A Tensor-based Markov Decision Process Representation

Daniela Kuinchtner[0000−0002−3559−0384], Felipe Meneguzzi[0000−0003−3549−6168], and Afonso Sales[0000−0001−6962−3706]

Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, RS, 90619-900, Brazil
`daniela.kuinchtner@edu.pucrs.br`, {`felipe.meneguzzi`, `afonso.sales`}`@pucrs.br`

Abstract. A Markov Decision Process (MDP) is a sequential decision problem for a fully observable and stochastic environment. MDPs are widely used to model reinforcement learning problems. Researchers developed multiple solvers with increasing efficiency, each of which requiring fewer computational resources to find solutions for large MDPs. However, few of these solvers leverage advances in tensor processing to further increase solver efficiency, such as Google's TPUs[1] and TensorFlow[2]. In this paper, we formalize an MDP problem in terms of Tensor Algebra, by representing transition models of MDPs compactly using tensors as vectors with fewer elements than its total size. Our method aims to facilitate implementation of various efficient MDP solvers reducing computational cost to generate monolithic MDPs.

Keywords: Artificial intelligence · CANDECOMP/PARAFAC decomposition · Compact transition model · Markov decision process · Tensor algebra · Tensor decomposition.

1 Introduction

Markov Decision Processes (MDPs) are the underlying model for optimal planning for decision theoretic agents in stochastic environments [11]. Several works proved MDPs are useful in a variety of sequential planning applications where uncertainty is crucial to account in the process [15], such as Autonomous Robots [6, 4] and Machine Maintenance [1]. These concepts are essential for theory and algorithms of modern reinforcement learning [17, Ch. 1].

Examples of extensively developed methods for solving MDPs are Linear and Dynamic Programming algorithms [17, Chs 3 and 4]. The main problem with these solvers is in virtually any real-life domain the state space is large enough that solvers cannot compute policies for tabular representations of MDPs in reasonable time. This limitation is often referred to as *the curse of dimensionality*, since the number of states grows exponentially with the number of state

[1] https://cloud.google.com/tpu
[2] https://www.tensorflow.org/

variables [3]. By contrast, many large MDPs can be modeled compactly if their structure is exploited in the representation [11]. A method to represent large MDPs in a compact representation is called *Factored Markov Decision Process* (Factored MDP). Factored MDPs allow compact representations of complex uncertain dynamic systems [12] and often allow for an exponential reduction in representation complexity.

In this paper we propose a compact representation of the transition model of an MDP using tensor decomposition. Tensor decomposition aims to extract data from arrays, allowing to represent an array with fewer elements than its total size. Tensor decomposition is applied in several applications, such as: signal processing, computer vision, data mining, neuroscience, etc. [13]. In our particular method, we use a tensor decomposition called CANDECOMP/PARAFAC (CP), which decomposes a tensor as sums of rank-one tensors. Our method is to decompose the MDP problem into smaller tensor components, aiming to improve efficiency of MDP solvers.

2 Background

2.1 Markov Decision Process

Richard Bellman introduced Markov Decision Process (MDP) in 1957 [2]. An MDP is a sequential decision problem for a fully observable and stochastic environment. The way an agent transitions through an MDP is by sequential decision making, i.e., choosing actions which lead from one state to another. Reinforcement learning uses MDPs to model problems [16, Ch. 17, p. 647], where the agent's goal is to maximize the discounted expected reward over a sequence of actions. State transitions and decision making in MDPs are characterized by: i) a stochastic transition system, which determines probabilities to which state the decision making agent reaches after taking an action; and ii) the *Markov property*, which dictates every transition between states depends exclusively on the last visited state, rather than the history of states before that [17, Ch. 3, p. 49].

An MDP is formally defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ [17, Ch. 3, p. 48-49], where i) $\mathcal{S}$ is the state space; ii) $\mathcal{A}$ is the action space; iii) $\mathcal{P}$ is a transition probability function $\mathcal{P}(s' \mid s, a)$; iv) $\mathcal{R}$ is a reward function; and v) $\gamma \in [0..1]$ is a discount factor. More specifically, at each time step t the agent interacts with the environment by taking an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. As a consequence, the agent receives a reward r_{t+1} (also known as r') $\in \mathcal{R}$ and reaches a new state s_{t+1} (also known as s') $\in \mathcal{S}$ with probability $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ given the transition probability function [17, Ch. 3, p. 48]. A transition model describes the stochastic outcome of each action in each state, denoted by $\mathcal{P}(s_{t+1}|s_t, a_t)$, to determine the probability of reaching state s_{t+1} if action a is taken in state s [16, Ch. 17, p. 645-646].

The way an agent behaves in an environment is by following a policy, which maps states to actions. A policy is a solution for the MDP; an *optimal policy*

(π^*) maximizes the reward an agent receives over the long run. After taking an action at a state, the environment provides feedback to the agent in terms of an immediate reward [17, Ch. 1, p. 6]. The interaction with the environment terminates when the agent reaches one of the terminal states.

2.2 Tensor Algebra

Leopold Kronecker [14] proposed a tensor-based operation, called *Kronecker product*, represented by $\otimes$. The Kronecker product is used in Tensor Algebra, extension of Linear Algebra, allowing generalization of matrices where more than two dimensions can be represented.

Aiming to improve efficiency of MDP solvers, we propose a representation of transition models using *tensor decomposition*, extension of Tensor Algebra. Our method is to decompose the problem into smaller component elements to improve runtime of the solution.

Tensor Decomposition. The process that factorizes a tensor into sums of individual components is called CANDECOMP (canonical decomposition) and PARAFAC (parallel factors). This method provides a parallel proportional analysis and an idea of multiple axes for analysis [13]. We illustrate in Fig. 1 a CANDECOMP/PARAFAC decomposition of a third-order tensor and define three dimensions (i, j, k) of tensor $\mathcal{X}$ by $x_{ijk} = a_i b_j c_k$. The *order* of a tensor is the number of dimensions.

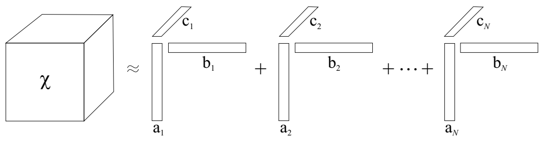

Fig. 1. CANDECOMP/PARAFAC decomposition example.

As the third-order tensor example $\mathcal{X} \in \mathbb{N}^{I*J*K}$ shows, where N is a positive integer and $a_n \in \mathbb{N}^I$, $b_n \in \mathbb{N}^J$, and $c_n \in \mathbb{N}^K$ for n = 1, ..., N; we can express the sum of individual components as:

$$\mathcal{X} \approx \sum_{n=1}^{N} (a_n \otimes b_n \otimes c_n). \tag{1}$$

3 Computational Cost of a Regular Method

Markov Decision Process with large state spaces arise frequently when applied to real world problems. Optimal solutions to such problems exist, but may not be computationally tractable, as the required processing scales exponentially with number of states. Related work contemplates comparisons of factored MDPs experimental results with traditional approaches. However, in order to compare a small example with our method, in this section we show the ineffectiveness of a regular method, called *Bellman's Operator Method*, to generate the stochastic outcome of state transition for a 4×3 grid (Fig. 2).

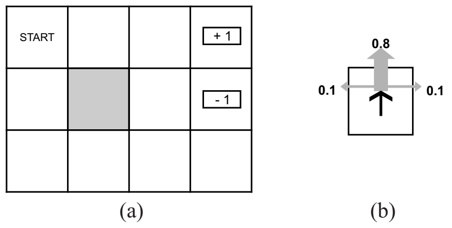

Fig. 2. (a) A simple 4×3 environment grid and (b) an illustration of the transition model of the environment. Adapted from [16, Ch. 17, p. 646].

In Fig. 2 the interaction with the environment terminates when the agent reaches one of the terminal states, marked by +1 or −1 rewards. The agent's actions in a given state, in this example, are *North*, *East*, *West* and *South*, where the expected outcome occurs with probability 0.8, but with probability 0.1 the agent moves at right angles to the intended direction and with probability 0.0 the agent moves at opposite direction. A collision with a wall (the shaded square and/or the limits of the environment) results in no movement [16, Ch. 17, p. 645-646].

In order to illustrate the disadvantage of Bellman's operator method, we calculate its computational cost by Equation 2 based on Fig. 2 example.

$$\prod_{i=1}^{A} a_i \times \prod_{j=1}^{A} a_j \times \prod_{k=1}^{S} s_k \times \prod_{l=1}^{S} s_l \implies O(|A|^2 \times |S|^2). \tag{2}$$

In the example proposed, $A = 4$ and $S = 12$ resulting in a computational cost of 2,304 multiplications, because all actions multiplied by all states provides the stochastic outcome.

4 Proposed Method Formalization

We now introduce a Markov Decision Process (MDP) compact representation using Tensor Algebra, since none of the related work leverage advances in tensor-based computation to further increase solver efficiency. Our method is to represent an MDP using tensors to decompose the transition probability model into smaller components, with the goal to solve MDPs leveraging advances in Tensor Processing. Our formalization represents an approximate result of Bellman's operator method by sums of tensors component. This proposal allows the implementation to improve runtime of the solution, solving smaller problems faster than the original.

4.1 Definition of MDPs

We propose a Markov Decision Process formalization only for two-dimensional grids using concepts of Tensor Algebra, as follows:

- $\mathcal{X}$ set of states x, where $\mathcal{X} = \{x_1, x_2, x_3, ..., x_X\}$ and $X = |\mathcal{X}|$ is the number[3] of rows x;
- $\mathcal{Y}$ set of states y, where $\mathcal{Y} = \{y_1, y_2, y_3, ..., y_Y\}$ and $Y = |\mathcal{Y}|$ is the number of columns y;
- $\mathcal{S}$ set of all states xy, where $x \in \mathcal{X}$, $y \in \mathcal{Y}$, $\mathcal{S} = \{s_1, s_2, s_3, ..., s_S\}$ and $S = |\mathcal{S}|$ is the number of states xy;
- $\mathcal{E}$ environment matrix of order X rows by Y columns.
- $\mathcal{M}$ transition model matrix of order X×Y rows by X×Y columns.

Definition 1. Set of states $\mathcal{X}$ and $\mathcal{Y}$ contain $|\mathcal{X}|$ states x and $|\mathcal{Y}|$ states y, respectively, where[4] $x \in$ [1..X] and $y \in$ [1..Y].

Definition 2. The position (xy) is obtained through the function $\mathcal{E}(xy)$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Four subsets are also applied to formalize an MDP:

- $\mathcal{A}$ set of actions, where $\mathcal{A} = \{a_1, a_2, a_3, ..., a_A\}$ and $A = |\mathcal{A}|$ is the number of actions;
- $\mathcal{O}$ set of obstacles, where $\mathcal{O} = \{(o_{xy1}), (o_{xy2}), ..., (o_{xyO})\}$ and $O = |\mathcal{O}|$ is the number of obstacles;
- $\mathcal{T}$ set of terminals, where $\mathcal{T} = \{(t_{xy1}), (t_{xy2}), ..., (t_{xyT})\}$ and $T = |\mathcal{T}|$ is the number of terminals;
- $\mathcal{R}$ set of rewards, where $\mathcal{R} = \{[(xy_1), r_1], [(xy_2), r_2], ..., [(xy_{XY}), r_R]\}$ and $R = |\mathcal{R}|$ is the number of rewards.

Definition 3. A tuple of rewards $\mathcal{R}$ [(xy), r] is composed by: i) (xy), position xy in the environment where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$; and ii) r, respective reward of the state.

By using a formalization with tensors, we exploit the representation in a way MDP becomes more compact. Therefore, we solve an MDP by creating functions that precompute wherever possible the actions, the obstacles and the terminals of a problem, minimizing computational cost.

[3] The notation adopted is $|\mathcal{X}|$ to define the cardinality of a set $\mathcal{X}$.

[4] The notation adopted is $[i..j]$ referring to a number in range from i to j, inclusive, belonging to set of natural numbers; and notation $[i, j]$ refers to a number within range i to j, including number of real numbers.

4.2 State Transition Probability Matrix

In order to represent the stochastic outcome of actions, we formalize the *state transition probability matrix*, as:

$\mathcal{P}$ set of state transition probabilities, where $\mathcal{P}$ = { $[p_{11'}(a_1{,}a'_1)]$, $[p_{12'}(a_1, a'_2)]$, ..., $[p_{1A'}(a_1, a'_A)]$, $[p_{21'}(a_2, a'_1)]$, $[p_{22'}(a_2, a'_2)]$, ..., $[p_{2A'}(a_2, a'_A)]$, ..., $[p_{A1'}(a_A, a'_1)]$, $[p_{A2'}(a_A, a'_2)]$, ..., $[p_{AA'}(a_A, a'_A)]$} and $P = |\mathcal{P}|$ is the number of state transition probabilities.

Definition 4. Matrix $\mathcal{P}$ is defined by $A \times A$, covering transition probabilities (p) between a current state (xy) and a target state ($x'y'$) of the environment with a desired action (a) and the probability of that action happening (a').

4.3 Limits

In order to define environment limits, we introduce two fundamental sets:

$\mathcal{D}$ set of dimensions of the environment matrix, where $\mathcal{D} = \{d_1, d_2, d_3, ..., d_D\}$ and $D = |\mathcal{D}|$ is the number of dimensions;
$\mathcal{L}$ set of limits of the environment matrix, where $\mathcal{L} = \{[\alpha(d1), \Omega(d1)], [\alpha(d2), \Omega(d2)], ..., [\alpha(d_D), \Omega(d_D)]\}$ and $L = |2 \times \mathcal{D}|$ is the number of limits.

Definition 5. Set of limits $\mathcal{L}$ is defined by the number of dimensions $|\mathcal{D}|$ multiplied by 2. Thus, we will consider a initial limit, defined by α, and a final limit, defined by Ω, for each dimension. We define *limit states* as collision with walls, specifically environment's limits.

4.4 Successor States

In order to pre-compute a new-reachable state ($x'y'$) of a current state (xy), we define a concept of *successor states* by:

$succ_a(xy)$ is a valid-reachable successor state $x'y'$, from xy and an action a.

Definition 6. Successor states for a specific set of actions $\mathcal{A} = \{North, East, West, South\}$ and for a two-dimensional environment matrix $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, are defined by:

```
7.1. if a = North, then
         if      $x \neq \alpha(\mathcal{X})$, then x'=x-1 and y'=y;
         else    $x = \alpha(\mathcal{X})$, then x'=x and y'=y

7.2. if a = East, then
         if      $y \neq \Omega(\mathcal{Y})$, then x'=x and y'=y+1;
```

```
        else    $y = \Omega(\mathcal{Y})$, then x'=x and y'=y

7.3.  if a = West, then
        if      $y \neq \alpha(\mathcal{Y})$, then x'=x and y'=y-1;
        else    $y = \alpha(\mathcal{Y})$, then x'=x and y'=y

7.4.  if a = South, then
        if      $x \neq \Omega(\mathcal{X})$, then x'=x+1 and y'=y;
        else    $x = \Omega(\mathcal{X})$, then x'=x and y'=y
```

In Definition 6, we only deal with collision with environment's limits, thus a valid successor state can belong to set of obstacles ($\mathcal{O}$).

5 Bellman's Operator Method Optimizations

In this section we propose four optimizations of Bellman's operator method in order to reduce computational cost. The first optimization eliminates zero-probabilities of state transition $\mathcal{P}$:

$\mathcal{Z}^{(a)}$ set of actions with non-zero state transition probabilities of action a, where $\mathcal{Z}^{(a)} = \{[p_{11'}(a_1,a'_1)], [p_{12'}(a_1,a'_2)], [p_{13'}(a_1,a'_3)], ..., [p_{1Z'}(a_1,a'_Z)]\}$ and $Z^{(a)} = |\mathcal{Z}^{(a)}|$ is the number of non-zero state transition probabilities of action a. Therefore, $\mathcal{Z}^{(a)} \subset \mathcal{P}$.

The second optimization is focused on eliminating terminal states. The third optimization eliminates obstacle states as we address in set $\mathcal{V}$.

$\mathcal{V}$ set of valid states, where $\mathcal{V} = \{v_1, v_2, v_3, ..., v_V\}$ and $V = |\mathcal{V}|$ is the number valid states. Valid states are determined by excluding obstacles and terminal states of $\mathcal{S}$. Therefore, $\mathcal{V} \subset \mathcal{S}$.

Finally, the fourth optimization is focused on representing transition models of MDPs with tensor decomposition as third-order tensors.

$\mathcal{C}^{(a)}$ set of tensor components of action a. The set $\mathcal{C}^{(a)}$ is a third-order tensor, where $\mathcal{C}^{(a)} = \{[xy_1, x'y'_1, \mathcal{P}(a,a')_1], [xy_2, x'y'_2, \mathcal{P}(a,a')_2], ..., [xy_C, x'y'_C, \mathcal{P}(a,a')_C]\}$ and $C^{(a)} = |\mathcal{V} \times \mathcal{Z}^{(a)}|$ is the number of tensor components of action a.

Definition 7. Each action constitutes a third-order tensor (i.e., $|\mathcal{D}| = 3$). Each tensor component is composed by: i) xy, which is the current state; ii) $x'y'$, is the intended state; and iii) $\mathcal{P}(a,a')$, is the probability of state transition.

In Fig. 3 we illustrate the Definition 7 of tensor components.

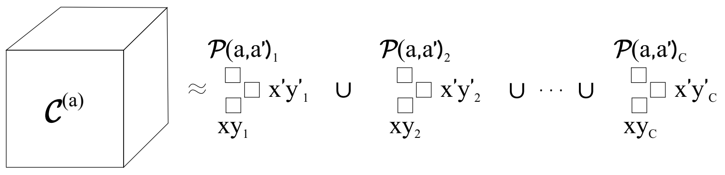

Fig. 3. Definition of a 3^{th}-order tensor of action a.

6 Algorithm

In this section we propose a tensor-based algorithm to generate compact transition models, as we modeled in our formalization. Algorithm 1 consists in composing the set of components $\mathcal{C}^{(a)}$ at each iteration with a set of values [xy, x'y', $\mathcal{P}(a,a')$], where xy is the current state, $x'y'$ is the target state and $\mathcal{P}(a,a')$ is the probability of state transition.

Algorithm 1: Tensor-based algorithm to generate transition models of MDPs.

```
 1: for ∀a ∈ $\mathcal{A}$ do
 2:    for ∀x ∈ $\mathcal{X}$ do
 3:       for ∀y ∈ $\mathcal{Y}$, where xy ∉ $\mathcal{T}$ and xy ∉ $\mathcal{O}$ do
 4:          for ∀a' ∈ $\mathcal{A}$ | $\mathcal{P}$(a, a') ≠ 0 do
 5:             x'y' = succ$_{a'}$(xy), x' ∈ $\mathcal{X}$, y' ∈ $\mathcal{Y}$
 6:             if x'y' ∉ $\mathcal{O}$ then
 7:                $\mathcal{C}^{(a)} \leftarrow \mathcal{C}^{(a)} \cup$ [xy, x'y', $\mathcal{P}(a,a')$]
 8:             else
 9:                $\mathcal{C}^{(a)} \leftarrow \mathcal{C}^{(a)} \cup$ [xy, xy, $\mathcal{P}(a,a')$]
10:             end if
11:          end for
12:       end for
13:    end for
14: end for
```

As previously mentioned, $succ_a(xy)$ can return successor states $(x'y')$ belonging to set of obstacles $(\mathcal{O})$. In order to address this issue, in lines 8-9 we define target states as current ones, i.e., no movement is performed.

7 Proposed Method Computational Cost Analysis

In this section we use Fig. 2 (see Section 3) as example to explain the formalization and optimizations of our proposed method. As we defined in Section 4.1, the first properties are:

$\mathcal{X} = \{1,2,3\}$ and $|\mathcal{X}| = 3$ states x;
$\mathcal{Y} = \{1,2,3,4\}$ and $|\mathcal{Y}| = 4$ states y;
$\mathcal{S} = \{11,12,13,14,21,22,23,24,31,32,33,34\}$ and $|\mathcal{S}| = 12$ states xy;
$\mathcal{A} = \{N, E, W, S\}$ and $|\mathcal{A}| = 4$ actions;
$\mathcal{O} = \{(22)\}$ and $|\mathcal{O}| = 1$ obstacle state;
$\mathcal{T} = \{(14),(24)\}$ and $|\mathcal{T}| = 2$ terminal states;
$\mathcal{R} = \{[(11),-3], [(12),-3], [(13),-3], [(14),100], [(21),-3], [(22),-3], [(23),-3], [(24),-100], [(31),-3], [(32),-3], [(33),-3], [(34),-3]\}$ and $|\mathcal{R}| = 12$ rewards.

In Section 4.2 we define the state transition probability matrix, and in Section 4.3 we define the dimensions and the limits of an environment. Based on Fig. 2, we formalize these properties as:

$\mathcal{P} = \{[[0.8(N,N')], [0.1(N,E')], [0.1(N,W')], [0.0(N,S')]], [[0.1(E,N')], [0.8(E,E')], [0.0(E,W')], [0.1(E,S')]], [[0.1(W,N')], [0.0(W,E')], [0.8(W,W')], [0.1(W,S')]], [[0.0(S,N')], [0.1(S,E')], [0.1(S,W')], [0.8(S,S')]]\}$ and $|\mathcal{P}| = 4{\times}4 = 16$ state transition probabilities;
$\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ and $|\mathcal{D}| = 2$ dimensions;
$\mathcal{L} = \{[\alpha(\mathcal{X}), \Omega(\mathcal{X})], [\alpha(\mathcal{Y}), \Omega(\mathcal{Y})]\}$, where $\alpha(\mathcal{X}) = 1$, $\Omega(\mathcal{X}) = 3$, $\alpha(\mathcal{Y}) = 1$, $\Omega(\mathcal{Y}) = 4$ and $|\mathcal{L}| = 4$ limits.

Finally, we define the optimizations of Bellman's operator method (see Section 5), as follows:

$\mathcal{Z}^{(N)} = \{[0.8(N,N')], [0.1(N,E')], [0.1(N,W')]\}$ and $|\mathcal{Z}^{(N)}| = 3$ non-zero state transition probabilities for action North (N);
$\mathcal{Z}^{(E)} = \{[0.1(E,N')], [0.8(E,E')], [0.1(E,S')]\}$ and $|\mathcal{Z}^{(E)}| = 3$ non-zero state transition probabilities for action East (E);
$\mathcal{Z}^{(W)} = \{[0.1(W,N')], [0.8(W,W')], [0.1(W,S')]\}$ and $|\mathcal{Z}^{(W)}| = 3$ non-zero state transition probabilities for action West (W);
$\mathcal{Z}^{(S)} = \{[0.1(S,E')], [0.1(S,W')], [0.8(S,S')]\}$ and $|\mathcal{Z}^{(S)}| = 3$ non-zero state transition probabilities for action South (S);
$\mathcal{V} = \{11,12,13,21,23,31,32,33,34\}$ and $|\mathcal{V}| = 9$ valid states.
$\mathcal{C}^{(N)} = \{[xy_1, x'y'_1, \mathcal{P}(N,N')_1], [xy_2, x'y'_2, \mathcal{P}(N,E')_2], ..., [xy_{27}, x'y'_{27}, \mathcal{P}(N,W')_{27}]\}$ and $|\mathcal{V} \times \mathcal{Z}^{(N)}| = 27$ tensor components;
$\mathcal{C}^{(E)} = \{[xy_1, x'y'_1, \mathcal{P}(E,N')_1], [xy_2, x'y'_2, \mathcal{P}(E,E')_2], ..., [xy_{27}, x'y'_{27}, \mathcal{P}(E,S')_{27}]\}$ and $|\mathcal{V} \times \mathcal{Z}^{(E)}| = 27$ tensor components;
$\mathcal{C}^{(W)} = \{[xy_1, x'y'_1, \mathcal{P}(W,N')_1], [xy_2, x'y'_2, \mathcal{P}(W,W')_2], ..., [xy_{27}, x'y'_{27}, \mathcal{P}(W,S')_{27}]\}$ and $|\mathcal{V} \times \mathcal{Z}^{(W)}| = 27$ tensor components;
$\mathcal{C}^{(S)} = \{[xy_1, x'y'_1, \mathcal{P}(S,E')_1], [xy_2, x'y'_2, \mathcal{P}(S,W')_2], ..., [xy_{27}, x'y'_{27}, \mathcal{P}(S,S')_{27}]\}$ and $|\mathcal{V} \times \mathcal{Z}^{(S)}| = 27$ tensor components.

In order to demonstrate the idea of tensor components generated by Algorithm 1, we show the components of tensor $\mathcal{C}^{(N)}$, where we express the dimensions (i, j, k) of $\mathcal{C}^{(N)}$ by $c^N_{ijk} = xy_c, x'y'_c, \mathcal{P}(N,a')_c$, as follows:

$[11_1, 1'1'_1, \mathcal{P}(N,N')_1]$	$[21_{10}, 1'1'_{10}, \mathcal{P}(N,N')_{10}]$	$[32_{19}, 3'2'_{19}, \mathcal{P}(N,N')_{19}]$
$[11_2, 1'2'_2, \mathcal{P}(N,E')_2]$	$[21_{11}, 2'1'_{11}, \mathcal{P}(N,E')_{11}]$	$[32_{20}, 3'3'_{20}, \mathcal{P}(N,E')_{20}]$
$[11_3, 1'1'_3, \mathcal{P}(N,W')_3]$	$[21_{12}, 2'1'_{12}, \mathcal{P}(N,W')_{12}]$	$[32_{21}, 3'1'_{21}, \mathcal{P}(N,W')_{21}]$
$[12_4, 1'2'_4, \mathcal{P}(N,N')_4]$	$[23_{13}, 1'3'_{13}, \mathcal{P}(N,N')_{13}]$	$[33_{22}, 2'3'_{22}, \mathcal{P}(N,N')_{22}]$
$[12_5, 1'3'_5, \mathcal{P}(N,E')_5]$	$[23_{14}, 2'4'_{14}, \mathcal{P}(N,E')_{14}]$	$[33_{23}, 3'4'_{23}, \mathcal{P}(N,E')_{23}]$
$[12_6, 1'1'_6, \mathcal{P}(N,W')_6]$	$[23_{15}, 2'3'_{15}, \mathcal{P}(N,W')_{15}]$	$[33_{24}, 3'2'_{24}, \mathcal{P}(N,W')_{24}]$
$[13_7, 1'3'_7, \mathcal{P}(N,N')_7]$	$[31_{16}, 2'1'_{16}, \mathcal{P}(N,N')_{16}]$	$[34_{25}, 2'4'_{25}, \mathcal{P}(N,N')_{25}]$
$[13_8, 1'4'_8, \mathcal{P}(N,E')_8]$	$[31_{17}, 3'2'_{17}, \mathcal{P}(N,E')_{17}]$	$[34_{26}, 3'4'_{26}, \mathcal{P}(N,E')_{26}]$
$[13_9, 1'2'_9, \mathcal{P}(N,W')_9]$	$[31_{18}, 3'1'_{18}, \mathcal{P}(N,W')_{18}]$	$[34_{27}, 3'3'_{27}, \mathcal{P}(N,W')_{27}]$

As the third-order tensor representation $\mathcal{C}^{(a)} \in \mathbb{C}^{I*J*K}$, we express the sum of individual components by Equation 3.

$$\mathcal{C}^{(a)} \approx \sum_{c=1}^{C} (xy_c \otimes x'y'_c \otimes \mathcal{P}(a,a')_c) \tag{3}$$

Where C is a positive integer and $xy_c \in \mathbb{C}^I$, $x'y'_c \in \mathbb{C}^J$, and $\mathcal{P}(a,a')_c \in \mathbb{C}^K$ for c = 1, ..., C. Thus, we write Equation 3 as follows:

$$C_{ijk} \approx \sum_{c=1}^{C} (xy_{ic}, x'y'_{jc}, \mathcal{P}(a,a')_{kc}) \text{ for } i = 1, ..., I;\ j = 1, ..., J;\ k = 1, ..., K.$$

Therefore, we define the computational cost of our optimizations previously mentioned by Equation 4 and its respective complexity (O).

$$\prod_{i=1}^{A} a_i \times \prod_{j=1}^{C^{(a_i)}} c_j \times \prod_{k=1}^{D} d_k \implies O(|C^{(a)}| \times |A| \times |D|) \tag{4}$$

In example proposed, $C^{(a_i)} = 27$, $A = 4$ and $D = 3$, resulting in a computational cost of 324 multiplications.

8 Related Work

A method to represent large MDPs in a compact representation is called factored MDPs. The idea of representing a large MDP using a factored model was first proposed by Boutilier et al. [5]. The benefit of using such a representation is the state transition model can be compactly represented using one of several methods, the most common being a Dynamic Bayesian Network (DBN). This technique allows a compact representation of the transition model, by exploiting the fact the transition of a variable often depends only on a small number of other variables [11].

Guestrin et al. [9–11] developed three approaches to solve factored MDPs using DBNs, respectively. The first approach introduces an efficient planning algorithm for cooperative multiagent dynamic systems. The second approach proposes an approximation scheme to the solution using an approximate value function with a compact representation. The third approach presents two approximate solution algorithms using factored MDPs, where the first algorithm uses approximate linear programming, and the second one uses approximate dynamic programming. All three approaches prove to be able to deal with problem of exponentially large representations of vectors.

Delgado et al. [7] use Markov Decision Process with Imprecise Transition Probabilities (MDP-IP), due to the necessity when transition probabilities are imprecisely specified [8]. Thus, the authors introduce the factored MDP-IP, by proposing to replace the usual Dynamic Bayes Networks (DBNs) used in factored MDPs by Dynamic Credal Networks (DCNs). Results show up to two orders of magnitude speedup in comparison to traditional dynamic programming approaches.

In general, all proposed methods and algorithms in related work prove decomposing the representation into smaller subsets (a factored representation) allows an exponential reduction in representation size of structured MDPs. Moreover, no single method proving superior in all applications is provided, so it still remains an active area of research. In our approach, instead of using DBNs and DCNs to represent transition models of factored MDPs, we provide a representation of MDPs using tensor decomposition, concept of Tensor Algebra, which represents a multidimensional tensor by sums of separable arrays.

9 Conclusion and Future Work

Since investigating methods for efficiently determining optimal or near-optimal policies remains an active area of research, in this paper we formalize an MDP using Tensor Algebra to represent transition models compactly, aiming to allow developers to implement efficient MDPs solvers. In Table 1 we show a comparison of computational cost required to create the transition model of the 4×3 grid example (Fig. 2) of our approach and Bellman's operator method.

Table 1. Computational cost comparison

Method	Number of multiplications
Bellman's operator method	2,304
Proposed method	324
Complexity reduction	85.94 %

We compose out method by sums of tensors components, which represents an approximate result of Bellman's operator method. This optimized proposal enables a complexity reduction of 85.94% compared to the regular method.

Therefore, as future work, we plan to: i) generalize our formalization for N-dimensional environments; ii) compile Relational Dynamic Influence Diagram

Language (RDDL)[5] representations into tensor algebra; iii) implement the solution using tensors as we modeled; and iv) solve MDPs leveraging advances in Tensor Processing Frameworks that run on Graphics Processing Units (GPUs) to further increase solver efficiency, by processing each action apart and at the same time.

In closing, we envision this kind of formalization will allow developers to implement efficient MDP solvers. Moreover, in the lack of solvers leveraging advances in tensor processing, the tensor-based representation is a potential stand-in solution, as computational cost calculations show it provides complexity reduction compared to the regular method.

References

1. Amari, S.V., McLaughlin, L., Hoang Pham: Cost-effective condition-based maintenance using markov decision processes. In: RAMS '06. Annual Reliability and Maintainability Symposium, 2006. pp. 464–469 (2006)
2. Bellman, R.: A markovian decision process. Journal of Mathematics and Mechanics pp. 679–684 (1957)
3. Bellman, R.E.: Dynamic Programming. Dover Publications, Inc., New York, New York, USA (1957)
4. Boger, J., Hoey, J., Poupart, P., Boutilier, C., Fernie, G., Mihailidis, A.: A planning system based on markov decision processes to guide people with dementia through activities of daily living. IEEE Transactions on Information Technology in Biomedicine **10**(2), 323–333 (2006)
5. Boutilier, C., Dearden, R., Goldszmidt, M.: Exploiting structure in policy construction. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2. pp. 1104–1111. IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, California, USA (1995), http://dl.acm.org/citation.cfm?id=1643031.1643043
6. Cassandra, A.R., Kaelbling, L.P., Kurien, J.A.: Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96. vol. 2, pp. 963–972 vol.2 (1996)
7. Delgado, K.V., Sanner, S., de Barros, L.N.: Efficient solutions to factored mdps with imprecise transition probabilities. Artificial Intelligence **175**(9), 1498 – 1527 (2011). https://doi.org/https://doi.org/10.1016/j.artint.2011.01.001
8. Delgado, K.V., Sanner, S., de Barros, L.N., Cozman, F.G.: Efficient solutions to factored mdps with imprecise transition probabilities. In: Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling. pp. 98–105. ICAPS'09, AAAI Press (2009), http://dl.acm.org/citation.cfm?id=3037223.3037237
9. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored mdps. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. pp. 1523–1530. NIPS'01, MIT Press, Cambridge, Massachusetts, USA (2001), http://dl.acm.org/citation.cfm?id=2980539.2980737

[5] https://github.com/ssanner/rddlsim

10. Guestrin, C., Koller, D., Parr, R.: Solving factored pomdps with linear value functions. In: Planning under Uncertainty and Incomplete Information. pp. 67–75. IJCAI'01, Seattle, Washington, USA (2001)
11. Guestrin, C., Koller, D., Parr, R., Venkataraman, S.: Efficient solution algorithms for factored mdps. J. Artif. Int. Res. **19**(1), 399–468 (2003), http://dl.acm.org/citation.cfm?id=1622434.1622447
12. Guestrin, C.E.: Planning Under Uncertainty in Complex Structured Environments. Ph.D. thesis, Stanford University, Stanford, California, USA (2003), aAI3104233
13. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X, http://dx.doi.org/10.1137/07070111X
14. Kronecker, L.: Über einige interpolationsformeln für ganze funktionen mehrerer variabeln. Lecture at the Academy of Sciences pp. 133–141 (1865)
15. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, New York, USA, 1st edn. (1994)
16. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Press, Upper Saddle River, New Jersey, USA, 3nd edn. (2009)
17. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press, Cambridge, Massachusetts, USA (2018)