedings of the 9th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2018 (Ninth Annual Meeting of the BICA Society), held August 22-24, 2018 in Prague, Czech Republic.
- [26] I. Posner, Robots thinking fast and slow: On dual process theory and metacognition in embodied AI, 2020. URL: <https://openreview.net/forum?id=iFQJmvUect9>.
- [27] P. Kerschke, H. H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection:

- Survey and perspectives, *Evol. Comput.* 27 (2019) 3–45. doi:10.1162/evco_a_00242.
- [28] A. Tarzariol, Evolution of algorithm portfolio for solving strategies, in: A. Casagrande, E. G. Omodeo (Eds.), *Proceedings of 34th CILC*, Trieste, Italy, June 19-21, 2019, volume 2396 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 327–341. URL: <http://ceur-ws.org/Vol-2396/paper37.pdf>.
- [29] 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. doi:10.1016/j.neuron.2013.07.007.
- [30] R. Haldar, D. Mukhopadhyay, Levenshtein distance technique in dictionary lookup methods: An improved approach, arXiv preprint arXiv:1101.1232 (2011).
- [31] F. Kominis, H. Geffner, Beliefs in multiagent planning: From one agent to many, in: *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015*, Jerusalem, Israel, June 7-11, 2015, AAAI Press, 2015, pp. 147–155. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617>.
- [32] X. Huang, B. Fang, H. Wan, Y. Liu, A general multi-agent epistemic planner based on higher-order belief change, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, Melbourne, Australia, August 19-25, 2017, ijcai.org, 2017, pp. 1093–1101. doi:10.24963/ijcai.2017/152.