Abstract

Optimal behaviour policies for non-degenerate Markov Decision Processes (nMDPs) can be intractable to find, when state space size is prohibitive. In this paper, a theoretical method of deconstruction nMDPs into a set of concurrent MDPs is demonstrated. Such concurrent representations may be degenerate in state space, and require exponentially less memory to store. In offset of the state space savings, it frequently believed that such degeneracies lead to sub-optimal policy regression. Surprisingly, given weak preconditions, policy regression for the degenerate Concurrent MDP sets (CMDP) can converge upon optimal policies for the coincident non-degenerate nMDPs. To illustrate this effect a theoretical exploration is followed by a simple localization and box pushing simulation. In validation of the position, regression approaches using Q-Learning did not yield statistically significant behaviour policies in terms of reward maximization, despite the fact that the degenerate CMDP required exponentially less state space to specify (pairwise p-value $<$ X.XXXX).

1 INTRODUCTION

In prior work Concurrent Markov Decision Processes were presented [], and shown via experiment to enable discovery locally policy when compared with nMDPs. Among some theoretical questions, the question of how a set of degenerate MDPs, with partial state spaces, could converge on identical or superior policies as compared with the non-degenerate MDP was left as future work. Indeed, for people, we routinely choose to represent partial truths to ourselves, or give attention to subsets of our sensory information, such that our daily tasks are simplified. Thus, future intelligent agents, including robots, may need to rely on similar coping mechanisms that broadly introduce state degeneracy into problem models for the sake of efficiency. The work on CMDPs demonstrated that this approach, of introducing state degeneracy, may have theoretical merits. This paper grounds the state degeneracy discovery in theory.

This paper exposes and explores some weak problem assumptions that are made when performing a split of a non-degenerate MDP into a set of Concurrent Markov Decision Processes. When these weak assumptions are met, it is demonstrated that identical or superior local policy regression is expected both theoretically and empirically. Thus, this work contributes this set of assumptions which may be validated when simplifying nMDP problems into CMDP problems.

This paper outlines a brief background related to state space representation in Section 1, before outlining the core CMDP theory in Section 2. Section 3 outlines how a simple version of Q-Learning can be used to contrast the state space representation experiment. In Section 4, results are demonstrated for a simple localization and box pushing experiment.

2 BACKGROUND

The idea of state space factorization is not novel, and falls broadly into three categories.

In all approaches, it is frequently assumed in the literature that models which are degenerate in state space will expectedly suffer in the quality of their local optima. Broadly speaking, in Machine Learning literature, this is sometimes expressed as the Bias-Variance trade off. Practically speaking, when creating degenerate MDPs, the question of *how degenerate* these MDPs may be until they become biased is sometimes considered to be a core question.

Thus in this paper we present a method, given what is believed to be a non-degenerate and intractable nMDP, to construct and converge on an optimal behaviour policy. The method includes splitting the nMDP into a tractable CMDP form. We demonstrate both analytically and empirically, that the regressed policies are identical in terms of local optimality, if certain conditions hold.

3 THEORY

First, it is shown that an nMDP can be split into sub dependent MDPs, denoted a CMDP. Afterwards a lemma is provided which may be reused when approaching nMDP deconstruction.

3.1 Definitions, Concurrent Markov Decision Processes

To begin MDP problem definition, we broadly consider the task of finding the optimal location in a discrete state space in $S = \mathbb{N}^N$, using the maximum value function $V : S \to \mathbb{R}$. To ensure the space S has an optimal element it is assumed that a certain rearrangement of the value image $V(S)$ monotonic.

Thus, to discover a solution to the problem, the help of an agent may be elicited, who has allowed actions in a set $A \subset \mathbb{N}$, for every instant $t \in \mathbf{T} \subset \mathbb{N}$. The convention that an agent chooses an action $a_t \in A$ for every state $s_t \in S$, before arriving in state $s_{t+1} \in S$ is followed. The probability $T(s_{t+1}|s_t, a_t)$ represents the true and observed transitional probability of the system, also described as the dynamics of the environment, which are assumed to be stably stochastic, unless started otherwise. The epoch $E_e$is a set of contiguous time indexes, with restrictions: $e_i \in \mathbb{N}$, $\bigcup_{e=1}^{\mathbb{N}} E_e = \mathbf{T}$, and $\bigcup_{e_1,e_2 \in \mathbb{N}} E_{e_1} \cap E_{e_2} = \{\varnothing\}$. We assume the size of the epochs increase $|e_i| < |e_{(i+1)}|$, meaning that over time more time instants are contained within each. It is assumed that some discoverable infinite horizon reward exists $R(s_t, a_t, s_{t+1})$ such that agents may attempt to "maximize" the value given by the trajectory, where the value of a state is given by $V(S)$.

$$V(s_t) = \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \left(R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})\right) \tag{1}$$

The goal is to (a) explore and map the quality $Q_t : S \times A \to \mathbb{R}$ for the space, such that (b) a behavioural policy $a_t = \pi_t(s_t, a_t)$ is partly characterized by the Q_t values of the state. Commonly, the naive optimal policy is given by π^*:

$$\pi^*(s_t) \leftarrow \arg\max_a Q_t(s_t, a) \tag{2}$$

The above MDP definition is frequently expressed as a tuple: $m_k = \langle S_k, A_k, T_k, R_k, \pi_k \rangle$, where the initial posed nMDP problem is noted as $m = \langle S, A, T, R, \pi \rangle$. The time index t, may sometimes be replaced with an epoch index e, in situations that an MDP $m_i$operates on a different discrete timescale than another MDP m_j.

3.1.1 Random Field Deconstruction Example

The problem of the random field with real values can be considered, $r : X^N \to \mathbb{R}^+$, $X \subset \mathbb{N}$, which we assume is well-ordered and monotonic. For brevity, we consider $N = 2$ as a visual example.

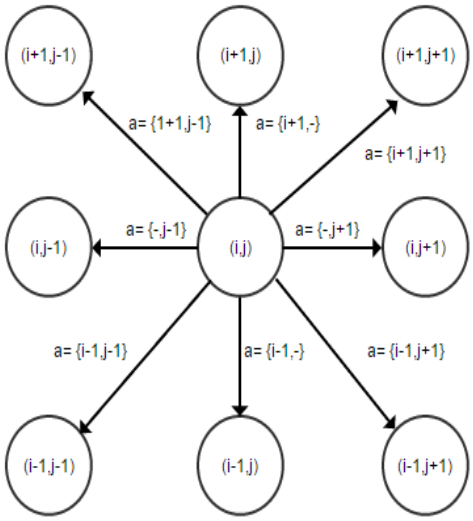


Fig. 1. The nMDP with transitions from (i, j)only.

3.1.1.1 An nMDP definition can be described:

- $S = X^N$ is the set of all possible states
- $A = A_u \times A_l$ s.t. $A_u, A_l = \{-1, 1, \varnothing\}$
  - with $a_u \in A_u, a_l \in A_l$
- $R\left(s_{t+1}|s_t, a_t\right) = r\left(l + a_l, u + a_u\right) - r(l, u)$
- $T\left(\{l + a_l, u + a_u\}|, \{a_l, a_u\}, \{l, u\}\right) = 1$, 0 otherwise
  In terms of notation the following equalities are always true: $s_{t+1} = \{i + a_u, j + a_l\}$, $a_t = \{a_u, a_l\}$, and $s_t = \{i, j\}$.

Thus, the problem for a reinforcement learning agent may be to localize and converge upon an optimal space s_t such that $r(s_t)$ is maximized. The agent can, without fail, move in any combination of “up”, “left”, ”down”, ”right”, and “nothing” to localize the best location. A discount reward and infinite time horizon are considered.

It turns out that the nMDP problem can be restated in terms of a degenerate manner using two models with degenerate state space. Colloquially, an agent can solve a degenerate CMDP problem of locating the best column and row independently.

3.1.2 An CMDP definition can be described

First, a *child* dependent MDP is defined

- $S_l = X^1, j$
- A_l

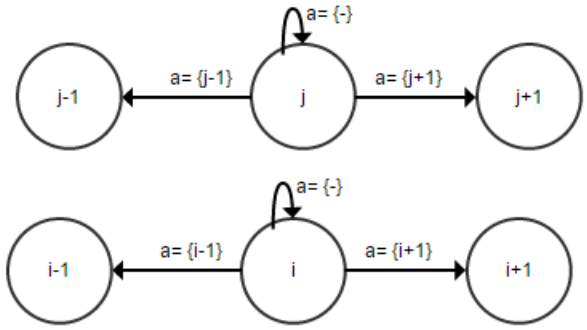


Fig. 2. Parent (a) and child (b) within the CMDP.

- $R_l\left(l+a_l|a_l,l,u\right)=r\left(l+a_l,u\right)-r(l,u)$
- $T_l\left(l+a_l|a_l,l,\right)=1,0$ otherwise
- The sequence index t is used for the child MDP.
- $M_l=\langle S_l,A_l,T_l,R_l\rangle$

Second, a *parent* dependent MDP is defined

- $S_u\ =X^1,i$
- A_u
- $R_u\left(u+a_u|a_u,u,E_e\right)=R_u\left(s_{u,e+1}|a_{u,e},s_{u,e},E_e\right)$
  - Noting $R_u\left(s_{u,e+1}|a_{u,e},s_{u,e},E_e\right)=\sum_{t\in E_e}\frac{R_l(s_{l,t+1}|a_{l,t},s_{l,t},s_{u,t})}{\overline{\overline{E}}_e}-\sum_{t\in E_e^o}\frac{R_l(s_{l,t+1}|a_{l,t},s_{l,t},s_{u,t})}{\overline{\overline{E}}_e^o}$
- $T_u\left(u+a_u|a_u,u\right)=1,\ 0$ otherwise
- $M_u=\langle S_u,A_u,T_u,R_u\rangle$
-
- The sequence index e is used for the parent MDP.

For notation, the equalities are always true: $s_{x,t+1}=x+a_x$, $a_{x,t}=a_x$, and$s_{x,t}=x$,$a_{x,t}$, where $x\in\{l,u\}$, where a time index t is substituted with and epoch e for the parent MDP. The notation $\overline{\overline{E}}_e$ represents the cardinality of an epoch, and is variable in $\mathbf{T}$.

Thus, as mentioned in prior [],two MDPs can form an interdependent set, such that the action a_e of the *parent* MDP describes reachable state space of a *child* MDP, though restricting state space exploration by defining variable u. Conversely, the child MDP's policy $\{a_t=\pi_t(s_t)\}_{t\in E_e}$ over all timesteps defines the reward obtained $R_u\left(s_{e+1}|a_e,s_e,E_e\right)$. A block diagram can illustrate these dependencies.

3.1.3 Policy Convergence

We consider elementary q-learning: ELEMENTARY Q-LEARNING

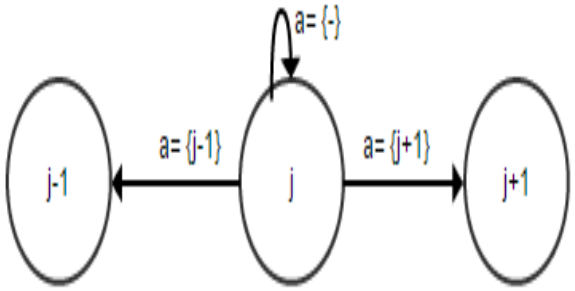


Fig. 3. Block diagram of a CMDP.

In order for convergence to be ensured $t \to \infty$, or $e \to \infty$, two conditions must be met: (a) an agent must visit each state $s \in S$ infinite times, (b) an agent's learning rate α must satisfy CONVERGENCE CRITERIA, and (c) the Reward function and transitions probabilities T, and R must be stably stochastic.

A rudimentary action selection model π_t can involve sampling the density function

$$\pi\left(s\right) \sim \left(s, a\right) = \frac{e^{Q_{t-1}(s,a)}}{\sum_{b \in A} e^{Q_{t-1}(s,b)}} \tag{3}$$

Which can be shown to converge readily for an nMDP which satisfies the three above conditions [].

3.1.4 Convergence

We now briefly show convergence of both the parent and child process based on π_t. Past interest, nothing is lost by moving to section XXX. In general, optimal policies will be denoted $\pi^*(s) \in \Omega^*(s)$, where Ω^* is the set of all optimal policies, and the action selection policies $\Omega^*(s)$ represent the set of optimal action selection policies for a state $s \in \mathbf{S}$.

3.1.4.1 Child MDP Convergence on optimal policy: For the child MDP, it is clear that the epoch $e \to \infty$ duration increases as $t \to \infty$. Thus we have:

$$\begin{aligned} \lim_{e \to \infty} \overline{\overline{E}}_e &= \infty \\ \text{and} \qquad \pi_l\left(s_l, a_l\right) &> 0 \end{aligned} \tag{4}$$

We also know that, given all epoch e, $\pi_u\left(s_u, a_u\right) > 0$. Thus, all states are reachable and infinite visitations at time infinity are assured. For a sub space S_l, for the child MDP, we note that the Q-Values $Q_t(s_l, a_l)$ will be reset to initialized values upon transition to every new epoch $e_{i+1} \neq e_i$; in this case, since $\overline{\overline{E}}_e \to \infty$, we are assured that policy convergence is possible. The practical problem of resetting the Q-Values in a non-optimal way is addressed in the implementation section, but in the worst case, (with resetting values,) eventual convergence is theoretically assured.

3.1.4.2 Parent MDP Convergence on optimal policy: Convergence for the parent MDP is apparent, as the reward function $R_u\left(s_{e+1}|a_e, s_e, E_e\right)$ is not only stably stochastic, but also increasingly reflective of the mean

reward expected from a learning agent given a target row j. Since $\pi_t(s_u, a_u) > 0$ and $e \to \infty$, with a stable transition model T_u convergence is assured.

3.1.4.3 Mapping Disjoint CMDP Policies onto an optimal nMDP Policy: Having two convergent sub optimal policies (π_u, π_l) is a promising initial step, but may not contribute to the solution for an optimal global policy π . It turns out that it can under some weak conditions, outlined below, such convergence can be shown readily. Specifically we wish to map the set two sub optimal policies onto an optimal nMDP policy:

$$f : \Omega_u^*(S_u) \times \Omega_l^*(S_l) \to \tilde{\Omega}^*(S) : \tilde{\Omega}^*(S) \subseteq \Omega^*(S) \tag{5}$$

For the above MDP problem, it can be readily demonstrated that such a mapping function f can be found, given the above CMDP problem. Specifically we must consider the nature of optimal policy for both the parent and child MDPs. We begin with the definition of an optimal value given a state s:

$$\Omega^*(s) = \arg\max_a \sum_{s' \in S} P(s'|s,a)\left(R(s,a,s') + \gamma V(s')\right) \tag{6}$$

And, it follows directly that a specific and optimal global policy can be expressed as every combination of optimal CMDP policies, $\pi^*(s) = \pi_u^*(s) \cup \pi_l^*(s)$. For details on this detailed derivation, please see Convergent Factors Lemma in the as the appendix. We also, then, know that execution of the regular action policies will lead the Q-values to converge on these optimal policies.

$$\pi(s) = \{\pi_u(s), \pi_l(s)\} \tag{7}$$

This technique can factor exponentially large state space problems into polynomial space, rendering intractable problems as tractable. With the addition a dynamic programming approach, the convergence speed can be optimized, trading off storage space for convergence speed.

3.1.5 Simulation

For simulation, we tackle the elementary two dimensional $S = X^2$, where $X = \{1, 2, \ldots, 100\}$. Each cell is assigned a valued $V(s \in S) \in \{1, \ldots, 1000\}$. We contrast a nMDP approach, and its convergence time, with two CMDP approaches, with all three described below.

3.1.6 Example Derivation of multi-agent robotics Application

After initial consideration of the nMDP problem, it can be illuminating to consider how a multi-agent MDP can be expressed as a CMDP problem. In general, when multi-agent MDPs are formed, they are approached from two directions. Either the MDP representation is centralized, meaning that a single nMDP contains all information across all agents, or each individual agent possesses a subset of the compelte MDP problem, sometimes called decentralized MDPs.

Decentralized MDPs commonly express degeneracy when compared with their original problem. Importantly there has been work on decentralized MDPs whose formulation allow seeming convergence on optimal solutions to related nMDP problems. However, the theoretical accuracy of policy convergence for multi-agent CMDP problems has been elusive.

Thus, the following derivation demonstrates how the general CMDP approach can be used to derive a multi-agent CMDP for a mult-agent robotics simulation, such that policy convergence for the multi-agent CMDP can be identical to the intractable nMDP counterpart.

3.1.7 Centralized nMDP

The centralized nMDP is defined as a decentralized MDP that may consist of a set of states S_U a set of system actions for each agent $A_U = \times_{i=1}^{n} \{A_I \cup A_T\}$, a real valued reward function R_U, and a stably stochastic transitional model T_U. The decentralized MDP is used in this paper as a control to contrast with other learning models.

The value of the reward R_T and transition functions T_T are assumed to be unpredictable up to some known discrete iteration t_s. After t_s the task allocation process's reward and transition function are assumed to have a constant expectation value, which may be affected by some unknown amount of zero mean noise.

It is worth emphasizing that within the Concurrent MDP model the two individual performance and task allocation processes explored concurrently. This concurrent learning approach distinguishes the model from a single

3.1.8 Task Allocation MDP

Team progress toward a goal can be defined as a $\langle S_T, A_T, T_T, R_T \rangle$ tuple, where$S_T \subset S_U$,

- S_T denotes a discrete set of states, which capture the individual performance characteristics between all agents and all tasks.
- A_T denotes a discrete set of actions, i.e., a function that assigns all available tasks to available agents.
- $T_T \left(s_T, a_T, s_T^{'} \right)$ denotes a stable transition model, i.e., the probability of executing action a_T starting from state s_T and ending up in state $s_T^{'}$. The transition function is completely specified by the individual performance process through a set of evidence E_{TI}.

3.1.9 Individual Performance MDP

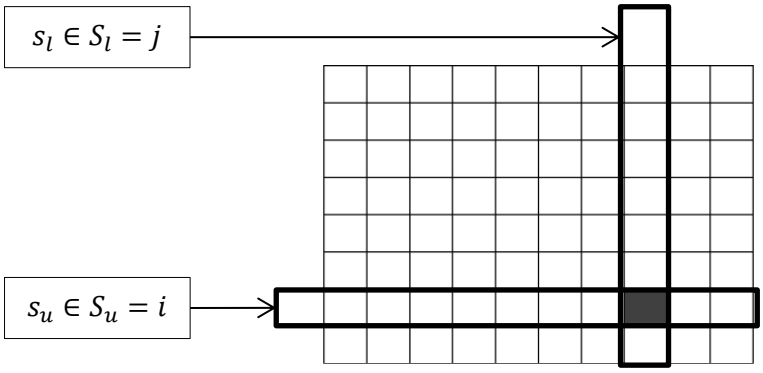
Individual agent progress toward a sub-task can be defined as a $\langle S_I, A_I, T_I, R_I \rangle$ tuple, $S_I \subset S_U$,

- S_I denotes a discrete set of states, whose intrinsic value is partially specified by the task allocation process through a set of evidence E_{IT}.
- A_I denotes a discrete set of actions.
- $T_I \left(s_I, a_I, s_I^{'} \right)$ denotes a stable transition model, i.e., the probability of executing action a_I starting from state s_I and ending up in state $s_I^{'}$.
- $R_I(s_I, a_I, s_I^{'})$ denotes a positive real number as a reward received for transitioning from state s_I into state $s_I^{'}$ using action a_I.

The evidence value $e_{\text{IT}} \in E_{\text{IT}}$ is defined by the task allocation MDP, such that maximal visitation of all states at time infinity is assured. Such a single-agent MDP can be seen as a straightforward process for the reinforcement learning.

Given these definitions, the derivation of a multi-agent CMDP can occur in three steps:

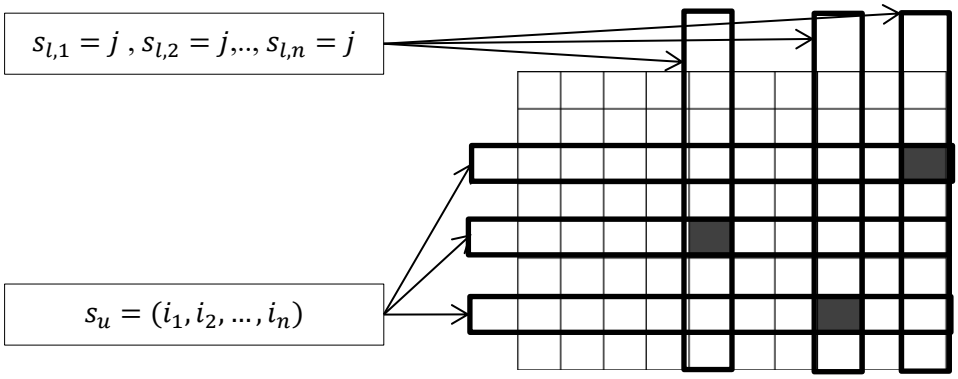
One, we can imagine the initial situation as our example problem for a single agent, with a child MDP, $\langle S_l, A_l, T_l, R_l \rangle$ and parent MDP $\langle S_u, A_u, T_u, R_u \rangle$ depicted on a grid. In this situation the reward and transition functions are defined in accordiance with section 3.0. In this arrangement, the parent MDP will have to try many rows, letting the child MDP visit an increasing amount of column states per row, as described in section 3.0.



The first alteration that can be made, is to convert the problem into multi-agent MDP; the problem is constructed of a parent MDP and a set of independent children MDPs. The parent MDP's definition can be expanded in the following manner:

- $S_u = X^1, X^n$
- $A_u = \{-1, 0, 1\}^n$
- Transition
- $R_u\left(u + a_u | a_u, u, E_e\right) = \sum_{i \in \{1,\dots,n\}} R_{u,i}\left(s_{u,e+1} | a_{u,e}, s_{u,e}, E_e\right)$
  - Noting $R_{u,i}\left(s_{u,e+1} | a_{u,e}, s_{u,e}, E_e\right) = \sum_{t \in E_e} \frac{R_l(s_{l,t+1} | a_{l,t}, s_{l,t}, s_{u,t})}{\overline{\overline{E}}_e} - \sum_{t \in E_e^o} \frac{R_l(s_{l,t+1} | a_{l,t}, s_{l,t}, s_{u,t})}{\overline{\overline{E}}_e^o}$

Expanding the state by a factor of n allows for the parent MDP to regress to an optimal policy over n agents with an exponential increase in both state space and action space. However, this state space expansion is bounded by $O(|X^n|)$ whereas the multi-agent nMDP space space requirements are bounded by $O(|X^n|\,|X^n|\,n)$.



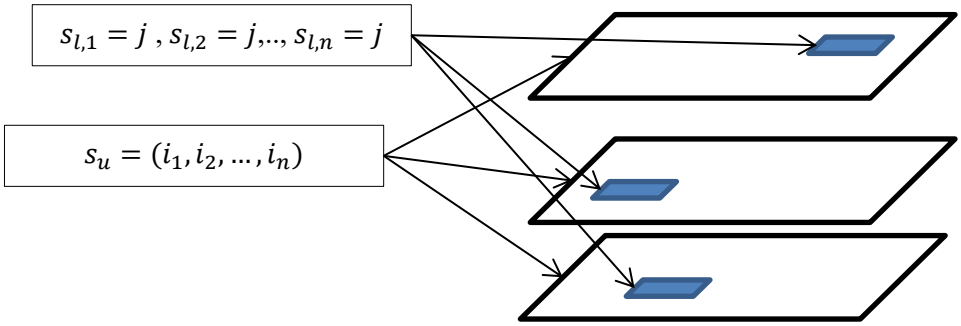
3.1.10 Convergence

We can analyze policy convergence for both the *multi-agent child* MDPs, and the *multi-agent parent* MDP: First, the *multi-agent child* MDPs remain unchanged, so given sufficient policy execution time, their convergence is

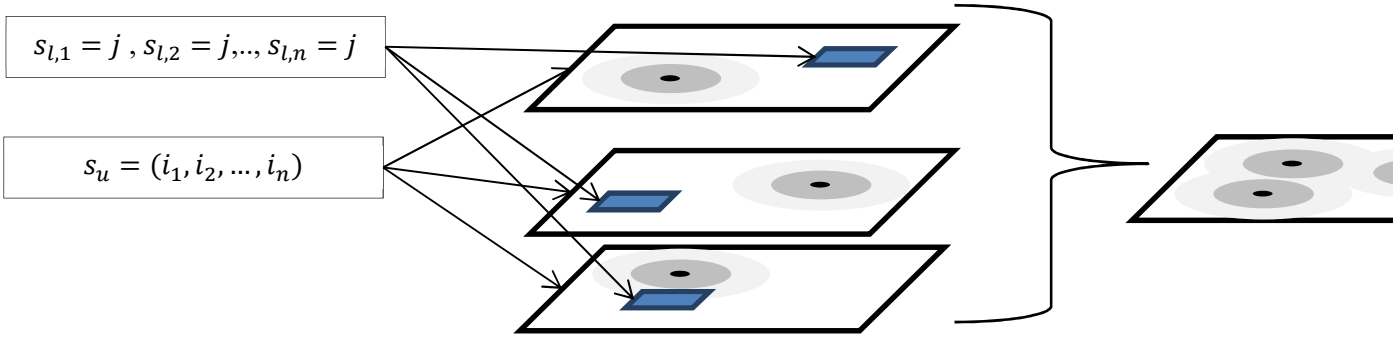
assured. Second, the *multi-agent* parent MDP converges similarly as shown in section 1: The state space, action space, and transition space have increased in dimension, which does not affect convergence[]. The Reward, although now a sum over the agents, is a sum over a set of stably stochastic functions, which is itself stably stochastic[].

3.1.11 complete multi-agent CMDP

Thus, we can imagine the current problem as a search for the highest cell numbers in a grid spread over number (n) of columns, where the number of total columns is unlimited. It is natural, at this point, to expand the *multi-agent child* MDP into a *complete multi-agent child* MDP, by solving a two dimensional row problem:



This division of space can be readily generalized to a foraging problem, where each layer i is assigned reward such that reward increases as a foraging targets location is nearer:



3.1.12

3.1.13 Convergence

The complete model differs only in that state space is larger, and that the model must contain a definition for a series of k sets, $k \in \{X, X\}^n$. If we assume the k values set for any experiment, we see that the only difference in models is that the state space has increased in dimension. Aside from an increase in computational and storage space burden, convergence in increased state spaces is assured.

Thus, using this methodology, many general multi-agent problems can be derived from the initial situation given within section 3.0, including the assurance that, as time increased policy uncertainty asymptotically decreases.

This remodelling, however, requires a human expert to complete. It may be more enticing to consider how CMDP models may reconfigure dynamically to solve problems, and how such models can be regularized. The

remainder of this paper considers how multiple approaches may be derived and combined to create a general and regularized reinforcement learning framework.

4 General Reinforcement Learning

In the following section, the approach to finding a policy for a singular MDP, $M = \langle S, A, T, R \rangle$, is generalized; instead of relying on a human to decide how to structure a CMDP, as described in sections 3.0 and 3.1, it may be desired that a reinforcement learning mechanism learn its own structure. This learning mechanism is inspired by all domans of research not limited to regularized neural networks and projected component analysis[]. In both of these domains models learn generative state space representations by initially (1) isolating principal components that information can be projected onto and following to (2) analyze and predict outcomes using the independent subspaces noting covariance information. These methods sometimes do not consider (3) the usage of information to make predictions of drive actions of an agent within their environment. Thus, in the following sections we construct a model that can automatically analyze the components of state and action space, fulfilling (1) and (2), where this method is tied to the performance of (3) the dynamic CMDP's ability to both exhibit reduced space consumption and have increased decision making and policy convergence behaviour.

The mechanism is formed using several sub behaviours. First, an nMDP can be broken into a CMDP, which we know converges at time infinity. For this decomposition and recomposition to be genrally applied, we need a method to map Transitional and Policy information between similar models. This is addressed in section 4.1 Learning and Mapping Transitional Knowledge. Afterward, knowing that nMDP models and CMDP models can be continually decomposed and recomposed, an approach to expressing the decomposition/composition behaviour as a reinforcement learning problem is expressed, in section 4.2 as Dynamic Reconfiguration of Concurrent Markov Decision Processes.

4.1 Learning and Usage of Transitional Knowledge

In this section we consider the benefits and drawbacks of modelling the regressed Q values directly for a behavioural policy. The classic initial approach is to discover Q.

$$Q : S \times A \rightarrow \mathbb{R}^{+} \tag{8}$$

Such that it is elementary to derive the behavioural policy online:

$$\pi^{*} : S \rightarrow A \tag{9}$$

Where a common policy definition is commonly used in casual experimentation, $\pi^{*}(s) = \arg\max_{a} Q(s, a)$.

The calculation of Q has been traditionally computed online using equation (XX), repeated for reference:

$$Q(s, a)^{'} \leftarrow Q(s, a) + \alpha \left(R\left(s, a, s^{'}\right) - Q\left(s, a\right) + \max_{a^{*}} Q(s^{'}, a^{*}) \right) \tag{10}$$

Where the symbol (') represents a function or variable value at the following time step. The procedure of executing Q-Learning (Q_l) computes the a map,

$$Q_l : R(S, A, S) \times T(S, A, S) \rightarrow Q(S, A) \tag{11}$$

, such that neither the complete reward ($R(S,A,S)$) or transition ($T(S,A,S)$) images are kept in memory; the QL algorithm functions online. The purpose of this section is to, first, provide a mechanism that stores and calculates Q values based on learning approximate characterizing functions, for transition $\tilde{T}(S,A,S) \approx T(S,A,S)$ and reward $\tilde{R}(S,A,S) = R(S,A,S)$, by executing transition learning and reward learning, T_l, R_l. Second, this section outlines a method to recover Q values from these images anytime, using *transitional learning*, Q_{tr}:

$$\begin{aligned} T_l &: \tilde{T}(S,A,S) \times \left\{ \left(s,a,s^{'} \right) \right\} \rightarrow \tilde{T}^{'}(S,A,S) \\ R_l &: \tilde{R}(S,A,S) \times \{(s,a,s^{'})\} \rightarrow \tilde{R}^{'}(S,A,S) \\ Q_{\text{tr}} &: \tilde{T}(S,A,S)^{'} \times \tilde{R}(S,A,S)^{'} \rightarrow Q^{'}(S,A) \end{aligned} \tag{12}$$

Thus, it is expected that the basis functions T_l and R_l are computed online, whereas the current Q-Values can be updated as needed using a process Q_{tr}, to be outlined.

The T_l and R_l functions can be discovered, in the following manner, through the tracking of local visitation frequencies:

$$T_l : \tilde{T}'(s,a,s') \leftarrow \tilde{T}(s,a,s') + \alpha \left(\frac{\text{fr}\left(s,a,s^{'} \right) + 1}{\text{fr}(s,a) + 1} - \tilde{T}(s,a,s') \right)$$

$$\frac{\text{fr}\left(s,a,s^{'} \right) + 1}{\text{fr}(s,a) + 1} = \frac{1 + \tilde{T}(s,a,s')}{1 + \sum_{s'} \tilde{T}(s,a,s')}$$

The Q_{tr} function can be defined using one of three regression functions, each with benefits and drawbacks, where ξ represents a collection of sample states action state tuples, $\xi(s,a) \subseteq S \times A \times S$, where a and b are constants:

- A greedy algorithm, which does not value any future reward.

$$Q^{'}(s,a) \leftarrow \frac{1}{\sum_{s' \in \xi(s,a)} \tilde{T}(s,a,s')} \sum_{s' \in \xi(s,a)} \tilde{T}(s,a,s') \tilde{R}(s,a,s') \tag{13}$$

- A complete algorithm, which recursively explores potential for future reward:

$$Q^{'}(s,a) \leftarrow \frac{1}{\sum_{s' \in \xi(s,a)} \tilde{T}(s,a,s')} \sum_{s' \in \xi(s,a)} \tilde{T}(s,a,s') \tilde{R}(s,a,s') + \gamma \max_{a^*} Q(s^{'}, a^*) \tag{14}$$

This approach is the most complete, but runs into issues related to recursive state space explosion.

- A monte carlo algorithm, which runs a batch monte carlo simulation to update policy values:

$$Q^{'}(s,a) \leftarrow Q(s,a) + \frac{\alpha}{n} \sum_{i=0}^{\text{n}} \tilde{R}(s,a,s') - Q(s,a) + \gamma \arg\max_{a^*} Q(s^{'}, a^*) \tag{15}$$

, where s' is sampled from the density $\tilde{T}(s,a,s')$. It is trivial to note that the monte carlo simulation approaches the value obtained by the complete algorithm as the limit of n approaches infinity.

Thus, using elementary techniques it is tractable to regress to transitional and reward models online, and it is furher possible to extract policy values online using at least three algorithms.

4.2 Transferring Knowledge

Given an nMDP, it may be desired to split or reformulate a problem into a CMDP. In this circumstance, given transition and reward data encoded in a manner according to section 4.1, the next step is to consider how a knowledge transference model may be constructed. For the following section we will generalize the notion of state action result sets to the notion of trajectories such that $l_i^j = \left\{(s_i, a_i, s_i^{'})\right\}_i^j$.

Using this notation $\tilde{T}(s, a, s') = P\left(s^{'}|s, a\right) = \tilde{T}\left(l_i^{i+1}\right)$ where $l_i^{i+1} = \left\{(s_i, a_i, s_i^{'})\right\}_i^{i+1}$. The value for a transition sequence is a set of values $\tilde{T}\left(l_i^j\right) = \left\{\tilde{T}\left(l_i^i\right)\right\}_i^j$. Lastly, to simplify analysis, we define a function $\tilde{F} \in \{\tilde{T}, \tilde{R}\}$. We also recall $m_k =< S_k, A_k, T_k, R_k, \tilde{T}_k, \tilde{R}_k, \pi_k >$.

4.2.1 Preliminary Deconstruction and Reconstruction

The goal is to describe a deconstruction $d_x^{\text{yz}}$and reconstruction c_{yz}^xfunction, which maintain all of the properties outlined in sections 3.0. This section begins by using the CMDP via a gating function and regression is discussed.

Decision Making and Gating

Before this, however, it is prudent to consider how a behaviour policy π_x can be mapped from a degenerate behaviour policy set $\{\pi_y, \pi_z\}$. Simply, we allow for definition of a surjective confidence function $g_x : S \times \pi_y(S) \times \pi_z(S) \rightarrow \mathbb{R}^+$. Where, typically, $\pi_x(s, a) = g_x(s, \pi_y(s), \pi_z(s))$. It seems obvious at this point to formulate this gating function as the result of a reinforcement learning mechanism.

Naïve Deconstruction and Reconstruction

$$d_x^{\text{yz}} : m_x \rightarrow \left\{m_y, m_z |\ S_y \times S_z = S_x,\ A_y \cup A_z = A_x,\ \tilde{c}_{\text{yz}}^x\left(\tilde{d}_x^{\text{yz}}\left(\tilde{F}\right)\right) = \left(\tilde{F}\right)\right\} \tag{16}$$

$$c_x^{\text{yz}} : \{m_y, \} \rightarrow \left\{m_x |\ S_y \times S_z = S_x,\ A_y \cup A_z = A_x,\ \tilde{c}_{\text{yz}}^x\left(\tilde{d}_x^{\text{yz}}\left(\tilde{F}\right)\right) = \left(\tilde{F}\right)\right\} \tag{17}$$

Thus, splitting and merging are invertible processes. Importantly, the character and performance of any chosen reconstruction sets $\left\{d_x^{\text{yz}}, c_{\text{yz}}^x\right\}$ are not implied by usage of the functions. It can be illuminating to define some entry level deconstruction set implementations for an nMDP:

Example Naïve Deconstruction

A deconstruction implementation is provided: d_x^{yz}. It is direct to see that this implementation of d_x^{yz} satisfies the axioms given in section 3.0.

- State

  A set of degenerate state spaces $\{S_y\ , S_z\}$ can be defined using principle component analysis on a parent state space S_x , being careful to include all dimensions from S_x in either S_y or S_z.
- Action

The action set $\{A_y\ , A_z\}$ can each be formed using a random selection of actions from A_z. If actions are missing, ($\{A_y \cup A_z\} \setminus A_x \neq \varnothing$,) then the missing actions can simply be added, $A_y \leftarrow A_y \cup \{A_x \setminus \{A_y \cup A_z\}\}$.

- Transition

  Note: $P\left(\left\{s_z^{'}, s_y^{'}\right\} | \{a_z, a_y\}, \{s_z, s_y\}\right) = P\left(s_x^{'} | a_x, s_x\right)$

  The parent transition function $\tilde{T}_x$ can be projected onto subspaces $\tilde{T}_y$ and $\tilde{T}_z$. It is direct to note through application of the chain rule that a probability density describing transition can be broken into sub constituent distributions$\backslash n$ $P\left(s_x^{'} | a_x, s_x\right) = P\left(s_y^{'}, s_z^{'} | a_x, a_y, s_y, s_z\right)$. Thus, In our current notation this notion can be stated briefly:

$$\tilde{T}_y\left(s_y^{'} | s_y, a_y\right) = \sum_{s_z^{'}} \sum_{a_z} \sum_{s_z} \tilde{T}_x\left(\left\{s_y^{'}, s_z^{'}\right\} | \{s_y, s_z\}, \{a_y, a_z\}\right) \frac{\pi_x\left(a_y, a_z | s_y, s_z\right)}{\sum_{a_z^{'}} \pi_x\left(a_y, a_z^{'} | s_y, s_z\right)} \tag{18}$$

It is worth noting that if we split the MDPs according to section 3.0, specifically such that one MDP is the parent and the other is the child, then we can reuse the *Convergent Factors Lemma(*appendix). Specifically the transition functions simplify:

For the child, z, the parent will always hold its action a_z constant, rendering its behavioural policy irrelevant for the purposes of projecting transitional probability.

$$\tilde{T}_z\left(s_z^{'} | s_z, a_z, a_y\right) \leftarrow \sum_{s_z^{'}} \tilde{T}_x\left(\left\{s_y^{'}, s_z^{'}\right\} | \{s_y, s_z\}, \{a_z\}, a_y\right) \tag{19}$$

For the parent, y, the transition model must take into account a sequence of actions undertaken

$$\tilde{T}_y\left(s_y^{'} | s_y, a_y\right) = \sum_{s_z^{'}} \sum_{a_z} \tilde{T}_x\left(\left\{s_y^{'}, s_z^{'}\right\} | \{s_y, s_z\}, \{a_y, a_z\}\right) \pi_z\left(a_z | s_z\right) \tag{20}$$

- Reward
- Policy

It becomes apparent that the degenerate policy $\pi_y\left(a_y | s_y\right)$ is unknown, and needs to be inferred from the complete policy $\pi_x\left(a_x | s_x\right)$. Briefly a direct mapping can be considered which is a projection of $\pi_x(\bullet)$ onto $\pi_y(\bullet)$, where σ is a normalization constant:

$$\pi_x\left(a_y | s_y\right) = \sigma \sum_{s_z^{'}} \sum_{s_z} \sum_{a_z} \tilde{T}_x\left(\left\{s_y^{'}, s_z^{'}\right\} | \{s_y, s_z\}, \{a_y, a_z\}\right) \pi_x\left(\{s_y, s_z\} | \{s_y, s_z\}\right) \tag{21}$$

In essence, the degenerate policy image $\pi_x\left(A_y | S_y\right)$becomes the expectation of the parent policy $\pi_x\left(A_x | A_x\right)$ weighted by the transitional expectation, and this can be taken as the $\pi_y\left(a_y | s_y\right) = \pi_x\left(a_y | s_y\right)$.

After policy is initialized, the behavioural policy can be updated using Q-Learning or any other mechanism.

Example Naïve Reconstruction

A reconstruction implementation is provided: c_x^{yz}. It is direct to see that this implementation of c_x^{yz} satisfies the axioms given in section 3.0.

Similar to deconstruction, reconstruction steps can be expressed direction as a mapping of $\langle S, A, T, R \rangle$ tuples:

- State
- $S_x = S_y \times S_z$
- Action
- $A_x = A_y \cup A_z$
- Transitional probability
- Knowing $s_x^{'} = \left\{ s_y^{'}, s_z^{'} \right\}$
- $\tilde{T}_x \left(\left\{ s_y^{'}, s_z^{'} \right\} | \left\{ s_y, s_z \right\}, a_x \right) = \tilde{T}_y \left(s_y^{'} | s_y, a_x \right) \tilde{T}_z \left(s_z^{'} | s_z, a_x \right)$
- Reward
-
- Policy
-

Concurrency and re-mapping

Given the state space savings given by a degenerate formulation, it may be desired to only represent nMDPs in their degenerate formats. Thus, in general,

4.2.2 Compensation for temporal scale difference

(page 88)

4.2.3 Compensation for action space factorization

Compensation for action space factorization

5

6 SIMULATION

6.1 Experiment Two

7 RESULTS

8 CONCLUSION

9 REFERENCES

10 APPENDIX: CONVERGENT FACTORS LEMMA

Note

The set of optimal policies $\Omega^* (i, j)$ can be expressed as a union of two independent policies, $\left(\Omega_u^* (j), \Omega_l^* (i|j) \right)$, for problems in the presented CMDP's class. For this proof, i and j are taken to be two subspaces S_I, S_J within a parent state space S, such that a bijective function exists, $f_s, : \ S_I \times S_J \rightarrow S$. We assume separable actions for with another bijective map, $f_a : \ A_I \times A_J \rightarrow A$. It's worth noting that in practicality, transformations such as

affine projections and arbitrary component projections, can be considered. To begin we can consider the set of all optimal actions that maximize expected future discounted reward.

$$\Omega^{*}(a)=\arg\max_{a}\sum_{s'\in S}P\left(s'|s,a\right)\left(R\left(s,a,s'\right)+\gamma V\left(s'\right)\right) \tag{22}$$

The argument is that there exists a mapping from two optimal sub policies onto this policy.

$$f:\Omega_{u}^{*}\left(S_{u}\right)\times\Omega_{l}^{*}\left(S_{l}\right)\rightarrow\ \tilde{\Omega}^{*}\left(S\ \right):\tilde{\Omega}^{*}\left(S\ \right)\subseteq\Omega^{*}\left(S\ \right) \tag{23}$$

For the following analysis, consultation of a visual grid can be beneficial. To simplify notation, (i,j) is taken to equal ij, and actions $(a_u,a_l)=\ a_{\text{ul}}$.

$$\Omega^{*}(i,j)=\arg\max_{(a_u,a_l)}\sum_{(i',j')\in S}P\left(i'j'|ij,a_{\text{ul}}\right)\left(R\left(ij,a_{\text{ul}},i'j'\right)+\gamma V\left(i'j'\right)\right) \tag{24}$$

knowing $\max_{(a_u,a_l)}R\left(\left(i,j\right),\left(a_u,a_l\right),\left(i',j'\right)\right)=\max_{(a_u,a_l)}R\left(\left(i,j'\right),\left(a_u,a_l\right),\left(i',j'\right)\right)$ allows the expression to be altered with j', using the exchange lemma.

$$\Omega^{*}((i,j))=\arg\max_{(a_u,a_l)}\sum_{(i',j')\in S}P\left(\left(i',j'\right)|\left(i,j\right),\left(a_u,a_l\right)\right)\left(R\left(\left(i,j'\right),\left(a_u,a_l\right),\left(i',j'\right)\right)+\gamma V\left(\left(i',j'\right)\right)\right) \tag{25}$$

It then becomes clear that the reward dependency can be expressed in terms of R_l only:

$$\Omega^{*}((i,j))=\arg\max_{(a_u,a_l)}\sum_{(i',j')\in S}P\left(\left(i',j'\right)|\left(i,j\right),\left(a_u,a_l\right)\right)\left(R_l\left(j^{'}|a_l,i',j\right)+\gamma V\left(\left(i',j'\right)\right)\right) \tag{26}$$

At this point, it is possible to note that the Transition function may be decomposed $P\left(\left(i',j'\right)|\left(i,j\right),\left(a_u,a_l\right)\right)=P\left(i'|a_l,i\right)P\left(j'|a_u,j\right)$.

$$\Omega^{*}((i,j))=\arg\max_{(a_u,a_l)}\sum_{(i',j')\in S}P\left(i'|a_l,i\right)P\left(j'|a_u,j\right)\left(R_l\left(j^{'}|a_l,i',j\right)+\gamma V\left(\left(i',j'\right)\right)\right) \tag{27}$$

Leading to a summation decomposition:

$$\Omega^{*}((i,j))=\arg\max_{(a_u,a_l)}\sum_{i'\in S}P\left(i'|a_u,i\right)\left[\sum_{j'\in S}P\left(i'|a_l,i\right)\left(R_l\left(j^{'}|a_l,i',j\right)+\gamma V\left(\left(i',j'\right)\right)\right)\right] \tag{28}$$

It is also possible to take the difference between two reward functions, differentiated by i and i', to achieve a favourable formulation. This addition is possible because our formulation is considering an optimal action, (a_u,a_l).

$$\Omega^{*}\left(i,j\right)=\arg\max_{(a_u,a_l)}\sum_{i'\in S}P\left(j'|a_u,j\right)\left[\sum_{j'\in S}P\left(i'|a_l,i\right)\left(R_l\left(j^{'}|a_l,i',j\right)-R_l\left(j'\ |a_l,i,j\right)+\gamma V\left(i',j'\right)-\gamma V\left(i',j\right)\right)\right] \tag{29}$$

And, an optimal decision policy can be substituted in for action selection $\Omega^{*}\left(j|i\right)=\arg\max_{(a_u,a_l)}\sum_{j'\in S}\ P\left(j'|a_l,j\right)\left(R\left(j,a_l,j^{'}\right.\right.$

$$\Omega^{*}\left(i,j\right)=\underset{\left(a_{u},a_{l}\right)}{\arg\max}\sum_{i'\in S}P\left(i'|a_{u},i\right)\left[\sum_{j'\in S}P\left(j'|\Omega^{*}\left(j|i\right),j\right)\left(R_{l}\left(j^{'}|\Omega^{*}\left(j|i\right),i',j\right)-R_{l}\left(j\ |\Omega^{*}\left(j|i\right),i,j'\right)+\gamma V\left(i',j'\right)-\gamma V\left(i',j\right)\right.\right. \tag{30}$$

The formulation can now be radically generalized:

$$\Omega^{*}\left(i,j\right)=\underset{\left(a_{u},a_{l}\right)}{\arg\max}\sum_{i'\in S}P\left(i'|a_{u},i\right)\left[R_{u}\left(i^{'}|a_{u},j,i\right)+\gamma V\left(i'\right)\right]\qquad\text{if }a_{u}\in\Omega_{l}^{*}\left(j|i\right) \tag{31}$$

From here it is direct to union the row selection policy:

$$\Omega^{*}\left(i,j\right)=\sum_{i'\in S}P\left(i'|a_{u},i\right)\left[R_{u}\left(i^{'}|a_{u},i\right)+\gamma V\left(i'\right)\right]\cup\left[\Omega_{l}^{*}\left(j|i\right)\right]\qquad\text{if }a_{u}\in\Omega_{l}^{*}\left(j|i\right) \tag{32}$$

And substitute a definition only dependent on independent policies.

$$\Omega^{*}\left(i,j\right)=\Omega_{u}^{*}\left(i\right)\cup\Omega_{l}^{*}\left(j|i\right) \tag{33}$$

And, it follows directly that a specific and optimal global policy can be expressed as every combination of optimal CMDP policies.

$$\pi^{*}\left(i,j\right)=\pi_{u}^{*}\left(i\right)\cup\pi_{l}^{*}\left(j|i\right) \tag{34}$$

We also know that execution of the regular action policies will converge on these optimal policies.

$$\pi\left(i,j\right)=\pi_{u}\left(i\right)\cup\pi_{l}\left(j|i\right) \tag{35}$$