Dynamics Based Control with PSRs

Ariel Adam
School of Engineering and
Computer Science
The Hebrew University of
Jerusalem
adama@cs.huji.ac.il

Zinovi Rabinovich
School of Electronics and
Computer Science
Southampton University
zr@ecs.soton.ac.uk

Jeffrey S. Rosenschein
School of Engineering and
Computer Science
The Hebrew University of
Jerusalem
jeff@cs.huji.ac.il

ABSTRACT

We present an extension of the Dynamics Based Control (DBC) paradigm to environment models based on Predictive State Representations (PSRs). We show an approximate greedy version of the DBC for PSR model, *EMT-PSR*, and demonstrate how this algorithm can be applied to solve several control problems. We then provide some classifications and requirements of PSR environment models that are necessary for the EMT-PSR algorithm to operate.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Experimentation

Keywords

Dynamics Based Control, Predictive State Representation

1. INTRODUCTION

Autonomous agents, and the entities they represent, regularly encounter systems that develop over time. Modifying network transmission rates, controlling the motion of remote actuators, or positioning a target within a field of vision (by moving the target or the vision system), can all be described as a control task of a dynamic stochastic system. One common model for formalising such tasks has been Partially Observable Markov Decision Problems (POMDPs) [11, 3]. POMDPs assume that the true state of the system is obscured, and can be only indirectly observed from the available sensory input. The control task is then translated into a utility function that uniformly describes the benefit of different steps in the system's development, and the action policy is optimised to produce the highest expected utility [9]. While theoretically attractive, such an approach in practise has high computational complexity [7, 5], which has encouraged the development of additional system modelling and control frameworks.

Predictive State Representations (PSRs) [6, 16] have been proposed as a new way of creating models of controlled dynamic systems. PSRs describe a system's dynamic development based exclusively on observed properties: sensory input, and the available actions. Based on the predicted success levels of some future action-observation sequences, termed *tests*, PSRs allow the recovery of the probability of success of any given sequence of actions that can be applied, and the observations that would follow them, resulting in a modelling scheme that is at least as powerful and compact as hidden state models [6].

Dynamics Based Control (DBC) [15] is a new approach to controlling agents in stochastic environments. Unlike previous techniques, which seek to optimise expected rewards (e.g., in POMDPs), DBC optimises system behaviour *towards specified system dynamics*. One specific instance of the overall DBC framework is the controller based on Extended Markov Tracking (EMT) [12, 13, 14]; EMT employs greedy action selection to provide an efficient control algorithm in Markovian environments. The EMT-based controller assumes that the system can be forced to behave as a specified Markov chain—an ideal dynamics—and uses the underlying EMT algorithm to estimate the effects of different actions. The action with the effect most aligned with the ideal dynamics is then applied.

In this paper, we combine the PSR representation with the DBC approach. We propose a new system-tracking algorithm under the PSR model, the EMT-PSR algorithm, which is designed to recover the transition rule between two consecutive actions and observations in the immediate future—that is, the transition between *tomorrow* and *the day after tomorrow*, under the assumption that such a transition is Markovian. The resulting polynomial-time control algorithm directly operates over observable quantities, the sensory inputs, and the available actions, allowing for a more intuitive and dynamics-directed specification of the control task. We experimentally show that behavioural heuristics can be directly incorporated into the algorithm's task specification, resulting in improved control performance.

The remainder of the paper is structured as follows. In Section 2 we review the PSR environment, review the DBC framework, and present the EMT-PSR algorithm in detail. In Section 3 we describe the domains used for the algorithm's performance evaluation, including domain-specific performance measures and the empirical results obtained for each of the domains. Sections 4 and 5 describe the necessary modelling characteristics and the control task specification guidelines for the EMT-PSR algorithm application. We conclude and discuss future work in Section 6.

2. THE EMT-PSR ALGORITHM

The recently introduced Dynamics Based Control (DBC) [15] framework, which we review below in Section 2.2, views the control task as follows: force a given algorithm for system identification to produce a result as close as possible to a specified (ideal)

one. This point of view requires an environment model to predict the response of the identification algorithm, and to operate in terms of connections between the actions that will be applied and the observations that will then be made. Although all previous implementations of the DBC framework [12] used a hidden state model, in this paper DBC is implemented using a new approach. We construct a DBC algorithm using an environment model that operates only in terms of direct observables: sensory input and actions selected, the elements of Predictive State Representations (PSRs).

PSRs [6, 16] are based on the fact that in some systems, given the history of their development, it is enough to know only a few predictions of their future development in order to predict their *entire* future. Among the benefits of PSRs (discussed below in Section 2.1) is their representation of a system directly in terms of observable and deliverable aspects of the environment, i.e., their representation directly captures the relationship between future observations an agent may have, and the future actions it may apply.

2.1 Predictive Representation of State

Predictive State Representations (PSRs) were introduced in [6]; the approach views a system as a set of dynamically developing predictions of future observations.

Given a (discrete) set A of actions available to an agent, and a (discrete) set O of possible observations, PSRs focus on sequences of future actions and observations or *tests* $(A \times O)^+$. A test $t = a_1 o_1 ... a_r o_r \in (A \times O)^+$ is said to succeed if the sequence of observations $o_1, ..., o_r$ is obtained as the sequence of actions $a_1, ..., a_r$ is executed. As actions are executed and observations are made, an interaction history $h \in (A \times O)^*$ is obtained, and the probability of any given test $p(t|h)$ varies.

The set of success probabilities of all tests is infinite. However, in some systems it is possible to identify a subset Q of tests, termed *core tests*, such that the success probability $p(t|h)$ of any test can be computed from the success probabilities of the core tests $p(Q|h) = \{p(q|h)\}_{q \in Q}$. The dependency between $p(Q|h)$ and $p(t|h)$ is independent of the interaction history h, though it may be non-linear.

In this paper we will concentrate on Linear Discrete PSRs [6, 16], which can be defined by the tuple $< A, O, Q, p(Q|\emptyset), \mathbf{M}, \mathbf{m} >$, where:

- A is the set of actions available to an agent under the model;

- O is the set of observations an agent can encounter under the model;

- Q is the set of key future points, *core tests*, $Q \subset (A \times O)^+$. In a linear PSR this set is finite, and its size dictates the *dimension* of the model: $Q = \{q_1, ..., q_k\}$;

- $p(Q|\emptyset) = [p(q_1|\emptyset), ..., p(q_k|\emptyset)]$ is the initial vector of predictions regarding the core tests. This is the probability that the test will succeed, given that the interaction history is empty. In general $p(Q|h)$, where $h \in (A \times O)^*$ is the history of interaction, is called the *PSR state*;

- $\mathbf{M} = \{M_{ao}^T = [m_{aoq_i}] | q_i \in Q\, a \in A, o \in O\}$ is the set of one-step extension weight matrices. Each row in M_{ao} is a weight vector $m_{aoq_i}^T$, so that $p(aoq_i|h) = m_{aoq_i}^T p(Q|h)$;

- $\mathbf{m} = \{m_{ao} | a \in A, o \in O\}$ is the set of weight vectors for the single-step tests ao, that is, $p(ao|h) = m_{ao}^T p(Q|h)$.

It is important to note some notation abuse: $P(a_1 o_1 ... a_r o_r | h) = P(o_1, ..., o_r | h, a_1, ..., a_r)$. The left term is the probability of the test $a_1 o_1 ... a_r o_r$, while the right term is its correct interpretation

as a conditional probability: the probability that the sequence of observations $o_1, ..., o_r$ will take place if the sequence of actions $a_1, ..., a_r$ will take place after history $h \in (A \times O)^*$.

The linearity assumption states that for any test $t = a_1 o_1 ... a_r o_r \in (A \times O)^+$ there exists a constant vector of weights m_t so that $p(t|h) = m_t^T p(Q|h)$, where $m_t = m_{a_r o_r}^T M_{a_{r-1} o_{r-1}} ... M_{a_1 o_1} p(Q|h)$. The PSR state update after applying action $a \in A$ and receiving observation $o \in O$ is given by:

$$P(q_i|hao) = \frac{m_{aoq_i} P(Q|h)}{m_{ao} P(Q|h)}.$$

2.2 Dynamics Based Control

The Dynamics Based Control (DBC) framework introduced in [15] is based on two key features: the perceptual control principle, and the focus on system dynamics.

The DBC framework can be decomposed into three major levels:

- The **Environment Design** level describes the system model. In this paper, we will concentrate on PSR models of the environment.

- The **User** level defines a system identification or a system tracking algorithm, and a measure of similarity is established between feasible outcomes of the tracking algorithm. An ideal dynamics (also termed a *tactical target*), towards which the outcome of the tracking algorithm has to be forced, is also defined at this level. The ideal dynamics describes one's preferences regarding, and requirements for, the system's development and performance.

- The **Agent** level defines an on-line control procedure which sets actions that force the tracking algorithm towards the ideal dynamics.

The data flow for the framework is depicted in Figure 1.

Despite the generality of the DBC framework, we are thus far aware of only one system-tracking algorithm capable of operating within this framework, which has been publicised by the DBC authors [12, 13, 14, 15]—Extended Markov Tracking (EMT).

The EMT algorithm assumes that the system is an autonomous single Markov chain, and it thus views the system dynamics as a single stochastic matrix. EMT performs a sequence of conservative updates of the system dynamics matrix, minimising the Kullback-Leibler divergence between the new and old estimates, with the limitation that the new estimate has to match the system transition sample that triggered the update.

Assume that two probability distributions p_{t-1}, p_t are given that describe two consecutive states of knowledge about the system, and τ_{EMT}^{t-1} is the old estimate of the system dynamics. Then the EMT update τ_{EMT}^t is the solution of the following optimisation problem, where D_{KL} is the Kullback-Leibler divergence:

$$
\begin{aligned}
\tau_{EMT}^t &= H[p_{t-1}, p_t, \tau_{EMT}^{t-1}] \\
&= \arg\min_\tau D_{KL}(\tau \times p_{t-1} \| \tau_{EMT}^{t-1} \times p_{t-1}) \\
s.t.& \\
& p_t(x') = \sum_x (\tau \times p_{t-1})(x', x) \\
& p_{t-1}(x) = \sum_{x'} (\tau \times p_{t-1})(x', x)
\end{aligned}
$$

Note the following abbreviation of the update:
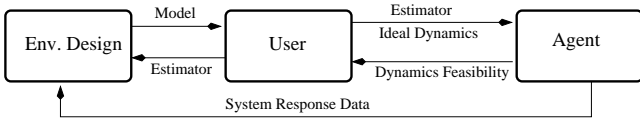$\tau_{EMT}^t = H[p_{t-1}, p_t, \tau_{EMT}^{t-1}]$.

Figure 1: Data flow of the DBC framework

2.3 EMT-PSR Implementation of DBC

In this section, we present an implementation of the DBC framework with the following assumptions imposed on its constituent levels:

- The Environment level produces a PSR environment model;

- The User level treats a distribution over future action-observation pairs at time t as the state of knowledge about the system at time t, and assumes consecutive time steps to be linked by a Markovian process. The User level uses EMT as the system tracking algorithm.

The core idea of the algorithm at the Agent level is to make EMT track the difference between the observation distribution of the next time step (termed *tomorrow*) and the time step after the next one (termed *the day after tomorrow*).

The Agent level control algorithm is a non-committing two-step lookahead. It selects a pair of actions based on the prediction of EMT as to how similar the difference between *tomorrow* and *the day after tomorrow* would be, relative to the tactical target. The algorithm then applies the first part of the pair, thus moving the system one step ahead. The old *day after tomorrow* becomes the new *tomorrow*, and the algorithm uses the second action of the pair it selected to recompute *the day after tomorrow* distribution of observations. It then calls EMT to update its estimate, based on the difference between the old *tomorrow* and the new *tomorrow*.

Given the PSR tuple $< A, O, Q, p(Q|\emptyset), \mathbf{M}, \mathbf{m} >$, denote as usual τ^* as the tactical target, and $H[p_t, p_{t+1}, \tau]$ as the EMT optimisation procedure computation.

0. **Initialise:**
   - Set the time to $t = 0$.
   - Set the EMT estimate τ_t to be a conditional uniform distribution.
   - Set the PSR model to its initial PSR state.
   - Denote by $p_t^{+a} \in \Pi(O)$ the distribution over the observations at time $t + 1$ (*tomorrow*), after taking action $a \in A$ as is estimated at time t; denote by $p_t^{+a_1+a_2} \in \Pi(O)$ the distribution over the observations at time $t+2$ (*the day after tomorrow*) after taking actions a_1 and $a_2 \in A$, as is estimated at time t.

1. **Prediction stage:** for each pair of actions $(a_1, a_2) \in A^2$ compute:
   - The distribution $p_t^{+a_1} = P(a_1 o_1 | h) \in \Pi(O)$ by applying the PSR model and its current state:
     $$\forall o_1 \in O, P(a_1 o_1 | h) = m_{a_1 o_1}^T P(Q|h).$$
   - The distribution $P(a_1 o_1 a_2 o_2 | h_t) \in \Pi(O^2)$ by applying the PSR model and its current state:
     $$\forall o_1, o_2 \in O, P(a_1 o_1 a_2 o_2 | h) = m_{a_2 o_2}^T M_{a_1 o_1} P(Q|h).$$
   - Compute $p_t^{+a_1+a_2} = \sum_{o_1 \in O} P(a_1 o_1 a_2 o_2 | h)$.

   - Compute $D_{a_1 a_2} = H[p_t^{+a_1}, p_t^{+a_1+a_2}, \tau_t]$.

2. **Action selection:**
   $$(a_1^*, a_2^*) = \arg \min_{(a_1, a_2) \in A^2} D_{KL}(D_{a_1 a_2} \times p_t^{+a_1} \| \tau^* \times p_t^{+a_1}).$$

3. **Application and Update:**
   - Apply a_1^*, receive observation $o_1^* \in O$.
   - Compute $p_t^{+a_2^*} = P(a_2^* o_2 | h a_1^* o_1^*) \in \Pi(O)$ as before:
     $$\forall o_2 \in O, P(a_2^* o_2 | h) = m_{a_2^* o_2}^T P(Q|h a_1^* o_1^*).$$
   - Update $\tau_{t+1} = H[p_t^{+a_1^*}, p_t^{+a_2^*}, \tau_t]$.
   - Update the PSR history $h = h a_1^* o_1^*$.
   - Update the PSR state as prescribed.
   - t=t+1.

3. TESTING THE EMT-PSR ALGORITHM

When testing and evaluating the performance of the EMT-PSR algorithm, one faces two challenges.

First, to the best of our knowledge, there is no optimal solution algorithm available within the DBC framework, nor a benchmark problem with a clear performance metric. Thus, instead of a comparative study, we have resorted to the use of a range of problems, each with its own intrinsic domain-specific performance metric. Most of these domains are more commonly modelled and solved using the POMDP approach [1, 8, 10, 4, 2]. However, in this paper we formalised them as PSRs, either directly or using the conversion algorithm provided in Littman et al. [6].

The second challenge comes from the need to provide DBC-type solutions with an *ideal dynamics matrix*, or *tactical target*. DBC assumes that the tactical target is completely evident from the definition of the domain or from a specific instance at hand. We find, however, that in many cases the ideal dynamics matrix is generally formed from a set of *heuristics* for a domain.

3.1 Domain Dependent Tactical Targets

From the EMT point of view (and it is that to which the tactical target has to conform), the system is guided by a single Markovian conditional distribution, a matrix that represents the ideal rules of system development. In general such rules would have the form of a conditional implication between two consecutive expressions of system knowledge. Under our system modelling assumptions, the most simple rule would have a form of : **if** *tomorrow's observation* **then** *the day after tomorrow's observation* has ζ degree of preference.

This allows for a rather flexible specification of the desired system behaviour, which may include dependencies between actions, sensory observations, and an external feedback of performance evaluation.[1] Furthermore, if the exact rule set is unknown, one may formulate the tactical target via a set of heuristic rules. In this case, the ideal dynamics matrix can be viewed as behavioural guidance to the Agent Level algorithm, rather than an ideal system development to which an agent has to adhere.

In the following subsections we apply the EMT-PSR algorithm towards solving a range of control and continual planning problems in stochastic domains. For each domain, we provide a specific set of behavioural rules that were used to create the ideal dynamics

---

[1] Such an evaluation is commonly termed *reward* in the Reinforcement Learning literature [17].

matrix. These rules, although domain specific, stem from general heuristics for solving the class of problems to which each domain belongs. For instance, the Maze Domains (Sections 3.3, 3.4) and the Shuttle Domain (Section 3.2) can be characterised as support of motion through a graph. In this type of domain, the ideal system development is simply a path through the graph which has optimal properties with respect to the specifications of the domain. As a result, the set of rules that form the ideal dynamics matrix simply describes the path, or a heuristic for its reconstruction.

The domains of Sections 3.2–3.6 adhere to the development of the EMT-PSR evaluation, and form an ascending order of complexity and challenge to the control algorithm. From the simplest to the most difficult, all domain descriptions follow the same presentation pattern. Every section begins with a description of the domain, and introduces the rules used to form the ideal dynamics matrix. We then discuss the performance measures relevant to the domain, and provide experimental results of the EMT-PSR algorithm's execution with respect to that measure.

3.2 The Shuttle Domain

In the shuttle domain [1] there are two identical space stations each containing a loading dock; the stations are separated from one another. The goal of the problem is to provide continual transport of supplies between the two stations. The agent's aim is thus to continually move from the most-recently visited station (MRV) to the least-recently visited (LRV) station.[2] For docking, the agent should reach the station, turn around so that he is positioned with his back to the dock, and back up into the station. If the agent attempts to move into the station's dock while facing it (instead of backing in), a collision occurs.
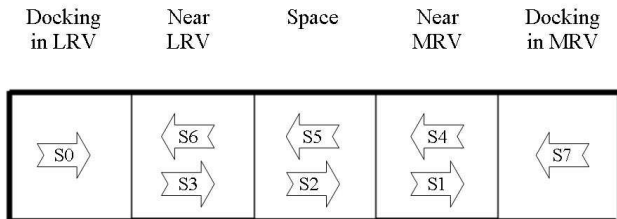

Figure 2: Shuttle domain environment description. The agent has 8 potential directed positions.

The domain's *Actions* space is {GoForward, Backup, TurnAround}. Backup is the only noisy action, with a 70% success rate. The *Observations* space, {See LRV, See MRV, See docked in MRV, See docked in LRV, See nothing}, provides the agent with the orientation information relative to the two stations. Note the ambiguity of the *See nothing* observation obtained in the open space between the stations—it provides positioning, but not orientation information, and can be received both when the Shuttle is directed *towards* and *away from* the LRV.

As can be seen from Figure 2, the shortest route from MRV to LRV, assuming that all actions succeed, consists of five steps: three *Forward* actions, followed by the *TurnAround* and *Backup* actions.

Intuitively, these five steps are characterised by the following rules, which were used to create the *ideal dynamics matrix* for this domain:

- If tomorrow you go forward and see the LRV station ahead,

---

[2]Of cause, once the LRV is reached it becomes the MRV, which naturally resets the problem.

then on the day after tomorrow you should TurnAround and hope to see nothing or the MRV ahead.

- If tomorrow you performed a TurnAround action and see nothing or the MRV station ahead, then on the day after tomorrow Backup and hope to see yourself docked in the LRV.

To verify the quality of the action sequence produced by the EMT-PSR control solution, we have chosen two domain characteristic measurements. The first is the number of steps it takes the agent to reach the LRV. This enables us to see how close the route chosen by the agent is to the shortest, five step, route. The second measure, which appears to be natural for control performance evaluation in this domain, is the number of times the agent crashed into a station compared to the number of times the agent has attempted to reach the LRV station.

In our experiment set, the agent attempted the MRV-LRV route 100,000 times using the EMT-PSR algorithm, and avoided ever crashing into either of the stations. The empirical distribution of the number of actions required to reach LRV from MRV is shown in Figure 3. In 71% of the attempts the agent reached its target in five steps, in 20% it took six steps, and so on with exponential decrease. Analysis of the action sequences showed that the agent took additional actions, beyond the necessary five, only if the concluding *Backup* action failed, and attempted to repeat it. This naturally resulted in the exponential decay of the number of longer docking sequences.

The experimental results show that the Docking Domain, although it contains both uncertain actions and sensory aliasing, presents little challenge to the EMT-PSR control algorithm. We thus proceeded to domains that supersede the Docking Domain with regard to the amount of sensory and system noise (uncertain actions) that they present, seeking the limit of the control algorithm's resilience.
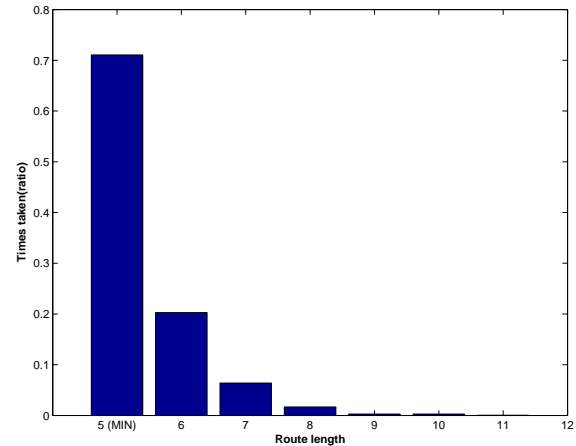

Figure 3: Shuttle Domain: Number of steps from MRV to LRV

3.3 The Cheese Maze Domain

In the cheese maze domain [8], the agent is placed on a non-regular grid (a maze), with one of the cells of the grid marked (i.e., containing a chunk of cheese). The agent is allowed to attempt to move in four directions on the grid, and is tasked to reach the cheese. Each time the agent reaches its target, it is randomly relocated to another cell of the grid.

Unlike in the Shuttle domain, the *Actions* space, {North, South, East, West}, consists of completely deterministic actions.[3] The

---

[3]Actions can (deterministically) fail if the agent attempts to move
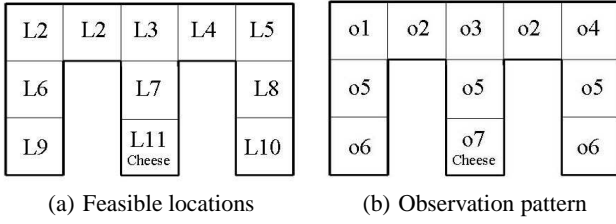
Figure 4: Cheese maze grid environment

only uncertainty comes from the random relocation and sensory aliasing. The agent's sensors provide information about the surrounding walls and the presence of the cheese chunk; the resulting space of seven possible *Observations* is depicted in Figure 4(b).

These rules were used for defining the *ideal dynamics matrix*:

- If tomorrow you go north and reach a corridor going east (west), then the day after tomorrow you should go east (west) and hope to see in front of you a corridor going east (west).

- If tomorrow you go east (west) and see that you are in a corridor going east (west), then on the day after tomorrow you should go east (west) and hope to see an intersection of east-west-south directions.

- If tomorrow you go east or west and see an intersection of east-west-south directions, then on the day after tomorrow go south and hope to see a corridor leading south.

- If tomorrow you go south and see a corridor going south, then on the day after tomorrow you should go south and hope to get the cheese.

To check the quality of the solution, we measured the number of steps needed for reaching the cheese for each of the possible starting positions. After 500,000 iterations[4] the number of extra steps was averaged over all possible starting locations, and the empirical distribution of extra steps depicted in Figure 5 was obtained. Once again, the number of extra steps decreases rapidly (at what appears to be an exponential rate), which seems to imply that the controller quickly recovers from the location ambiguity induced by the random relocation and the sensory aliasing.

3.4   The 4x3 Maze Domain

The 4x3 maze domain [10] is also a grid world domain where the agent can move in four directions, with the *Actions* space of {North, South, East, West}. However, in this domain even a feasible action can fail with some probability. The 4x3 Maze also has two marked positions: "+" denoting the desired goal, and "-" denoting a place to avoid. Every time the agent reaches any marked position, it is relocated to a random cell of the grid.

While in motion, the agent can perceive the properties of its location, i.e., the walls surrounding it and the markers, with the *Observations* set being {Wall to your left, Wall to your right, Walls on both sides, No walls, "+", "-"}.

These rules were used for defining the *ideal dynamics matrix*:

- If tomorrow you move in any direction and see walls to your left and right, then on the day after tomorrow move north and hope to see a wall to your left.

off the grid, in which case it remains at its old location.

[4]Similarly to the Shuttle Domain, reaching the target naturally resets the system.
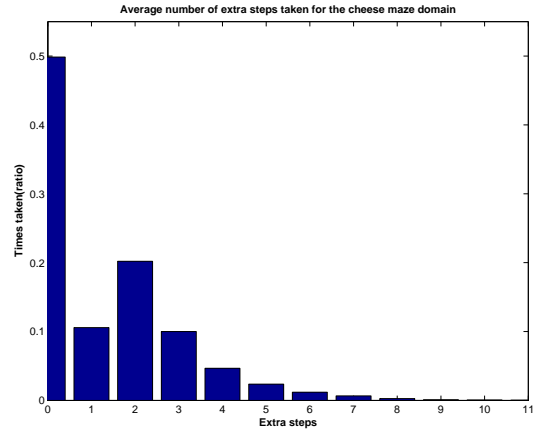

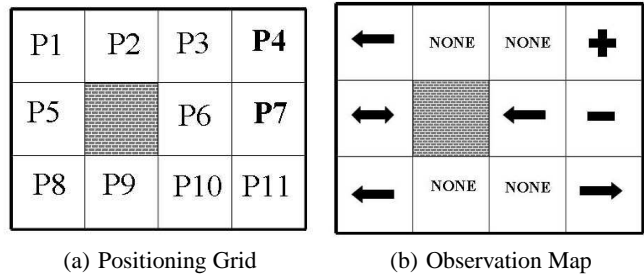**Figure 5: Cheese Maze: extra steps taken (0 denotes taking the optimal route)**

(a) Positioning Grid   (b) Observation Map

Figure 6: 4x3 maze domain environment description. Goals are reaching the "+" sign and avoiding the "-" sign.

- If tomorrow you move east or north and see no walls, then on the day after tomorrow go north and hope to see a wall to your left or go east and hope to see no walls or the "+" sign.

- If tomorrow you move north and see no walls, then on the day after tomorrow move east and hope to see a "+" sign.

- If tomorrow you perform any action and get observation "-", then on the day after tomorrow you should move west and hope to see a wall to your left.

Notice that the two tasks, reaching the "+" marks, and avoiding the "-" mark, are not strictly unifiable, and can be in conflict. Merging them into one single *tactical target* means that the behavioural trends have to be balanced at a low level, and their relative weights depend on the relative numerical values each one of the above rules is given in the tactical target.

For this domain we measured two parameters: the number of steps required beyond traversing the optimal route to the "+" goal position, and the ratio between the number of times the agent reached the "+" goal position and the number of times it reached the "-" avoidance position.

After 500,000 iterations of the problem, with a 90.1% success rate of reaching the "+" goal position, the empirical frequency of the number of extra steps taken beyond the optimal route was computed, averaging over all possible starting positions (see Figure 7). As with the previous domains involving route traversal, the delay en route from starting position to goal position decays exponentially.

In our tests, we used an ideal dynamics matrix that resulted (as mentioned above) in a 90.1% success rate of reaching the "+" mark.
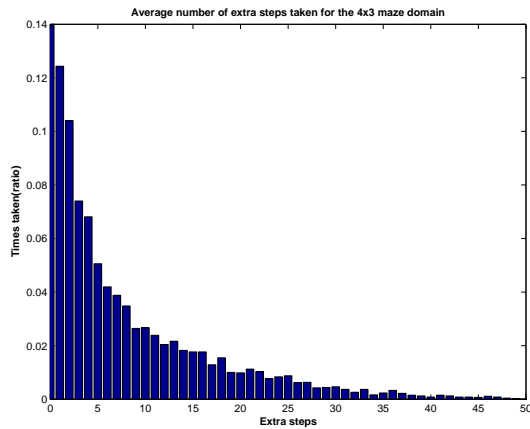
Figure 7: 4x3 Maze: Extra steps taken (0 denotes taking the optimal route)

This was accompanied by a larger divergence from the optimal route length, compared to previously described domains—41.6% of the time, the route taken exceeded the optimal route by more than 5 steps. This trade-off, however, was expected. Since reaching the "+" mark and avoiding the "-" mark comprise two conflicting behaviours, it results in a trade-off between the length of the route to the "+" mark, and the failure rate of reaching the "-" mark. We conjecture that, since both conflicting behaviours were explicitly weighed and merged into a single ideal dynamics matrix, the EMT-PSR algorithm reproduced that trade-off in practise. Additional experiments with a modified tactical target support this conjecture; these results are excluded from the paper due to space limitations.

3.5 The Network Domain

In [4], Littman describes a network domain with a controlled transmission rate that influences the general network utilisation level, with the seventh and last of the levels being an overload, at which the network "crashes". By applying one of the four available *Actions*, {unrestricted, steady, restricted, reboot}, the agent modulates the transmission rate (or the flow) of data through the network and thus its utilisation level. The *Unrestricted* flow increases the network load, the *Steady* flow may both increase or decrease the network load, and the *Restriction* of the flow reduces the network load. If the network crashes, then only the *Reboot* action can recover it and reset the system to the first and lowest utilisation level. Since both low utilisation level and an overload are disadvantageous, the task of this domain is to keep the network away from both extremes of the utilisation scale.

The observations available to an agent in this domain are multi-parametric, or multidimensional. First, the agent may assess if the network has actually crashed or not, is it "up" or "down", but the assessment is noisy and both values can appear even for medium network loads. Second, the agent can also assess the satisfaction of network users: if the network flow is too restricted, or the network is close to overload, the users are "unhappy", otherwise they are "content". As a result four possible *Observations* are formulated that describe the utilisation of the system: {underused (the network is "up", but users are "unhappy"), low-load ("up", and "happy"), high-load ("down", and "happy"), overload ("down" and "unhappy", the network has crashed)}.

These rules were used for defining the *ideal dynamics matrix*:

- If tomorrow you perform any action but unrestricted and observe low-load, then on the day after tomorrow you should

perform steady and hope to observe low-load.

- If tomorrow you perform any action and observe underused, then on the day after tomorrow you should perform unrestricted and hope to see low-load.

- If tomorrow you perform any action and observe high-load, then on the day after tomorrow you should perform restricted and hope to see low-load or high-load.

- If tomorrow you perform any action and observe overload, then on the day after tomorrow you should perform a reboot and hope to see any other observation.

To test the solution quality for the Network Domain, we measured the number of times each of the load levels was reached, and the number of times each one of the actions was taken. The empirical frequencies after 500,000 runs are shown in Figure 8.
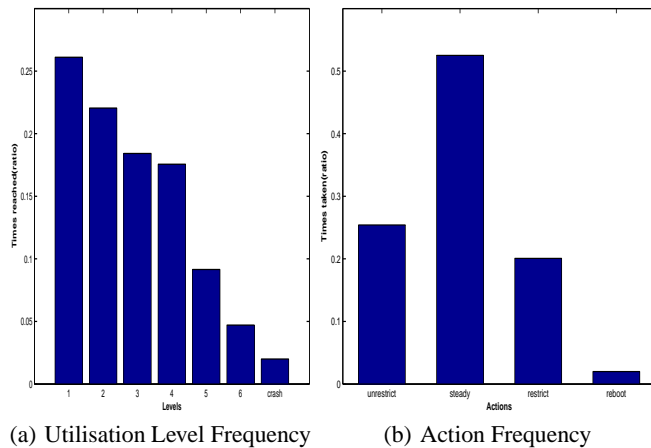

(a) Utilisation Level Frequency (b) Action Frequency

Figure 8: Network Domain: Empirical frequencies

As can be seen in Figure 8(a), the crash level was reached 2% of the time, and in complete accordance with that the "harsh" reboot operation was used 2% of the time (as can be seen in Figure 8(b)). This means that the control algorithm correctly interpreted the system model—it identified and consistently utilised the "reboot" action as the only action capable of restoring a crashed network.

Though the behaviour rules that define the ideal dynamics target explicitly require the agent to avoid utilisation level 1 (the lowest), the empirical results show that level was most frequently visited. Problem analysis showed that the mathematical model of the utilisation level transitions had a strong preference for naturally maintaining the low utilisation level, once it had been reached. Given that the utilisation level transitions also have a stochastic nature, this artificially kept the network at the low utilisation level, despite the controlling agent's efforts. The Network Domain is an example of a domain possessing strong tendencies towards certain goals that are derived from the environment's representation, and not only from the ideal dynamics matrix created for governing movement.

3.6 The Tiger Domain

In the Tiger Domain, an agent is facing the choice of opening one of two available doors. Behind one of the doors lurks a hungry tiger, and the agent would like to avoid such an encounter. The tiger growls slightly, and hearing the growl may assist the agent in determining which one of the doors not to open. Unfortunately, the growl echoes, and the agent may sometimes be misled.
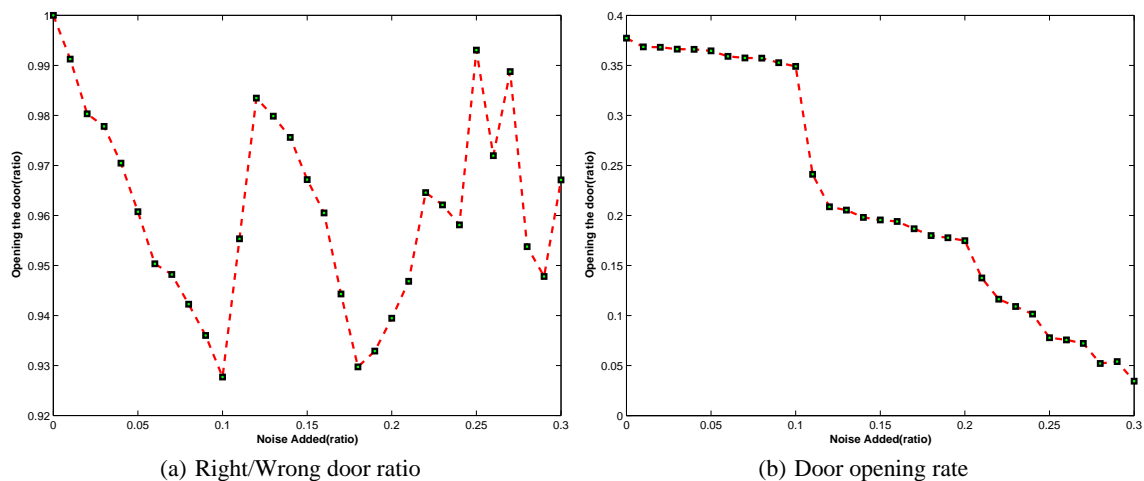
(a) Right/Wrong door ratio   (b) Door opening rate

Figure 9: Tiger Domain: Performance under noise 0-30%.

Unlike the classical modelling of the problem [2], we avoid explicit labelling of the doors as left and right, and the association of each door with an "open" action. Instead, we place the agent closer to one of the doors, and implicitly label them as the door closer to the agent, and the door further away. The agent can compare the sound that issues from behind each door by moving from one to the other, or open the door which at the moment is closest, thus forming the set of available *Actions:* {Open, Compare}. While comparing the sound coming from the doors, the agent may receive three possible *Observations:* {Louder, Same, Quieter}. *Louder* indicates that the sound of the tiger has become stronger when moving closer to a different door, *Same* indicates that the sound remains the same as before, and *Quieter* indicates that the sound appears to weaken.

The agent's observations depend on the system transitions: the *Open* action produces a *Louder* observation if the tiger is behind the opened door, and a *Quieter* observation if the tiger remains behind the closed door. The outcomes of the *Compare* action simulate the misleading echo of a growl that confuses the agent. The action produces the correct observation of the tiger's location only 85% of the time, the rest of the time giving the agent the *Same* observation.

In this domain, the ideal dynamics matrix is determined by a single rule: if tomorrow you perform an open operation and observe a louder sound, then on the day after tomorrow you should perform a compare action and hope to hear a quieter sound.

The algorithm was first run for 100,000 iterations on the domain. An interesting result was obtained—the agent always succeeded in opening the correct door without encountering the tiger. The agent always performed the sequence of operations where he first compared doors, until getting the *Quieter* observation, then opened the door. Taking a closer look at the observability of the domain, it became obvious that once an informative observation (*Louder* or *Quieter*) was obtained, the domain's uncertainty was completely resolved, which means the sequence created by the EMT-PSR algorithm will be optimal. Although the solution to the domain under the initial observability assumptions ended up being trivial, it was still interesting to note that this optimal solution was obtained automatically, through the use of a general purpose control algorithm.

To complicate the domain, additional forms of noise were added to the observations, with values from 1% to 30%. The meaning of adding 5% noise, for example, was that now instead of seeing the proper door 85% of the time, the agent sees the proper door 80% of the time. Then, the algorithm was run 50 times for each noise level with 1000 problem iterations in each run. The empirical data of the success level as a function of the noise level is shown in Figure 9(a). One would expect that the agent's performance would gradually decay with the increase in noise level, and it would open the wrong door more often. Surprisingly, this is not the case. The ratio between the correct door openings and wrong door openings spikes and fluctuates (Figure 9(a)), but never decays below 92%. Figure 9(b) suggests the source of the noise resistance: the number of times the agent actually opened a door, out of the 1000 steps of each system run, decays—the agent appears to become more careful, and tends to repeat the comparison of doors, rather than opening one. This link is further supported by the apparent phase transitions in both ratios between 0.10 and 0.12 noise values: as the rate of door openings plummets, the ratio of the correct door being open recovers to above 0.95.

4. REMARK ON DOMAIN MODELLING

When attempting to solve different domains using the EMT-PSR algorithm, we can identify two classes of domains that differ in the quality and feasibility of their solution, compared to other domains we have encountered.

The first group of domains are the *multi-dimensional observation domains*, such as the Network Domain [4]. In some control methods these domains would be divided, and only a subset of them would be used directly as a source of on-line information for the controller, while others would be used to drive the solver of the method. It occurs, for instance, in POMDPs where performance feedback (reward) and environment observations are treated as two separate classes of observable quantities, even in domains where the reward is not arbitrary delayed. We find such separation to be, in a sense, "improper", and treat all available observable quantities as a source of information. Thus, the observation space of the PSR models we have used explicitly includes actions, sensory observations, and the external performance signal (if one is available).

The second group is *domains containing identical "reset" actions*. When modelling some problems, one may be tempted to use several "reset" actions, which return the system to the initial knowledge and situation conditions. This can be allowed only if the actions are provided with some differentiating property with respect to obtained observations. Otherwise, we again consider the model to be in a sense "improper"; there are several actions that cannot be distinguished, and which require remodelling of the domain.

5. GUIDELINES FOR CREATING AN IDEAL DYNAMICS MATRIX

Ideally, we would like to state a set of behavioural rules, encode them into the ideal dynamics matrix, and run EMT-PSR to obtain the necessary behaviour. However, there are two issues one has to consider during the process of behaviour rule encoding:

1. The relative numerical expression each of the behavioural rules has within the ideal dynamics matrix is important. The ratio of the numerical expression translates through the EMT-PSR algorithm into expressiveness of each of the rules in the overall controlled system behaviour. Since the EMT-PSR uses a logarithmic, rather than a linear, comparison procedure, in many cases the behaviour rules' numerical encodings need to differ by an order of magnitude in order to express the proper overall mixture of expressiveness.

2. From our series of tests, it appeared that most systems have a strong tendency towards certain goals. This behaviour was clearly seen when the ideal dynamics matrix was replaced by a uniform dynamics matrix. Despite the uniform tactical target, the system did not perform in a random fashion, but rather moved towards clear goals that were not present in the dynamics matrix. We refer to this behaviour as *natural system tendencies*. This behaviour results from EMT-PSR choosing future actions based on other factors, such as the PSR system's representation m and M. When encoding the behavioural rules into the ideal dynamics matrix, the natural system's tendencies must be taken into account. In some domains, this influence is weak compared to the encoded rules, and can be disregarded. In other domains, the influence is strong and behavioural rules contradicting the system's natural tendencies must also be encoded into the ideal dynamics matrix in order to obtain clear results. An example of natural system tendencies was observed in the Network Domain, Section 3.5.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a new instantiation of the DBC control framework, namely the EMT-PSR algorithm. It has provided a novel way for planning in PSR-modelled environments using the concept of an *ideal dynamics matrix*. By combining the DBC approach and the PSR environment, we have achieved a tighter relationship between the environment and the controlling algorithm, since both use only actions and observations.

We have demonstrated the performance and viability of the algorithm on a number of small domains that were represented by PSR models. Several of the environment models were constructed by conversion from other modelling techniques using the procedure from [6]. For each of the domains, an ideal dynamics matrix was constructed based on domain-specific properties of the control task. These control task properties were first expressed as behavioural rules describing the system motion through the next two steps of development, and then numerically encoded into the ideal dynamics matrix. In all the domains, the algorithm showed good performance trends.

From the experimental data we obtained, we were able to provide some insight into the DBC framework's operation, including the treatment of multi-dimensional observable quantities, and multiple reset actions during the formal modelling of the domains. Some experiments have suggested that the automated discovery and utilisation of the intrinsic system dynamic properties can be established, and it is a part of our future work to do so.

We also plan to explore the possibility of minimising the necessary set of behavioural rules for the construction of the ideal dynamics matrix, and to consider the improvement of the algorithm with respect to multiple behavioural trends along the lines of [14].

8. REFERENCES

[1] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 183–188, 1992.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[3] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *The 11th Int. Conference on Machine Learning*, pages 157–163, 1994.

[4] M. L. Littman. *Algorithms for sequential decision making*. PhD thesis, Brown University, 1996.

[5] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 394–402, 1995.

[6] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representation of state. In *Advances in Neural Information Processing Systems (NIPS-14)*, pages 1555–1561, 2001.

[7] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1–2):5–34, July 2003.

[8] R. A. McCallum. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the 10th International Conference on Machine Learning*, 1993.

[9] K. P. Murphy. A survey of POMDP solution techniques. Technical report, University of California at Berkeley, 2000.

[10] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *The 14th Int. Joint Conference on Artificial Intelligence*, 1995.

[11] M. L. Puterman. *Markov Decision Processes*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. Wiley-Interscience, New York, 1994.

[12] Z. Rabinovich and J. S. Rosenschein. Extended Markov Tracking with an application to control. In *The Workshop on Agent Tracking: Modeling Other Agents from Observations, at AAMAS'04*, pages 95–100, New-York, July 2004.

[13] Z. Rabinovich and J. S. Rosenschein. Multiagent coordination by Extended Markov Tracking. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 431–438, July 2005.

[14] Z. Rabinovich and J. S. Rosenschein. On the response of EMT-based control to interacting targets and models. In *The 5th Int. Joint Conference on Autonomous Agents and Multiagent Systems*, pages 465–470, May 2006.

[15] Z. Rabinovich, J. S. Rosenschein, and G. A. Kaminka. Dynamics based control with an application to area-sweeping problems. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 785–792, Honolulu, Hawaii, May 2007.

[16] S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *The 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 512–519, 2004.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. The MIT Press, 1998.