A Question of Balance

The Benefits of Pattern-Recognition when Solving Problems in a Complex Domain

Martyn Lloyd-Kelly[1], Fernand Gobet[1], and Peter C. R. Lane[2]

[1] Psychological Sciences, University of Liverpool, L69 3BX, UK
{M.Lloyd-Kelly, Fernand.Gobet}@liverpool.ac.uk
[2] School of Computer Science, University of Hertfordshire, AL10 9AB, UK
peter.lane@bcs.org.uk

Abstract. The dual-process theory of human cognition proposes the existence of two systems for decision-making: a slower, deliberative, *problem-solving* system and a quicker, reactive, *pattern-recognition* system. We alter the balance of these systems in a number of computational simulations using three types of agent equipped with a novel, hybrid, human-like cognitive architecture. These agents are situated in the stochastic, multi-agent *Tileworld* domain, whose complexity can be precisely controlled and widely varied. We explore how agent performance is affected by different balances of problem-solving and pattern-recognition, and conduct a sensitivity analysis upon key pattern-recognition system variables. Results indicate that pattern-recognition improves agent performance by as much as 36.5% and, if a balance is struck with particular pattern-recognition components to promote pattern-recognition use, performance can be further improved by up to 3.6%. This research is of interest for studies of expert behaviour in particular, and AI in general.

1 Introduction

The notion of a *dual-process* cognitive system proposes that, given a problem, humans are equipped with two systems to make a decision about what to do. The formal logic-like *problem-solving* system is slow and deliberative, whereas the quicker *pattern-recognition* system uses fuzzier judgements of pattern similarity to propose solutions [40].

The pattern-recognition system has been proposed to be capable of creating, retrieving and using *productions* (a prescribed action for a particular environment state) to achieve the agent's relevant goal(s) [13, 40]. Psychological validity of this system is buttressed by human experimental evidence [15, 46] and implementations in computational cognitive architectures designed to emulate and explain human cognition [43].

Tension between use of problem-solving and pattern-recognition to solve problems has been identified by many [20, 47], resulting in the proposal that domain experts are, in some cases, inflexible problem-solvers, since they are so entrenched in established paradigms [39, 41]. This has been proven to be true, but only to a certain degree of expertise; once an above-average level of knowledge has been acquired about a domain, the so-called *Einstellung ~~E~~ffect* (a hallmark of expert inflexibility where satisfactory solutions block better ones) is removed [6].

A quantitative, scientific analysis of the potential effects on agent performance by weighting the usage of these systems and configuring their constituent components differently in particular complexities of a stochastic environment is lacking, to our knowledge, in the literature. So, in this paper, we provide such an analysis by investigating what balance of problem-solving and pattern-recognition is most effective when these systems are encapsulated in agents that are equipped with a human-like computational architecture of cognition. These agents are then situated in an environment whose complexity can vary considerably and where precise compile-time prescriptions of optimal actions using techniques such as Markov Decision Processes are implausible.

We compare three ways of making decisions: pure problem solving, “pure” pattern-recognition, and an equal mixture of the two decision-making systems. Our results are especially interesting for those who intend to design robust, self-learning systems that must achieve high levels of performance using information that is learned, whilst being engaged in reactive, sequential decision-making tasks. Our results are also of interest to those who wish to understand how pattern-recognition can improve performance in complex domains and how environment complexity affects the learning rates and performance of autonomous agents in general.

Section 2 discusses the simulation environment in detail and justifies its applicability. Section 3 presents a relevant overview of human cognition, and discusses the computational cognitive architecture in detail. Section 4 covers the implementation details of the agents implemented and Sect. 5 outlines the simulations run to gather data to answer the research questions posed. Section 6 delineates the results obtained and how they have emerged. The paper concludes with Sect. 7, containing the salient points raised by the simulation results, their implications for the current state of the art, and some future research directions.

2 Simulation Environment

The Tileworld environment [33] is a two-dimensional grid of homogeneously-sized squares that contains (along with multiple agents) a number of tiles and holes that exist for a finite period of time (see Fig. 1). An agent can move by up to one square per action, and its main goal is to push tiles into holes, earning the agent a reward and causing the tile and hole in question to disappear. An agent’s actions and goal achievement are episodic and delayed, since several actions may need to be performed in succession before a hole is filled, and only one tile can be pushed at any time by an agent. Explicit obstacles have not been included in this version of Tileworld, since tiles, holes and agents act as natural obstacles, given that a square can only be occupied by one object at a time.

Environmental complexity can be controlled by altering parameter values to manipulate *intrinsic* and *extrinsic* complexity. Intrinsic complexity is complexity that the environment has direct control over whereas extrinsic complexity is complexity that the environment has no direct control

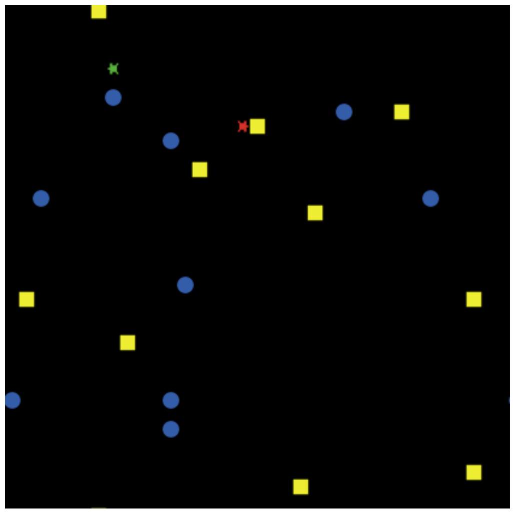


Fig. 1. Tileworld environment: agents are denoted by red and green turtle shapes, tiles by yellow squares and holes by blue circles.

over. In our simulations, intrinsic complexity is controlled by parameters that define when new tiles and holes can be created, the probability of a new tile or hole being created, and how long these artefacts exist for before being removed. Extrinsic complexity is controlled solely by altering the number of agents present in the environment: introducing more agents increases the environment state space by virtue of extra agents but also by increasing the chance of environment resources being interacted with. This potentially produces more environment states and increases environment complexity.

In a simplified version of Tileworld where only holes and one agent (no opponents) may exist, optimal policy calculations for a computer with reasonable resources using a Markov Decision Process (MDP) becomes intractable when the total number of squares in the Tileworld equals 16 or 25 (a 4×4 or 5×5 grid) [37]. In comparison, the Tileworld implemented in this paper is much more complex: tiles, holes, and opponents can exist simultaneously in mixed quantities and the environment comprises 1225 (35×35) squares in total (however, an agent can only "see" 25 squares in total at a time; see Sect. 4).

Taking the complexity of the entire environment into account at once, an environment with 2 to 8 agents yields $\approx 10^{589}$ to $\approx 10^{600}$ possible states (these numbers cover the range of agents used, see Sect. 5). From the perspective of a single agent, complexity ranges from $\approx 10^{12}$ states (2 agents) to $\approx 10^{14}$ states (8 agents).

Consequently, we assert that the complexity of this environment, along with the ability to exert fine-grained control over this complexity, provides a suitable test-bed to address the research questions posed.

3 Cognitive Architecture

In studying human cognition, much scrutiny has been focused upon explaining expert behaviour in complex domains; chess, in particular, has benefited from such effort. Research has identified that the difference in performance between chess masters and amateurs hinges upon the breadth and quality of knowledge possessed by masters [11].

The total amount of information in chess has been calculated to contain 143.09 bits of information or, 10^{43} positions ($2^{143.09}$) [23]. However, some of these positions are redundant or implausible; rectified calculations give a total space of 50 bits of information or, 10^{15} positions [11]. A promising psychological theory that accounts for the ability of chess masters to learn and retain such large[3] amounts of information, given known limitations on human cognition, is *chunking theory* [31]. Chunking theory suggests that information in memory is stored as aggregated environmental information or *chunks*; their existence and application in chess has been established through rigorous testing (see [8, 11] for good examples). Indeed, computational cognitive models that implement chunking theory have closely mimicked human behaviour in domains other than chess too (see Sect. 3.1 for details).

With regard to decision-making, chess masters demonstrate pattern-recognition frequently: key features of certain board configurations are recognised extremely quickly [10] and typical, good moves are used when better, relatively uncommon moves exist [6, 35]. This indicates that when an adequate knowledge base exists for a domain, pattern-recognition is the preferred *modus operandi* for human decision-making (examples in other domains are given in [6]). However, given that there may exist many possible solutions for a particular situation, how exactly is a decision regarding what to do reached?

One proposal is that pattern-recognition is underpinned by the existence and use of *productions* stored in memory. We define productions as being *if-else* conditions where the *if* component is visual information and the *then* component is an action that should be performed. Productions can be assigned utility ratings by a process akin to reinforcement learning [44], whose presence in human cognition has been extensively validated [12, 21]. These ratings enable production selection to be *rational*: better productions should be selected more frequently than worse ones [32]. When applied to domains like Tileworld, where an agent's actions and goal achievement are episodic, rating production utility entails using *discounted rewards*: productions performed closer to the time a reward for goal achievement is received are assigned, or have their utility ratings incremented by, an amount that reflects a greater share of the reward received than productions performed further in the past, according to a discount factor β, $(0 < \beta < 1)$ [19]. Human production selection is also non-deterministic; worse productions may be selected given better alternatives [9, 34].

[3] With respect to both the number of positions and the amount of information within each position.

Consequently, we have used the *Chunk Hierarchy and REtrieval STructures* (CHREST) architecture that implements chunking theory computationally [16, 11] as the core of our cognitive architecture. Additionally, we have implemented a pattern-recognition and problem-solving system. The pattern-recognition system is considered a part of CHREST since it depends upon particular CHREST components and handles CHREST-compatible information. The problem-solving system is domain-specific and does not directly interact with any part of CHREST.

To enable correct operation of the pattern-recognition system, the *Profit Sharing with Discount Rate* (PSDR) reinforcement learning theory [4] and the *Roulette* selection algorithm [5] have also been used. The CLARION cognitive architecture [42, 43] adopts a similar combination of elements but usually combines a backpropagation network, Q-learning theory [45] and a Boltzmann distribution selection algorithm. However, such a combination is not capable of handling stochastic environments so is unsuitable for our purposes; Q-learning in particular fails to converge in domains similar to and smaller than the version of Tileworld used in this paper [4]. PSDR also requires that an agent have access to an *episodic memory* data structure that tracks what actions have been performed by the agent in response to particular environmental stimuli.

Due consideration must be afforded to how the problem-solving and pattern-recognition systems exert control over decision-making given that we are interested in examining how a human-like system of cognition performs in the stochastic domain described. The research discussed proposes that problem-solving and pattern-recognition never operate simultaneously in human cognition. So, after observing the current state of the environment, agents will first use pattern-recognition to generate a solution. If no solution is output from the pattern-recognition system, the problem-solving system is used instead. Implementing decision-making in this way creates a *modular* dual-process decision-making system [26], and extends CHREST's existing functionality.

In this section, Sect. 3.1 discusses pertinent features of CHREST and Sects. 3.2 and 3.3 outline the problem-solving and pattern-recognition systems. Section 3.4 details the episodic memory structure and Sects. 3.5 and 3.6 discuss the PSDR and Roulette algorithms.

3.1 CHREST

CHREST is an example of a symbolic cognitive computational architecture [36] and is capable of creating extensible knowledge-bases that enable human-like storage, organisation and retrieval of memory. CHREST also provides functionality to create productions by generating directed links between chunks and can store utility ratings for productions by associating numeric values with these links. The validity of CHREST as a theory of human-like cognition has been established in a variety of domains, including board games [7, 11], implicit learning [26] and natural language acquisition [14, 22]. A version of CHREST similar to that described in this section has also investigated how binding rationality affects the performance and learning rates of Tileworld agents [29].

CHREST comprises two main components: *short-term memory* (STM) and *long-term memory* (LTM). Unlike cognitive architectures such as Soar [25] and ACT-R [2], CHREST does not encode information in LTM as procedural, declarative or semantic. Rather, information is organised according to its *modality*, of which CHREST defines three types: action, visual and verbal (no verbal information is used in our simulations).[4]

Knowledge Representation. In the CHREST framework, units of information called *patterns* are used to represent knowledge about the environment and chunks refer to LTM knowledge. To create a chunk, a single pattern or combination of patterns must be used to create or modify a LTM *node*. When this occurs, the input pattern becomes, or is added to, a node's *image*. Therefore, a chunk is strictly defined as being the content of a node's image.

Since patterns represent knowledge about the environment, they are created by an agent's domain-specific *input/output* component (see Sect. 4.2 for implementation details of this component). In the simulations run, the patterns used are instances of three-item tuples called *item-on-square* patterns that contain an identifier string and two numbers (see captions for Figs. 2 and 3 for examples).

For visual item-on-square patterns, the identifier string represents the *class* of object seen. In these simulations, `T` encodes a tile, `H` encodes a hole and `A` encodes another agent. An agent does not encode its own location in a pattern (and consequently, the chunks it creates) since all objects are encoded relative to its current location. The first and second numbers represent how many squares to the north and east of the agent the object is, respectively.[5] For example, the visual pattern `<[T 1 2]>`, states that there is a tile (`T`) located 1 square north (`1`) and 2 squares east (`2`) of the agent's current location.

For action item-on-square patterns, the identifier string represents an action that an agent should perform (see Table 1 for the mappings between identifiers and actions). The first of the two numbers represents the compass direction the agent should face when performing the action: 0 = north, 90 = east etc., and the second number denotes the number of squares that should be moved by the agent when the action is performed; equal to either 0 (for the *remain-stationary* and *problem-solve* actions only) or 1 (see Sect. 2). For example, the action pattern `<[PT 90 1]>` states that an agent should face east (`90`) and perform the *push-tile* action (`PT`), pushing a tile and moving itself 1 square (`1`) in this direction. There are a total of 18 actions that can be performed by an agent: 4 variations of 4 actions plus 2 actions with no variations.

Short and Long-Term Memory. STM consists of three fixed-length, first-in-first-out lists, one for each modality supported in CHREST. These lists store chunks that have been retrieved from LTM.

[4] See [28] for a detailed comparison of ACT-R and CHREST's LTM implementation.
[5] South and west are represented by negative numbers.

Table 1. Mappings of identifiers for action item-on-square patterns, the action identified and variations thereof.

Identifier	**Action**	**Variations**
MR	Move-randomly	North, east, south, west
MAT	Move-around-tile	
MTT	Move-to-tile	
PT	Push-tile	
RS	Remain-stationary	N/A
PS	Problem-solve	

LTM is composed of a discrimination network that acts as a retrieval device and a similarity function; it is analogous to the hidden layers of a connectionist network, or the RETE network of Soar [25]. Nodes in LTM are connected using *test links*; to learn or retrieve information from LTM, a pattern, ϕ, is input to LTM and the discrimination network is traversed by sorting ϕ from the general root node of LTM to the corresponding modality root node and then along matching test links until a leaf node is reached or no further test links match. If ϕ does not match the chunk retrieved after traversal, θ, the network is modified using one of two learning mechanisms: *discrimination* or *familiarisation*. Implementation details of these mechanisms that are of interest to our simulations are discussed in the following section.[6]

Discrimination, Familiarisation and Production Creation. These mechanisms take a user-defined period of time; performing one of blocks performance of another. Hence, the model of cognition implemented is very human-like: learning is slow when an agent is first placed in an environment but accelerates as interaction occurs.

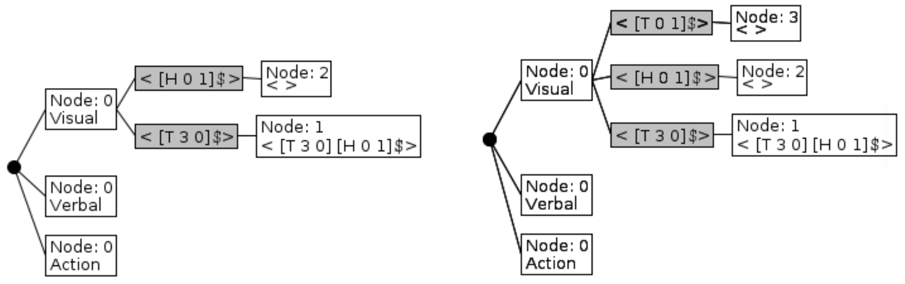
Discrimination. Increases the number of chunks stored in LTM and occurs when a pattern input to LTM, ρ (this may be a sub-pattern of a larger pattern, ϕ), is either not present in LTM or is present in LTM but the chunk retrieved, θ, does not contain ρ, and θ is “finished”: no new patterns can be appended to it (indicated by a dollar, \$ sign). In the first case, a new node is created and connected to the relevant modality root node (see Fig. 2(b)). In the second case, a new node is created and connected to θ (see Fig. 2(c)). The connection created is a test-link that contains ρ.

Familiarisation. Increases the size of a chunk and occurs when a chunk, θ, is retrieved from LTM given a pattern, ϕ, as input and the following conditions are all true:

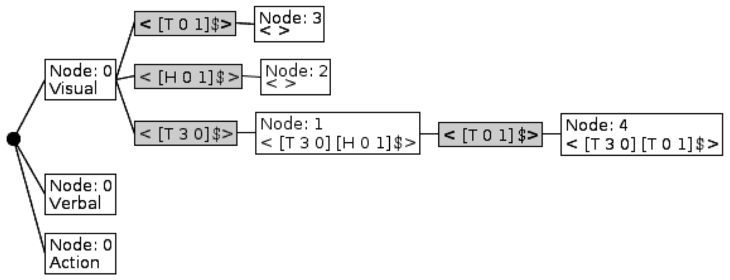
- ϕ contains a sub-pattern, ρ, that exists as a chunk in LTM.
- ρ is not present in θ.
- Sub-patterns preceding ρ in θ and ϕ are the same.
- θ is not “finished”.

If the above conditions are all true, CHREST adds ρ to θ (see Fig. 3).

[6] See [18, 26] for further details of these mechanisms.



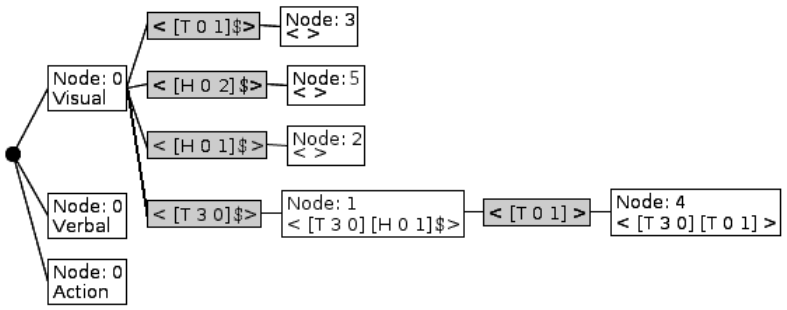
(a) LTM before presentation of pattern. (b) LTM after first presentation of pattern.



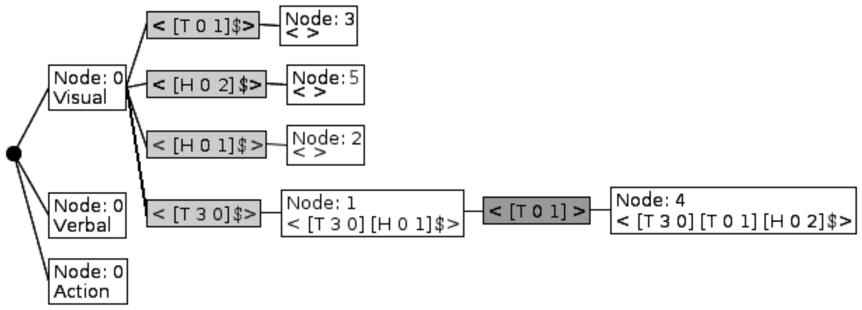
(c) LTM after second presentation of pattern.

Fig. 2. Discriminating visual item-on-square pattern `<[T 3 0][T 0 1]$>` that indicates locations of two tiles relative to an agent's location: first is 3 squares north, second is 1 square east. Test links are indicated by grey rectangles.

Production Creation. Productions are created from information contained in *episodes* (see Sect. 3.4) and implemented using hash map data structures contained in visual LTM nodes; keys contain pointers to action LTM nodes and values denote the production's utility rating, as defined in the introduction to Sect. 3 and illustrated in Fig. 4. Attempts to create productions that already exist are ignored.
For a production to be created, the visual and action parts of the production must be capable of being *recognised*, i.e. a chunk must be returned when each part is input to LTM. Due to CHREST's intention to simulate human cognition as closely as possible, it may be that, after passing the visual part of a production, ϕ, as input to LTM, the recognised visual chunk, θ, does not match ϕ exactly. Instead, θ may only contain *some* sub-patterns common to itself and ϕ. For example, if the visual pattern `<[T 1 0][H 2 0]$>` is passed to LTM as input, `<[T 1 0]>` may be retrieved if `<[T 1 0][H 2 0]$>` has not been fully familiarised. Thus, *over-generalisation* of production selection may occur, a very human-like cognition trait [24].
In these simulations, two broad types of production can exist in LTM and are differentiated by the type of action node they terminate with. The first production type terminates at an explicit action node, for example: *push-tile north*, the second terminates at an action node that



(a) LTM before presentation of pattern.



(b) LTM after presentation of pattern.

Fig. 3. Familiarising visual item-on-square pattern `<[T 3 0][T 0 1][H 0 2]$>` that indicates locations of two tiles and one hole relative to an agent's location: first tile is 3 squares north, second is 1 square east and hole is 2 squares east.

prescribes use of the problem-solving system. Differences in how these two production types are handled embody the three types of decision-making mentioned in Sect. 1 and create the three agent types discussed in Sect. 4.1.

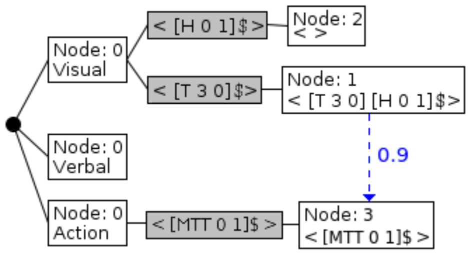


Fig. 4. An example production with an utility rating of 0.9. The production reads as: "If I can see a tile 3 squares north and a hole 1 square east then, I should move north to the tile."

3.2 Problem-Solving System

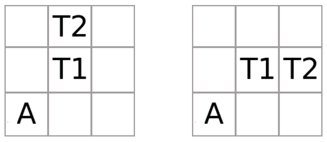
Actions generated by the problem-solving system are intended to achieve the agent's currently active goal; these are not explicitly represented in any data structure available to the agent but are ostensibly activated after analysing visual input from the environment. The result of this analysis is used to run one of three hand-coded procedures: **move randomly**, **secure tile** or **push tile to hole**. Note that we have conflated the concepts of *goal* and *environment state* since these have a simple one-to-one mapping in the environment modelled.
There are three sub-goals that need to be achieved to fulfil the agent's main goal of `fill hole with tile`. These are: `find tile`, `secure tile` and `find hole`. The problem-solving system therefore follows the procedure outlined below. Active goals are highlighted using `fixed-width` font, procedures run are highlighted in **bold** and actions generated are highlighted in *italics*. Note that "adjacent" is defined as an object being one square north, east, south or west of the object referred to.

1. Agent is surrounded, i.e. all adjacent squares are occupied by non-movable tiles, holes or other agents: *remain stationary* generated.
2. Agent is not surrounded.
   – Tiles and holes can be seen: determine closest hole to the agent, H, and tile that is closest to H, T.
      - T is adjacent to agent and can be pushed closer to H from agent's current position: `fill hole with tile` activated, **push tile to hole** run, *push tile* generated.
      - T is adjacent to agent but cannot be pushed closer to H from agent's current position: `secure tile` activated, **secure tile** run, *move around tile* generated.
      - T is not adjacent to agent: `secure tile` activated, **secure tile** run, *move to tile* generated.

   – Tiles can be seen but no holes: determine distance of agent from closest tile, T.
      - T is adjacent to agent: `find hole` activated, **push tile to hole** run, *push tile* generated.
      - T is not adjacent to agent: `secure tile` activated, **secure tile** run, *move to tile* generated.

   – No tiles can be seen: `find tile` activated, **move randomly** run, *move randomly* generated.

Note that some procedures generate actions non-deterministically in some circumstances; consider the environment states in Fig. 5. The active goal of agent A in both states is `secure tile`, specifically, tile $T1$, so it runs the **secure tile** procedure to generate an action to try and achieve this goal. The optimal action in the case of Fig. 5(a) is for A to move north around $T1$ so that it is able to push $T1$ to the east thus securing it. However, this action is non-optimal if the environment state in Fig. 5(b) is considered since A cannot push $T1$ east because $T2$ blocks $T1$ along this heading. Consequently, the optimal action in one state may be the non-optimal action in a similar state. So, in this case, the `secure tile` procedure has a 0.5 probability of generating either a *move around tile north* or a *move around tile east* action.



(a) State 1. (b) State 2.

Fig. 5. Environment state examples to justify non-determinism of action production by problem-solving procedures.

3.3 Pattern-Recognition System

The pattern-recognition system's operation is intended to be analogous to habitual behaviour in human beings: behaviours become habitual when they are frequently selected in response to particular goals being activated [1]. The system therefore uses visual patterns and production utility ratings as input to propose actions to perform. Other than the fact that the pattern-recognition system is considered to be a part of CHREST whereas the problem-solving system is not (see introduction to Sect. 3), there are two crucial differences between this system and the problem-solving system:

1. The pattern-recognition system cannot generate novel actions, it can only select actions contained in existing productions.
2. The pattern-recognition system may have to choose an action from many potential productions, depending upon how many productions exist for the visual pattern input to LTM.

After inputting a visual pattern, ϕ, to the pattern-recognition system, an attempt is made to recognise and retrieve productions from LTM, Ψ. If no productions are retrieved, execution of the system halts and decision-making control is passed to the problem-solving system. Otherwise, the utility ratings of Ψ are used as input to the Roulette selection algorithm (see Sect. 3.6) to select an action for execution.

If a production is selected by the pattern-recognition system, its action is not passed as input to CHREST to be learned if the action is performed successfully (see Sect. 4.3). This is because the action must have been learned for CHREST to have created the production (see Sect. 3.1: *Discrimination, Familiarisation and Production Creation*). Therefore, further learning of the action is redundant.

3.4 Episodic Memory

The episodic memory structure used by agents is analogous to STM and is therefore implemented as a fixed-length first-in-first-out list. An episode contains four pieces of information: a visual pattern, υ, an action pattern, α (executed by the agent in response to υ), the time α was executed (required by PSDR, see Sect. 3.5) and whether α was produced by the problem-solving or pattern-recognition system (enables type 3 agents to modify productions correctly, see Sect. 4.3).

3.5 Profit Sharing with Discount Rate

PSDR uses a *credit assignment function* (1) to calculate production utility ratings, P_σ. For example: at time t, an agent executes an action in response to the current visual environment state generating a episode, E_t. At time $t + 1$, the agent executes another action in response to the current visual environment state, producing another episode, E_{t+1}, and continues this cycle until it receives a reward, R, at time T. At time T, the agent's episodic memory will contain the following episodes if the number of episodes from E_t to E_T is less than, or equal to, the maximum size of episodic memory: $(E_t, E_{t+1} \ldots E_T)$. With $R = 1$ and discount rate $\beta = 0.5$, the production corresponding to episode E_T receives 1 as credit, E_{T-1}'s production receives 0.5, E_{T-2}'s production' receives 0.25 etc. The credit generated for a production is then added to that production's current utility rating.

$$P_\sigma = P_\sigma + (R \cdot \beta^{T-t}) \ \ (0 < \beta < 1) \tag{1}$$

PSDR [4] was chosen as a reinforcement learning theory for three reasons: first, it can be used in domains where mathematical modelling of the domain is intractable, a property of the version of Tileworld implemented (see Sect. 2). Second, PSDR's production utility rating mechanism is congruent with that discussed earlier (see introduction to Sect. 3) since it uses discounted rewards. Third, PSDR's effectiveness in enabling agents to learn and apply robust, effective productions autonomously in dynamic, multi-agent domains that are similar to the version of Tileworld used in these simulations has been validated by others [3, 4].

3.6 Roulette Algorithm

The Roulette algorithm [5] uses production utility ratings to select an action for execution given a number of candidate productions. Equation (2), generates a value, ω, for a candidate production, P, from P's utility rating, P_σ, divided by the sum of each candidate production's utility rating, P^n_σ to P^N_σ. Candidate productions are then organised in ascending order according to their ω value and used to create a range of values; candidate productions with greater ω values occupy greater ranges. Finally, a random number, $0 < R < 1$, is generated and used to select a production from the candidates. Therefore, the algorithm is non-deterministic and abides by the principle of rationality defined in the introduction to Sect. 3: candidate productions with greater utility ratings will be more likely to be selected whereas it is less likely for candidate productions with lower utility ratings to be selected.

$$\omega = P_\sigma / \sum_{n=1}^{N} P^n_\sigma \tag{2}$$

4 Agent Implementation

Agents are equipped with the cognitive architecture described in Sect. 3 and a domain-specific input/output component. Agents are goal-driven, non-communicative, non-cooperative, and have limited vision. The size of an agent's observable environment is controlled by a parameter that takes a number as input to indicate how many squares north, east, south and west the agent can "see". We keep the value of this parameter constant at 2 since agent performance should not be affected by differences in "physical" capabilities. The size of visual patterns generated is dependent on this parameter, so any visual pattern constructed can only contain 24 item-on-square patterns at most.[7] This is important since larger values may result in the agent constantly discriminating and familiarising due to large input patterns and thus blocking production creation (see Sect. 3.1). Setting the sight parameter to 2 is also the minimum value that allows the agent to see "around" a tile so that its ability to be pushed can be determined (important to enable valid solutions to be provided by the problem-solving system).

Fig. 6 illustrates the cognitive architecture structures discussed in Sect. 3, the agent-specific components discussed in this section and how information flows between them. Note that the sequencing illustrated in Fig. 6 does not always apply due to the agent's type and how the problem-solving and pattern-recognition systems operate. These details, along with the sequencing changes mentioned, are delineated in Sect. 4.1.

This section proceeds as follows: we discuss the implementation details of the three agent types that embody the three different types of decision-making outlined in Sect. 1 in Sect. 4.1. Operation of the agent-specific input/output component is outlined in Sect. 4.2 and the execution cycle for agents is provided in Sect. 4.3.

4.1 Agent Types

In Sect. 3.1: *Discrimination, Familiarisation and Production Creation*, we delineated two types of productions that can be created by agents: productions terminating with explicit actions, i.e. *push-tile north*, and those terminating with the action that prescribes usage of the problem-solving system. The three types of agents implemented in these simulations are defined by the types of production they can create:

- Agent Type 1 (pure problem-solver): neither type of production are created in LTM; agents of this type will always use problem-solving to decide upon what action to perform next.
- Agent Type 2 ("pure" pattern-recogniser): only creates productions that terminate with explicit action chunks.
- Agent Type 3 (problem-solver and pattern-recogniser): creates both types of production.

Since agent type 1 does not use the pattern-recognition system at all, sequences 3-6 and 11-13 do not occur in Fig. 6.

Agent type 2 uses problem-solving to generate actions initially but after a production, P, has been constructed and rated, problem-solving will no longer be used to generate an action when P's visual condition is

[7] A square in Tileworld can only contain one item (see Sect. 2), one item-on-square pattern encodes one item and the agent doesn't encode its own location.

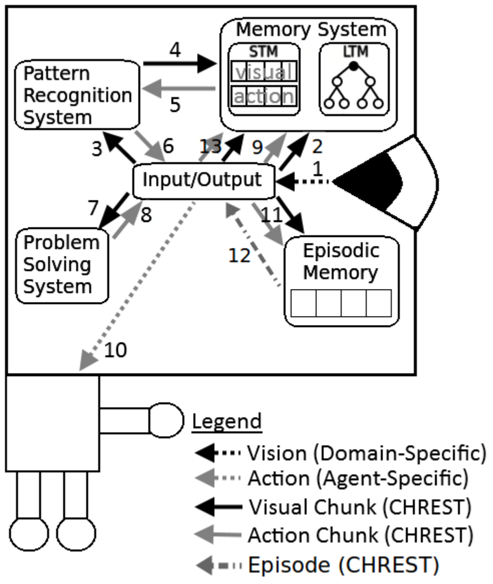


Fig. 6. Agent architecture and flow of information throughout (numbers denote sequence of information processing and are discussed in Sect. 4.1).

encountered (pattern-recognition is always used before problem-solving by agents, see introduction to Sect. 3). Hence, agents of type 2 are not "pure" pattern-recognisers (hence the quotes around "pure") in the same sense as agents of type 1 are pure problem-solvers since problem-solving is still used to some degree.

Agent type 3 strikes more of a balance between problem-solving and pattern-recognition system use than agent type 2. The problem-solving system will be used more in initial decision-making (as it is for agent type 2) but, as LTM develops, it may be that productions generated result in either an explicit action being performed or the problem-solving system being used to generate a potentially novel and better action.

With regard to agent types 2 and 3 and Fig. 6, the sequencing illustrated can be altered in the following ways depending on the outcome of the pattern-recognition system:

– Agent types 2 and 3.
    - If pattern-recognition does not propose an action, sequences 5 and 6 not not occur.
    - If the pattern-recognition system proposes an action, sequences 7-9 do not occur (see Sect. 3.3 for an explanation of why 9 does not occur).
– Agent type 3.
    - If the pattern-recognition system proposes an action that prescribes usage of the problem-solving system, sequence 9 does not occur.

4.2 Input/Output Component

In the simulations, translation between domain-specific visual information, agent-specific action information and CHREST is required: this is provided by an agent's input/output component.
Visual patterns are produced by translating domain-specific visual information and output is sent to the agent's memory system to be learned and the pattern-recognition system, if applicable (see Sect. 4.1). The unmodified domain-specific visual information may also be sent to the problem-solving system, if applicable (again, see Sect. 4.1).
Actions produced by the problem-solving or pattern-recognition system are also passed to this component. Agent-specific actions received from the problem-solving system are converted into CHREST-compatible action patterns whereas output from the pattern-recognition system is converted from CHREST-compatible action chunks into agent-specific actions (see Sects. 3.2 and 3.3 for details). In either case, the original action information and its converted form are retained; the CHREST-compatible action pattern is sent to CHREST's memory system so it can be learned whilst the agent-specific version is executed by the agent, causing the agent to perform the action in the environment.
Episodes (see Sect. 3.4) are also encoded and decoded by an agent's input/output component. An example of an episode is as follows: `[ <[T 1 0][H 2 0]$> <[PT 0 1]$> 110 true ]`. This episode should be interpreted thus: an agent saw a tile 1 square to the north and a hole 2 squares to the north (`<[T 1 0][H 2 0]$>`) and used its problem-solving system (`true`) to generate a *push tile north* action (`<[PT 0 1]$>`) that was executed at time 110 (`110`). When decoding an episode, the visual and action pattern are retrieved and sent to the agent's memory system to enable production creation or modification (see Sect. 4.3).

4.3 Execution Cycle

The agent execution cycle runs for each agent in turn after every time increment in the Tileworld environment. The order of agent execution is randomised so, at time t, agent 0 may execute first then agent 1 whereas at time $t + 1$, agent 1 may execute before agent 0.
Note that agents have a specific *intention reconsideration* strategy implemented: when the current time, T, equals the time that an action, α, is to be performed, t, the agent generates a new visual pattern, χ, and compares this to the visual pattern, ϕ, used to generate α. If $\phi \neq \chi$, the agent does not perform α and instead generates and loads a new action for execution based upon the information in χ.
The execution cycle proceeds as follows. Note that some steps in the execution cycle require additional explanation to justify their inclusion, these explanations follow the execution cycle delineation. Each agent begins by checking to see if there is an action loaded for execution:

1. No action loaded for execution.
   (a) Generate visual pattern, ϕ.
   (b) Pass ϕ as input to LTM and attempt to learn.

   (c) Use ϕ to generate a new action pattern, α, using problem-solving or pattern-recognition system, depending upon agent type.
   (d) Load α for execution, pass α as input to LTM and attempt to learn. Set time for execution of α to t and exit execution cycle.
2. Action α is loaded for execution, check to see if current time, T, equals t.
   (a) $T = t$: generate new visual pattern, χ, and compare this to ϕ.
      i. $\phi = \chi$: attempt to perform α.
         A. Agent successfully performs α. Check agent type.
            – Agent type 1: exit execution cycle.
            – Agent type 2/3:
              If α is not a *move-randomly* action, create new episode in episodic memory and attempt to create a production in LTM between ϕ and α.
              If α achieves agent's primary goal, apply PSDR to productions representing episodes in episodic memory and clear episodic memory.
         B. Agent unsuccessfully performs α: exit execution cycle.
      ii. $\phi \neq \chi$: unload α for execution and exit execution cycle.
   (b) $T \neq t$: exit execution cycle.

The refusal to create a new episode and production using ϕ and α when α is a *move-randomly* action is due to two reasons. First, if such productions were created, agents would be biased in their random movement so, if the agent selects productions like this in future, the movement performed is not truly random. Second, given the stochastic nature of Tileworld, biasing random movement could artificially influence an agent's performance. For example, it may be that tiles and holes are generated, by chance, to the west of an agent more frequently. If this agent's random movement were biased to the east, it would encounter resources less frequently, reducing its ability to score and impinging its performance.

To justify why agents create/modify productions after an action is successfully performed rather than before, we appeal to the definition of rationality provided in the introduction to Sect. 3: less useful productions should be more strongly suppressed than more useful ones. Actions are only ever not performed because the decision-making system has failed to take into account some environmental resource that stops the action being performed successfully (trying to push a tile along a heading when there is another tile in the way). Therefore, productions that are completely useless are never created.

On this note, it is worthwhile to explain under what circumstances an agent may fail to perform an action since it appears that the intention reconsideration strategy implemented should prevent such an event. Essentially, this can only occur for agents that use pattern-recognition. For example, an agent may attempt to perform a *push-tile* action on tile $T1$ after using pattern-recognition. However, due to over-generalisation of production selection (see Sect. 3.1: *Discrimination, Familiarisation and Production Creation*), it may be that the agent fails to consider a tile present in the visual input that blocks $T1$ from being pushed in the direction suggested. Thus, when the agent attempts to push $T1$, $T1$ is blocked and the action fails.

When creating/modifying production utility ratings after an action is performed, special consideration must be afforded to type 3 agents. Since these agents can create/modify productions whose actions propose using an explicit action or the problem-solving system, they must make a choice given that production creation/modification occurs in CHREST and CHREST is not able to create/modify two productions simultaneously. So, if an episode indicates that its action was generated using problem-solving, type 3 agents create or select the type of production to create/modify randomly. This is implemented by the agent generating a random float R, $(0 <= R < 1)$. If $R < 0.5$ a/the production prescribing use of the problem-solving system given the visual part of the episode is created/modified. Otherwise, a/the production prescribing use of the explicit action performed given the visual part of the episode is created/modified.

5 Simulation Details

Two sets of simulations were run. The first explores what balance of problem-solving and pattern-recognition system use maximises agent performance given different environmental complexities and how differing environment complexities affect pattern-recognition and problem-solving use. The second comprises a sensitivity analysis of episodic memory size and discount rate for agent types 2 and 3 and allows us to ascertain if significantly altering these values affects the dependent variables outlined for the first set of simulations. Hence, this section is split in two: Sect. 5.1 details the first set of simulations, Sect. 5.2 outlines the second.

5.1 Decision-Making System Use, Environment Complexity and Performance Details

For this set of simulations, 27 conditions were simulated and run. Conditions are representative of various degrees of intrinsic/extrinsic environmental complexity and agent types (see Sect. 4.1). Each condition was repeated 10 times to harvest a data set large enough to provide a robust analysis. For each repeat, average frequencies of problem-solving and pattern-recognition system use were recorded along with the average score for all agents (to determine performance). The overall values for these dependent variables over each condition were then calculated by averaging the averages obtained for the repeats.
Our null hypotheses state that:

- Using problem-solving instead of pattern-recognition and vice-versa does not have any significant effect on the performance of agents.
- Altering extrinsic and intrinsic environment complexity does not have any significant effect upon problem-solving or pattern-recognition use.
- Altering extrinsic and intrinsic environment complexity does not have any significant effect upon the performance of agents.

Intrinsic environment complexity is controlled by the values of the *hole appearance probability*, *tile appearance probability*, *hole lifespan* and *tile lifespan* parameters. Greater *tile/hole appearance probability* values and smaller *tile/hole lifespan* values equate to greater complexity; more tiles/holes appear but for shorter periods of time resulting in a greater number of novel environment states occurring. One may expect the values for the *tile/hole appearance interval* parameters to also be varied. However, the intrinsic complexity of the environment can be significantly modified by varying the values of the parameters mentioned. Values for the *tile/hole appearance probability* parameters were derived by simply taking the median probability, 0.5, as the moderate complexity value and then taking the lowest/highest values possible without guaranteeing tile/hole appearance since this would significantly skew the results. The *tile/hole appearance probability* and *tile/hole lifespan* parameter value mappings for each level of environment complexity are provided below:

- Environment complexity: low
  - Tile/hole appearance probability: 0.1
  - Tile/hole lifespan: 80 seconds
- Environment complexity: moderate
  - Tile/hole appearance probability: 0.5
  - Tile/hole lifespan: 40 seconds
- Environment complexity: high
  - Tile/hole appearance probability: 0.9
  - Tile/hole lifespan: 20 seconds

Extrinsic environmental complexity is determined by the number of agents in the environment (see Sect. 2 for justification). We set this variable to either 2, 4 or 8.

All other variable values are kept constant (see Table 2). There are three major groups of conditions differentiated by the degree of intrinsic environment complexity used. These major groups then consist of a further three sub-groups of conditions differentiated by the degree of extrinsic environmental complexity. Finally, each sub-group consists of three sub-sub-groups differentiated by agent type (see Sect. 4.1). Note that the agent types used in a condition are homogeneous.

5.2 Sensitivity Analysis

In this set of simulations, 270 conditions were simulated and run and each condition was repeated 10 times to harvest a data set large enough to provide a robust analysis. The dependent variables recorded and methods used to calculate their values remain unchanged from the simulations detailed in Sect. 5.1.

Our null hypotheses states that there is no significant effect upon agent performance or pattern-recognition/problem-solving system use when episodic memory size or discount rates are altered.

Note that in this set of simulations, we have conflated extrinsic and intrinsic environment complexity into a generic environment complexity to make the analysis simpler. The *tile/hole appearance probability* and *tile/hole lifespan* parameter value mappings for each level of environment complexity are provided below:

Table 2. Mappings of independent variable names to owner (agent, CHREST, environment), value used and justification for value used.

Independent Variable	**Owner**	**Value**	**Justification**
Problem-solving time	Agent	1 sec	Equals value of the tile/hole birth interval parameters so planned actions may be reconsidered due to the appearance of a new tile or hole.
Sight radius	Agent	2	See Sect. 4.
Add link time	CHREST	10 sec	Taken from [38].
Discount rate	CHREST	0.1 to 0.9	Kept constant at median value (0.5) for simulations detailed in 5.1 and 6.1, varied in simulations detailed in 5.2 and 6.2.
Discrimination time	CHREST	10 sec	Taken from [38].
Episodic memory size	CHREST	6 to 14	Kept constant at median value (10) for simulations detailed in 5.1 and 6.1, varied in simulations detailed in 5.2 and 6.2.
Familiarisation time	CHREST	2 sec	Taken from [38].
Pattern-recognition time	CHREST	0.2 sec	Taken from [17].
Hole appearance interval	Env.	1 sec	Equals value of the problem-solving time parameter so planned actions may be reconsidered due to the appearance of a new tile or hole.
Play time	Env.	28800 sec	Allows pattern-recognition systems to learn enough information to be useful.
Reward value	Env.	1	Equal to the single point received for an agent achieving its main goal of pushing a tile into a hole.
Tile appearance interval	Env.	1 sec	Equals value of the problem-solving time parameter so planned actions may be reconsidered due to the appearance of a new tile or hole.
Time increment	Env.	0.1 sec	CHREST operations measured in milliseconds: this is the smallest major time unit possible.

- Environment complexity: low
  - Tile/hole birth probability: 0.1
  - Tile/hole lifespan: 80 seconds
  - Number of agents: 2
- Environment complexity: moderate
  - Tile/hole birth probability: 0.5
  - Tile/hole lifespan: 40 seconds
  - Number of agents: 4
- Environment complexity: high
  - Tile/hole birth probability: 0.9
  - Tile/hole lifespan: 20 seconds
  - Number of agents: 8

We take the values used for the discount rate and episodic memory size in the simulations detailed in Sect. 5.1 as median values for these variables in this set of simulations. Discount rate is then incremented in steps of 0.1 from 0.1 to 0.9 and episodic memory size is incremented in steps of 2 from 6 to 14 for each agent type in each of the environment complexities defined. All other variable values are kept constant (see Table 2).

There are three major groups of conditions differentiated by the degree of environment complexity used. These major groups then consist of a further two groups of conditions differentiated by agent type (agent type in each condition is homogeneous). Each of these agent type groups then consists of five groups determined by size of episodic memory. Finally, each of these episodic memory size groups is divided into nine groups distinguished by the discount rate used. Thus, all combinations of environment complexity, agent type, episodic memory size and discount rates outlined for this set of simulations are tested.

6 Results and Discussion

This section is split into two, with Sect. 6.1 discussing results from the simulations described in Sect. 5.1 and Sect. 6.2 discussing results from the simulations described in Sect. 5.2.

6.1 Decision-Making System Use, Performance and Environment Complexity Results

Results in this section were analysed using a $3 \times 3 \times 3$ analysis of variance (ANOVA), with environment (intrinsic) complexity, number of agents (extrinsic complexity) and agent type as between-subject variables.[8] As mentioned in Sect. 5.1, we have collected data for three dependent variables: average score, average frequency of problem-solving system use and average frequency of pattern-recognition system use. This section is in two parts: the first covers how the between-subject variables affect decision-making system use, and the second looks at how the between-subject variables affect agent performance.

[8] Due to an error in the simulation code used in a previous version of this paper [27], the results reported in this section consistently differ from those reported in the corresponding section of [27] by a factor of 10. The results reported in this section use a rectified version of the simulation code and are correct.

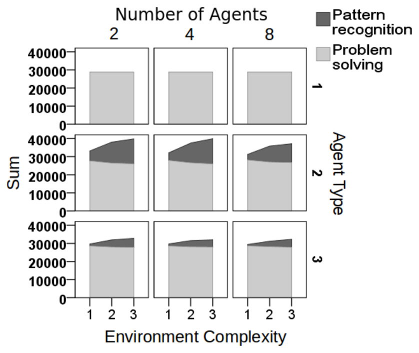


Fig. 7. Average frequency of problem-solving and pattern-recognition system use by agents as a function of agent type and each intrinsic/extrinsic complexity setting outlined in Sect. 5.1.

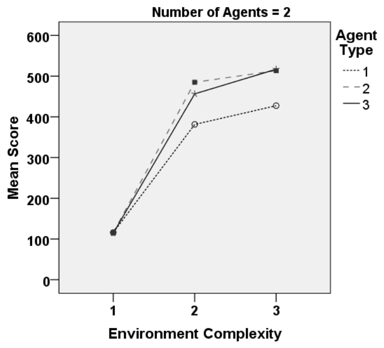


Fig. 8. Average performance of agents for each intrinsic complexity setting outlined in Sect. 5.1 when 2 agents are present in Tileworld.

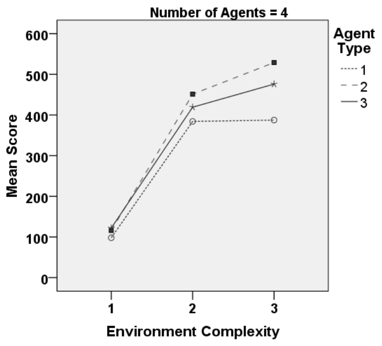


Fig. 9. Average performance of agents for each intrinsic complexity setting outlined in Sect. 5.1 when 4 agents are present in Tileworld.

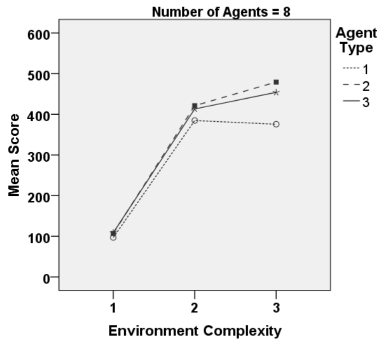


Fig. 10. Average performance of agents for each intrinsic complexity setting outlined in Sect. 5.1 when 8 agents are present in Tileworld.

Decision-Making System Use. The average total amounts of problem-solving and pattern-recognition system use are shown in Fig. 7. The stacked segments in the figure indicate the proportion of the total taken up by each of the two kinds of decisions: in all cases (for agents of type 2 and 3), the amount of pattern-recognition was significantly smaller than the amount of problem solving.
The results indicated a main effect of intrinsic complexity, $F(2, 243) = 295.2$, extrinsic complexity, $F(2, 243) = 24.0$, and agent type, $F(2, 243) = 1,511.7$ with all $p < 0.001$. As expected, type 1 agents never used pattern-recognition whilst type 2 agents used pattern recognition more frequently than agents of type 3. By increasing extrinsic complexity, agents of type 2 and 3 decreased their use of pattern-recognition; in contrast, increasing intrinsic complexity had the reverse effect. The following interactions were also statistically significant for agent types 2 and 3: intrinsic complexity and agent type $F(4, 243) = 113.6$, $p < 0.001$, extrinsic complexity and agent type: $F(4, 243) = 15.0$, $p < 0.001$.
Average frequency of problem-solving use yielded results that were the mirror-image of those obtained for average frequency of pattern-recognition use. This is expected since, if an agent does not use pattern-recognition then it will use problem-solving. There was a main effect of agent type, $F(2, 243) = 1,512.1$, $p < 0.001$, with type 1 agents using problem-solving most frequently on average and type 2 agents least (as expected). A main effect of intrinsic complexity, $F(2, 243) = 295.2$, $p < 0.001$, reflects that average frequency of problem-solving use tended to decrease with increasing intrinsic complexity. Finally, the main effect of extrinsic complexity was significant, $F(2, 243) = 24.0$, $p < 0.001$, reflecting a small increase in problem-solving use as extrinsic complexity increased.

Performance. Figures 8, 9 and 10 show average scores achieved by each agent type for each degree of environment complexity organised by number of agents. The three main effects were statistically significant: intrinsic complexity, $F(2, 243) = 2,437.7$, extrinsic complexity, $F(2, 243) = 16.6$, and agent type, $F(2, 243) = 70.8$ with all $p < 0.001$.
Irrespective of environment complexity, the average score achieved by agents of type 2 was either approximately equal to, or greater than, the average score of type 3 agents. Type 1 agents consistently achieved the lowest average scores. Neglecting average scores achieved when intrinsic environment complexity is low (since there is not much difference between the agent types), agents of type 2 offered a performance increase of up to 36.5% when compared to agents of type 1 (see Table 3).
By increasing extrinsic complexity, average scores were decreased for each agent type whilst increasing intrinsic complexity caused average scores to increase for each agent type. The only exception to this trend was the average score obtained by type 1 agents when there were 4 or 8 players in the environment and intrinsic complexity was increased from moderate to high. In these conditions, the average score for type 1 agents either remained equal or decreased. This complex pattern of results is reflected by a statistically significant interaction between environment complexity and agent type, $F(4, 243) = 15.3$ with $p < 0.001$.

Table 3. Percentage difference between average scores achieved by agent types 1 and 3 in the moderate and highly complex environment conditions outlined in Sect. 5.1.

Complexity		**Agent Type**		
Intrinsic	*Extrinsic*	*1*	*3*	**% Increase**
Moderate	Low	381.5	485.0	27.13
Moderate	Moderate	384.3	451.3	17.43
Moderate	High	384.8	421.3	9.49
High	Low	427.5	513.5	20.12
High	Moderate	387.5	529.0	36.52
High	High	375.6	479.4	27.64

Discussion.

Decision-Making System Use. We will first consider the effects of increasing extrinsic and intrinsic complexity on decision-making system use. By increasing extrinsic complexity, competition for environmental resources increases. Thus, there are fewer interactions between agents and environmental resources, so fewer productions are created (productions are only ever created when an action is performed successfully, see Sect. 4.3), pattern-recognition system use decreases (since the system does not have the resources required to operate) and problem-solving system use increases. By increasing intrinsic complexity, availability of environmental resources increases. This results in agents interacting with environmental resources more frequently, in turn increasing the number of productions created in LTM. Consequently, the opposite effect to increasing extrinsic environmental complexity is observed: pattern-recognition system use increases and problem-solving system use decreases.

Conversely, *a priori* reasoning would suggest that, when intrinsic environment complexity is increased, the state space of the environment increases: this should result in agents discriminating/familiarising more often, inhibiting the creation of productions (see Sect. 3.1: *Discrimination, Familiarisation and Production Creation.*). Consequently, pattern-recognition use should decrease due to the unavailability of productions. However, when intrinsic complexity is high, it is likely that an agent will encounter similar environment states frequently due to an over-abundance of environmental resources. This, coupled with the fact that the observable space of the environment is relatively small for an agent compared to the total size of the Tileworld environment (25 squares against 1225), means that the number of completely familiarised visual and action chunks in an agent's LTM will increase, facilitating production creation. Thus, it is more likely for an agent that can use pattern-recognition to do so.

Explaining the effect of agent type on decision-making system use is trivial: type 1 agents only use problem-solving, type 2 agents use their pattern-recognition system more frequently than type 3 agents and type 3 agents use problem-solving more frequently than type 2 agents. For agent types 2 and 3: when a production is created for a visual state, type 2 agents will never use their problem-solving system again when that visual state is encountered whereas type 3 agents may do.

Performance. Since increasing extrinsic complexity elevates competition for resources (see previous section), this should result in performance declining as extrinsic complexity increases; this is observed in most of the data acquired (see Figs 8-10). Increasing intrinsic complexity has the opposite effect since there are more resources available to an agent to achieve their primary goal. Average scores are likely to increase as intrinsic complexity increases. Again, this is observed in most of the data acquired (see Figs 8-10).
To explain the effect of agent type on performance, we must consider the length of time taken to execute an action using either decision-making system against the regularity by which the environment's state may change. When an agent executes an action, it first checks to see if its observable environment has changed since it started to deliberate on the action that is to be performed (see Sect. 4.3). This intention reconsideration strategy means that, in some cases, an agent will not perform an action potentially resulting in the relevant environment resources expiring before the agent can use them to achieve its current goal.
Since the interval of time for an agent generating an action using problem-solving and the environment potentially creating new tiles and holes is equal (1 second), an agent's intention reconsideration is more likely to be triggered when the problem-solving system is used and when extrinsic and intrinsic environment complexity is increased. Therefore, in the space of time where the environment remains static, agents that use pattern-recognition can perform up to 5 actions. This enables agents that employ pattern-recognition more to achieve their goals more quickly and score more frequently. This explains why type 2 agents consistently perform better compared to agents of type 1 and 3 who use pattern-recognition less frequently.

Conclusions. All null hypotheses stated in Sect. 5.1 are refuted. Crucially, the results reported in Figs. 7-10 indicate that increased use of the pattern-recognition system by agents benefits performance since more actions can be performed before the environment state changes due to intrinsic environmental factors (extrinsic factors may still cause state changes, however). This results in an agent's intention reconsideration strategy being triggered less frequently so the agent acts more frequently to achieve its goals than it does deliberating about how to achieve them. This supports the position that acting quickly and, potentially, sub-optimally in complex, stochastic environments benefits performance more than taking time to re-evaluate productions to potentially optimise them. In this sense, the Einstellung ~~E~~ffect appears to be beneficial for agents in the environment modelled. This conclusion is further bolstered by two, seemingly anomalous, observations.
The first observation is illustrated in Fig. 9: performance for type 2 agents improves when intrinsic complexity is high and extrinsic complexity increases from low to moderate. As explained in the *Decision-Making System Use* and *Performance* discussions above, increasing ex-

trinsic complexity generally impairs performance whilst increasing intrinsic complexity improves performance. The simulation condition outlined therefore appears to strike a balance between these interactions so that the state-space is “just-right” to optimise pattern-recognition use:

– Agents are not overloaded with visual information causing excessive discrimination/familiarisation and blocking production creation.
– Availability of environmental resources is such that the agent is not blocked from moving/pushing tiles (due to too many resources) and the agent does not spend most of its time looking for resources (due to too few resources).

The same result is not observed for type 3 agents because use of their pattern-recognition system can entail use of problem-solving.
The second observation concerns the performance of type 1 agents when extrinsic complexity is moderate and high and intrinsic complexity increases from moderate to high (see Figs. 9 and 10). In these conditions, the performance of type 1 agents either plateaus or decreases since, as intrinsic complexity is increased, intention reconsideration will occur more frequently (as already argued). Since these agents can only use problem-solving, they will deliberate more frequently than they will act, resulting in their performance being hampered.

6.2 Sensitivity Analysis

All results in this section were analysed using a $3 \times 5 \times 9$ ANOVA, with environment complexity, episodic memory size and discount rate as between-subject variables, respectively. We focus on the following dependent variables: average score, average frequency of problem-solving system use and average frequency of pattern-recognition system use. The section is split into three parts: the first presents results concerning decision-making system use, the second presents results concerning performance and the third discusses the results and offers explanations for the observations noted.

Decision-Making System Use. Figures 11 to 14 display results relevant to this section. Note that we do not display results pertaining to the effects of discount rate on decision-making system use since the effect of this variable was, at best, marginal, and, at worst, non-significant. To calculate the results reported, an average of averages for problem-solving/pattern-recognition system-use was calculated. For example, average problem-solving system use reported for type 2 agents with episodic memory size 6 in the low complexity condition was calculated as follows:

1. Calculate the average frequency of problem-solving system use achieved by type 2 agents with episodic memory size 6 and discount rate 0.1 over the 10 repeats for the low environment complexity condition. Repeat for each discount rate.
2. Average the average frequencies from step 1; report result.

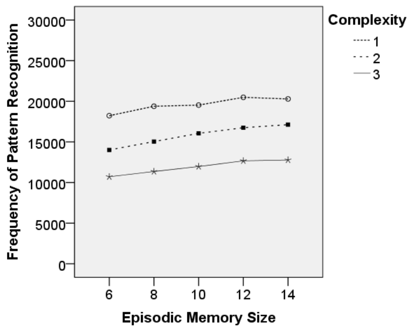


Fig. 11. Effect of episodic memory size upon the average frequency of pattern-recognition system use by type 2 agents for each environment complexity setting outlined in Sect. 5.2.

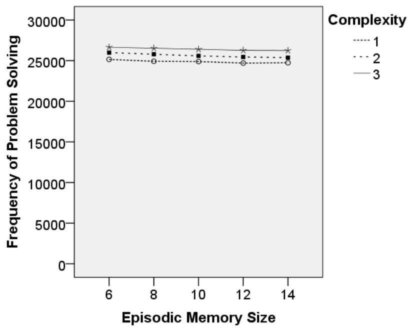


Fig. 12. Effect of episodic memory size upon the average frequency of problem-solving system use by type 2 agents for each environment complexity setting outlined in Sect. 5.2.

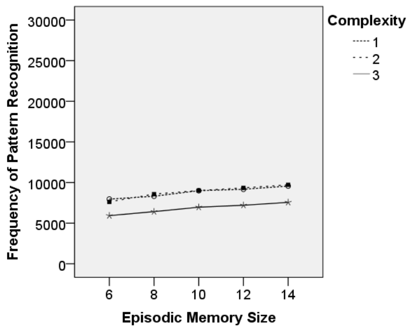


Fig. 13. Effect of episodic memory size upon the average frequency of pattern-recognition system use by type 3 agents for each environment complexity setting outlined in Sect. 5.2.

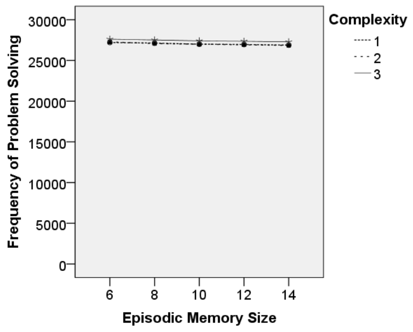


Fig. 14. Effect of episodic memory size upon the average frequency of problem-solving system use by type 3 agents for each environment complexity setting outlined in Sect. 5.2.

Agent Type 2. For average frequency of pattern-recognition system use, there was a main effect of environment complexity, $F(2, 1215) = 2018.0$, $p < 0.001$ and episodic memory size, $F(4, 1215) = 82.8$, $p < 0.001$, but no main effect of discount rate, $F(8, 1215) = 0.5$, $p = \text{ns}$. Only the interaction between environment complexity and episodic memory size was statistically significant, $F(8, 1215) = 2.1$, $p < 0.05$.
Average frequency of problem-solving system use results were a mirror-image to those for pattern-recognition system use. There was a main effect of environment complexity, $F(2, 1215) = 2017.9$, $p < 0.001$ and episodic memory size, $F(4, 1215) = 82.8$, $p < 0.001$, but no main effect of discount rate, $F(8, 1215) = 0.5$, $p = \text{ns}$. The only statistically significant interaction was that between environment complexity and episodic memory size, $F(8, 1215) = 2.1$, $p < 0.05$.

Agent Type 3. For average frequency of pattern-recognition system use, there was a main effect of environment complexity, $F(2, 1215) = 315.1$, $p < 0.001$ and episodic memory size, $F(4, 1215) = 68.1$, $p < 0.001$. Unlike type 2 agents, the main effect of discount rate was marginally significant, $F(8, 1215) = 1.9$, $p = 0.054$ as was the interaction between environment complexity and discount rate $F(16, 1215) = 1.6$, $p = 0.054$; no other interactions were present. While there was still less pattern-recognition system use on average in the high complexity condition (like type 2 agents), there was no significant difference in average frequency of pattern-recognition system use between low and moderate environment complexity conditions, $F(1, 810) = 0.3$, $p = \text{ns}$. This is an important difference between agents of type 2 and 3.
Results for average frequency of problem-solving system use were, again, a mirror image to those obtained for pattern-recognition system use.[9] There was a main effect of complexity and episodic memory size, a marginal effect of discount rate and an interaction between environment complexity and discount rate. Again, there is no significant difference in the average frequency of problem-solving system use between low and moderate environment complexity conditions.

Performance. Figures 15 and 16 display results relevant to this section. As with results concerning frequency of decision-making system use, we do not show results regarding the effects of discount rate upon performance since the effect of this variable was non-significant for both agent types. Results for this section were calculated in the same fashion as results for frequency of decision-making system use (see Sect. 6.2: *Decision-Making System Use*).

Agent Type 2. There was a main effect of environment complexity, $F(2, 1215) = 29{,}590.9, p < 0.001$ and episodic memory size, $F(4, 1215) = 11.8$, $p < 0.001$, but no main effect of discount rate, $F(8, 1215) = 0.521$, $p = \text{ns}$. There was also an interaction between environment complexity and episodic memory size, $F(8, 1215) = 671.6$, $p < 0.01$. No other interactions were statistically significant.

[9] All F and p values for the effects discussed are equal to those outlined for average frequency of pattern-recognition system use.

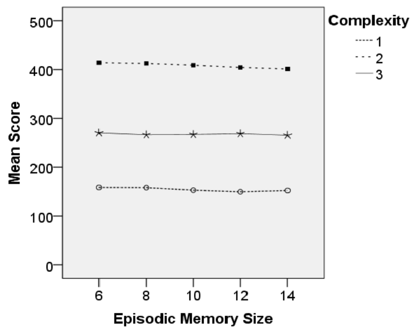


Fig. 15. Effect of episodic memory size upon the performance of type 2 agents for each environment complexity setting outlined in Sect. 5.2.

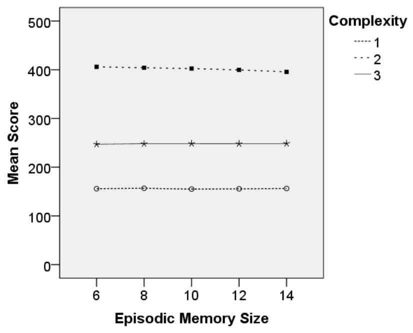


Fig. 16. Effect of episodic memory size upon the performance of type 3 agents for each environment complexity setting outlined in Sect. 5.2.

Agent Type 3. Like the performance of type 2 agents, there was a main effect of environment complexity, $F(2, 1215) = 36,627.9$, $p < 0.001$ and no main effect of discount rate, $F(8, 1215) = 1.4$, $p =$ ns. However, the main effect of episodic memory size just failed to reach statistical significance, $F(4, 1215) = 2.1$, $p = 0.074$. There was also an interaction between complexity and episodic memory size, $F(8, 1215) = 3.0$, $p < 0.005$. No other interactions reached statistical significance.

Discussion.

Decision-Making System Use. In Sect. 6.1: *Decision-Making System Use*, we noted that increasing intrinsic complexity increased pattern-recognition system use, but increasing extrinsic complexity had the opposite effect. It therefore follows that extrinsic environment complexity must have a more significant effect upon decision-making than intrinsic environment complexity. This is because intrinsic environment complexity increases along with extrinsic complexity in the complexity conditions used in this sensitivity analysis. However, frequency of pattern-recognition use decreases as complexity increases.

The observation that frequency of pattern-recognition system use significantly differed for type 2 agents, but not for type 3 agents, when environment complexity was increased from low to moderate must be due to type 3 agents reinforcing problem-solving productions more on average than explicit action productions. To explain: in the low complexity condition, environmental resource availability is low, so problem-solving is used more frequently (as already explained in Sect. 6.1: *Decision-Making System Use*). Then, as complexity is increased, pattern-recognition system use is promoted (again, as explained in Sect. 6.1: *Decision-Making System Use*). However, since type 3 agents can create productions resulting in use of their problem-solving system, if, by chance, they tend to create these productions more than ones that prescribe explicit actions, then increased use of the pattern-recognition system is cancelled out.

As expected, increasing episodic memory size increased average frequency of pattern-recognition system use and decreased average frequency of problem-solving system use, irrespective of agent type. Since increasing episodic memory size enables an agent to store more episodes, this increases the likelihood of the pattern-recognition system being employed since a greater range of visual states will be encoded as productions.

The respective effects of environment complexity and episodic memory size for type 2 agents should be noted: as environment complexity increased, average frequency of pattern-recognition system use slightly decreased. In contrast, as episodic memory size increased, average frequency of pattern-recognition system use slightly increased. Thus, environment complexity and episodic memory size appear to offset one another. However, the F values obtained indicate that environment complexity exerted a more significant effect upon decision-making system frequency than episodic memory size.

Performance. Irrespective of agent type, the best performance was found in the moderate complexity condition, with the worst performance achieved in the low complexity condition. Again, this is most probably because the balance of intrinsic and extrinsic environment complexity is optimal for performance promotion in the moderate complexity condition used in this sensitivity analysis:

– The environment does not change so much that agents spend more time deliberating about how to achieve goals than acting to achieve them, or constantly discriminating/familiarising rather than creating productions so pattern-recognition can be used.
– Intrinsic complexity is not so low that resources are scarce and primary goal achievement is impeded. Neither is it so high that resources can not be moved due to a lack of empty squares.
– Extrinsic complexity is not so high that it is difficult to secure resources required to achieve an agent's goals.

Interestingly, this result differs from the performance of these agent types in their equivalent complexity conditions in the results discussed in Sect. 6.1 (see Figs. 8, 9 and 10). In the original simulations, the performance of type 2 and 3 agents in the moderate complexity condition (value 2 on the x-axis of Fig. 9) is worse than their performance in the high complexity condition (value 3 on the x-axis of Fig. 10). Currently, this discrepancy is unexplained, and will be looked at in future work.

Importantly, smaller episodic memory sizes seem to produce better performance irrespective of environment complexity for type 2 agents (see Fig. 15). While this trend also appears to hold for type 3 agents in Fig. 16, its effect was not significant; type 3 agents use their pattern-recognition systems less frequently, on average, than type 2 agents (discussed in Sect. 6.1: *Decision-Making System Use* and reinforced by Fig. 13). By reducing episodic memory size from the median value of 10 (the value used for this variable in the first set of simulations), it is possible to tease out a performance increase of up to 3.61% (see Table 4).

Table 4. Average scores and percentage increases thereof as episodic memory size is decreased from original value specified in simulations run in Sect. 5.1 for type 2 agents across all sensitivity analysis complexity conditions.

Environment Complexity	**Episodic Memory Size**	**Avg. Score (% Inc.)**
Low	10	152.99
	6	158.52 (3.61)
Mod.	10	409.15
	6	414.30 (1.26)
High	10	267.70
	6	270.77 (1.15)

By retaining fewer episodes, actions that contributed little to the achievement of the agent's primary goal are ignored. This would result in agents creating and reinforcing more useful productions resulting in optimised goal achievement. For type 3 agents it may be the case that productions which prescribe use of the problem-solving system are created and subsequently reinforced more than productions prescribing explicit actions, frequently resulting in protracted periods of deliberation. This would cause either a decrease in performance or, at the least, performance to plateau; the latter was observed for the performance of these agents in low and moderate complexity conditions (see Fig. 16).
As noted in Sect. 6.2: *Performance*, an interaction between environment complexity and episodic memory size produced a negative effect on performance: as environment complexity increased, larger episodic memory sizes impinged performance. Since smaller episodic memory sizes reduce the chance of non-optimal productions being used (as argued in the previous paragraph) and more complex environments increase the likelihood of an agent reconsidering its intentions and remaining inanimate, it follows that being more discriminating with regard to the utility of one's productions will enhance performance and vice-versa.
Interestingly, this performance impingement was less significant for type 3 agents than type 2 agents, but was less likely to occur due to chance with type 3 agents. This result is most likely produced due to type 3 agents spending more time deliberating than acting, reducing the total number of productions generated. So, whilst type 2 and 3 agents may produce productions whose utility ratings are equivalent, agents of type 3 produce less of them in the same space of time and thus use fewer of them. This accounts for the plateau in performance observed in the low and moderate complexity conditions for type 3 agents too: agents spend less time reconsidering their intentions due to reduced environment dynamism but the amount of productions executed then attracts a premium. In such circumstances, type 2 agents profit since they can execute more (potentially non-optimal) productions than type 3 agents.

Conclusions. Around half of the null hypotheses stated in Sect. 5.2 are strongly refuted. Varying episodic memory size had a significant effect upon performance and average frequency of problem-solving and

pattern-recognition system use for agent types 2 and 3. Reducing episodic memory size improves performance for both types of agent, although not by any notable degree. In constrast, decision-making system use is notably affected by varying episodic memory size: larger memories promote pattern-recognition system use whereas smaller memories promote problem-solving system use. This is important since it was argued in Sect. 6.1 that increased use of the pattern-recognition system benefits performance more so than increasing use of the problem-solving system. So, whilst reducing episodic memory size appears to improve performance by creating more useful productions, it also reduces use of the pattern-recognition system in general; a balance must be struck.
Environment complexity also affected performance and decision-making system use significantly: for both agent types, moderate environment complexity produced the best performance whilst low complexity produced the worst performance. With regard to decision-making system use, the effect of environment complexity was significant on both performance and decision-making system use for type 2 agents whereas for type 3 agents, the effect was only consistently significant for performance. The effect of environment complexity on performance is different than that observed in the original simulations and it is our intention to investigate this further in future work.
Altering discount rates did not produce any significant effects upon performance or pattern-recognition system use apart from a marginally significant effect that is observed upon average frequency of problem-solving and pattern-recognition system use for type 3 agents. The reason for this is unclear but given that the effect is hardly significant, an explanation seems unwarranted.

7 Conclusions and Future Work

In this paper we have described and implemented a novel, modular dual-process [26] architecture for self-learning, computational agents. The architecture consists of a problem-solving and pattern-recognition decision-making system, created using a combination of the CHREST architecture, the PSDR algorithm and the Roulette selection mechanism. The system implemented different balances of problem-solving and pattern-recognition use: pure problem-solving, “pure” pattern-recognition and a mixture of both. These balances of the two systems were embodied as three types of agent situated in the Tileworld environment. We used these agents to ascertain how different balances of problem-solving and pattern-recognition system affected performance in this environment given different degrees of intrinsic and extrinsic environmental complexity. We also explored how environment complexity affects agent performance and decision-making before conducting a sensitivity analysis to ascertain if the dependent variables studied in the first set of simulations were significantly altered when salient variables governing the mechanism of the pattern-recognition system were varied.
Use of pattern-recognition was beneficial to agent performance, especially when intrinsic and extrinsic environment complexity was increased,

whereas use of problem-solving was less beneficial, due to the required time to solve problems. As overall environmental complexity increased, we found that agents using pure problem-solving (that is, the complete absence of pattern-recognition) are further disadvantaged whereas agents that were more likely to use pattern-recognition performed best. Our results therefore demonstrate that an agent which can use both problem-solving and pattern-recognition is at an advantage in the complex, dynamic environment modelled and even more so when pattern-recognition is favoured. Essentially, the results indicate that agent performance is maximised by generating (potentially sub-optimal) productions quickly and executing many of them, at least in the environment modelled here. This is an interesting finding given that the Einstellung effect [30] is likely to be manifest in the agents that perform best.
The sensitivity analysis performed corroborated these findings, but also revealed the variables which maximise the performance of agents capable of pattern-recognition given different complexities of the environment modelled. We discovered that, whilst episodic memory size affected agent performance significantly (albeit minimally), discount rate did not. Indeed, episodic memory size appears to exert an influence on the balance of agent performance and promotion of pattern-recognition use. Smaller episodic memory sizes enhanced agent performance but made it less likely for the agent to actually use pattern-recognition. This is because smaller episodic memory sizes provide the agent with fewer reinforced productions, resulting in problem-solving being used instead.
In future work, we would like to ascertain why the effect of environment complexity differs with respect to performance between the two sets of simulations run: is this due to chance or a more exact reason? We also intend to determine if these conclusions still hold when the same simulations are run for longer periods of time, and when heterogeneous agent types compete. Finally, we will consider if these conclusions generalise to other domains and when the amount of information capable of being reasoned with by agents is increased, by expanding the size of their observable environment.

References

1. Aarts, H., Dijksterhuis, A.: Habit as knowledge structures: Automaticity in goal-directed behavior. Journal of Personality and Social Psychology 78(1), 53–63 (2000)
2. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebière, C., Qin, Y.L.: An integrated theory of the mind. Psychological Review 111(4), 1036–1060 (2004)
3. Arai, S., Sycara, K.: Effective learning approach for planning and scheduling in multi-agent domain. In: Meyer, J.A., Berthoz, A., Floreano, D., Roitblat, H., Wilson, S.W. (eds.) From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior. pp. 507–516. MIT Press (2000)
4. Arai, S., Sycara, K.P., Payne, T.R.: Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. In:

Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence. pp. 125–135 (2000)
5. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application. L. Erlbaum Associates Inc. (1987)
6. Bilalić, M., McLeod, P., Gobet, F.: Inflexibility of experts - reality or myth? Quantifying the Einstellung effect in chess masters. Cognitive Psychology 56(2), 73–102 (2008)
7. Bossomaier, T., Traish, J., Gobet, F., Lane, P.C.R.: Neuro-cognitive model of move location in the game of Go. In: Proceedings of the 2012 International Joint Conference on Neural Networks (2012)
8. Chase, W.G., Simon, H.A.: Perception in chess. Cognitive Psychology 4, 55–81 (1973)
9. Dayan, P., Daw, N.D.: Decision theory, reinforcement learning, and the brain. Cognitive, Affective and Behavioral Neuroscience 8(4), 429–453 (2008)
10. de Groot, A.D.: Thought and Choice in Chess (First edition in 1946). Mouton, The Hague (1978)
11. de Groot, A.D., Gobet, F.: Perception and Memory in Chess: Heuristics of the Professional Eye. Van Gorcum, Assen (1996)
12. Erev, I., Roth, A.E.: Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. The American Economic Review 88(4), pp. 848–881 (1998)
13. Evans, J.S.B.T.: Dual-processing accounts of reasoning, judgment and social cognition. Annual Review of Psychology 59, 255–278 (2008)
14. Freudenthal, D., Pine, J.M., Gobet, F.: Simulating the referential properties of Dutch, German and English root infinitives in MOSAIC. Language Learning and Development 15, 1–29 (2009)
15. Gillan, C.M., Papmeyer, M., Morein-Zamir, S., Sahakian, B.J., Fineberg, N.A., Robbins, T.W., de Wit, S.: Disruption in the balance between goal-directed behavior and habit learning in obsessive-compulsive disorder. American Journal of Psychiatry (2011)
16. Gobet, F.: Les mémoires d'un joueur d'échecs. Editions Universitaires, Fribourg, Switzerland (1993)
17. Gobet, F.: A pattern-recognition theory of search in expert problem solving. Thinking and Reasoning 3, 291–313 (1997)
18. Gobet, F., Lane, P.C.R., Croker, S.J., Cheng, P.C.H., Jones, G., Oliver, I., Pine, J.M.: Chunking mechanisms in human learning. Trends in Cognitive Sciences 5, 236–243 (2001)
19. Grefenstette, J.J.: Credit assignment in rule discovery systems based on genetic algorithms. Machine Learning 3, 225–245 (1988)
20. Hesketh, B.: Dilemmas in training for transfer and retention. Applied Psychology 46(4), 317–339 (1997)
21. Holroyd, C.B., Coles, M.G.: The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity. Psychological Review 109(4), 679–709 (2002)

22. Jones, G.A., Gobet, F., Pine, J.M.: Linking working memory and long-term memory: A computational model of the learning of new words. Developmental Science 10, 853–873 (2007)
23. Jongman, R.W.: Het Oog Van De Meester. Assen: Van Gorcum (1968)
24. Kheirbek, M.A., Klemenhagen, K.C., Sahay, A., Hen, R.: Neurogenesis and generalization: a new approach to stratify and treat anxiety disorders. Nature Neuroscience 15(12) (2012)
25. Laird, J.E.: The Soar Cognitive Architecture. MIT Press (2012)
26. Lane, P.C.R., Gobet, F.: CHREST models of implicit learning and board game interpetation. In: Bach, J., Goertzel, B., Ikle, M. (eds.) Proceedings of the Fifth Conference on Artificial General Intelligence. vol. LNAI 7716, pp. 148–157. Springer-Verlag, Berlin, Heidelberg (2012)
27. Lloyd-Kelly, M., Gobet, F., Lane, P.C.R.: The art of balance: Problem-solving vs. pattern-recognition. In: Proceedings of the International Conference on Agents and Artificial Intelligence. pp. 131–142 (2015)
28. Lloyd-Kelly, M., Gobet, F., Lane, P.C.R.: Piece of mind: Long-term memory structure in ACT-R and CHREST. In: Noelle, D.C., Dale, R., Warlaumont, A.S., Yoshimi, J., Matlock, T., Jennings, C.D., Maglio, P.P. (eds.) Proceedings of the 37th Annual Meeting of the Cognitive Science Society. pp. 1422–1427. Cognitive Science Society (2015)
29. Lloyd-Kelly, M., Lane, P.C.R., Gobet, F.: The effects of bounding rationality on the performance and learning of CHREST agents in tileworld. In: Research and Development in Intelligent Systems XXXI, pp. 149–162. Springer International Publishing (2014)
30. Luchins, A.S.: Mechanization in problem solving: The effect of einstellung. Psychological Monographs 54(6), i–95 (1942)
31. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review 63, 81–97 (1956)
32. Miyazaki, K., Yamamura, M., Kobayashi, S.: On the rationality of profit sharing in reinforcement learning. In: 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing. pp. 285–288. Korean Institute of Intelligent Systems (1994)
33. Pollack, M., Ringuette, M.: Introducing the Tileworld: Experimentally evaluating agent architectures. In: Eighth National Conference on Artificial Intelligence. pp. 183–189. AAAI Press (1990)
34. Raza, M., Sastry, V.: Variability in behavior of command agents with human-like decision making strategies. In: Tenth International Conference on Computer Modelling and Simulation. pp. 562–567 (2008)
35. Saariluoma, P.: Error in chess: The apperception-restructuring view. Psychological Research 54, 17–26 (1992)
36. Samsonovich, A.: Toward a unified catalog of implemented cognitive architectures. In: Proceedings of the 2010 Conference on Biologically Inspired Cognitive Architectures. pp. 195–244. IOS Press, Amsterdam, The Netherlands (2010)

37. Simari, G.I., Parsons, S.D.: On approximating the best decision for an autonomous agent. In: Sixth Workshop on Game Theoretic and Decision Theoretic Agents. pp. 91–100. Third Conference on Autonomous Agents and Multi-agent Systems (2004)
38. Simon, H.A.: The sciences of the artificial. MIT Press, Cambridge, MA (1969)
39. Simonton, D.K.: Origins of genius: Darwinian perspectives on creativity. Oxford University Press, New York (1999)
40. Sloman, S.: The empirical case for two systems of reasoning. Psychological Bulletin 119, 3–22 (1996)
41. Sternberg, R.J.: The road to excellence: The acquisition of expert performance in the arts and sciences, sports, and games, chap. Costs of expertise, pp. 347–354. Hillsdale, NJ: Lawrence Erlbaum Associates (1996)
42. Sun, R., Merrill, E., Peterson, T.: From implicit skills to explicit knowledge: A bottom-up model of skill learning. Cognitive Science 25, 203–244 (2001)
43. Sun, R., Slusarz, P., Terry, C.: The interaction of the explicit and the implicit in skill learning: A dual-process approach. Psychological Review 112(1), 159–192 (2005)
44. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
45. Watkins, C.J.C.H., Dayan, P.: Technical note: Q-learning. Machine Learning 8, 279–292 (1992)
46. de Wit, S., Dickinson, A.: Associative theories of goal-directed behaviour: a case for animalhuman translational models. Psychological Research 73(4), 463–476 (2009)
47. Zeitz, C.M.: Expertise in context: Human and machine, chap. Some concrete advantages of abstraction: How experts' representations facilitate reasoning, pp. 43–65. Cambridge, MA: The MIT Press (1997)