Hindawi
Journal of Mathematics
Volume 2022, Article ID 8404716, 8 pages
https://doi.org/10.1155/2022/8404716

Research Article

Decomposition Methods for Solving Finite-Horizon Large MDPs

Bouchra el Akraoui ,[1] Cherki Daoui,[1] Abdelhadi Larach,[1] and khalid Rahhali[2]

[1]*Laboratory of Information Processing and Decision Support, Sultan Moulay Slimane University, B.P. 523, Beni-Mellal, Morocco*
[2]*Loboratory of Mathematics, Statistics and Applications, Mohammed V University, B.P. 1014 RP, Rabat, Morocco*

Correspondence should be addressed to Bouchra el Akraoui; b.elakraoui@usms.ac.ma

Received 11 March 2022; Accepted 8 July 2022; Published 13 August 2022

Academic Editor: Efthymios G. Tsionas

Conventional algorithms for solving Markov decision processes (MDPs) become intractable for a large finite state and action spaces. Several studies have been devoted to this issue, but most of them only treat infinite-horizon MDPs. This paper is one of the first works to deal with non-stationary finite-horizon MDPs by proposing a new decomposition approach, which consists in partitioning the problem into smaller restricted finite-horizon MDPs, each restricted MDP is solved independently, in a specific order, using the proposed hierarchical backward induction (HBI) algorithm based on the backward induction (BI) algorithm. Next, the sub-local solutions are combined to obtain a global solution. An example of racetrack problems shows the performance of the proposal decomposition technique.

1. Introduction

Stochastic models have recently gained a lot of attention in the artificial intelligent (*AI*) communities; it offers a suitable framework for solving problems with uncertainties. *MDPs* [1] are one such model that achieved promising results in numerous applications[2–4]. Most real-world problems have very large state spaces that require many mathematical operations and substantial memory. It is intractable to solve them with classical *MDPs* algorithms [5]. Motivated by these considerations, several recent types of research have used the decomposition technique to overcome the computational complexity. The decomposition approach introduced by Bather [6] divides the state space into strongly connected components (*SCCs*) according to certain levels and solves the small problems called restricted *MDPs* separately at each level to obtain the global solution of the original *MDP* by combining the partial solutions. Subsequently, Ross and Varadarajan [7] also proposed a similar decomposition technique for solving constrained limiting average *MDPs*. It is employed in various categories of *MDPs* (discounted, average, and weighted *MDPs*) in diverse studies [8, 9]. The weakness point of these approaches is their polynomial execution complexity. To accelerate the execution time Chafik and Daoui integrate the decomposition and parallelism schemes [10], unfortunately, the decomposition algorithm remains polynomial at runtime. Following, to accelerate the convergence time of the decomposition, Larach, and Daoui [11] investigated the state space decomposition approach into *SCCs* according to some levels based on Tarjan's algorithm [12]. Subsequent work [13–15] developed approaches to solve *MDPs* with factorization methods that were introduced by [16]. The goal of factoring a problem is to decompose it into smaller items. Factored *MDPs* produce compact representations of complex and uncertain systems allowing for an exponential reduction in the complexity of the representation [17]. These factorized approaches represent states as factorized states with an internal structure and state transition matrices as dynamic Bayesian networks (*DBNs*). However, methods for solving representations based on factorized *DBNs* do not exploit advances in tensor decomposition methods for representing large atomic *MDPs*. More recently, research such as [17, 18] exploits a similar idea to the thesis of Smart and [19] aims at improving the efficiency of *MDP* solvers by using tensor decomposition methods to compact state transition matrices. The solver uses the value iteration and policy iteration algorithms to compute the solution compactly. The authors try different ways to parallelize their proposed approaches, but no improvement in the execution time is observed. These methods are based on multiplications between small tensor components. More recently, work has been carried out to address this problem in the context of parallelism [20] presenting a way to decompose an *MDP* into *SCCs* and find dependency chains for these *SCCs*. They solve independent chains of *SCCs* with a proposed variant of the topological value iteration (*TVI*) algorithm, called parallel chained *TVI* aimed at improving the execution time on *GPUs*. In this context, research groups [21–23] have improved iterative algorithms in parallel versions to accelerate their convergence.

The literature mentioned above focuses only on solving different types of *MDPs* under the infinite-horizon criterion. It is difficult to find papers that focus on solving large non-stationary finite-horizon *MDPs*. This paper is oriented in this underexplored direction, the main objective of this work is to propose a new decomposition technique tackling the challenges of reducing memory requirements and computational cost.

The proposed technique consists in partitioning the global problem into smaller restricted finite-horizon *MDPs*, each restricted *MDP* is solved independently, in a specific order, using the backward induction algorithm. Next, the sub-local solutions are combined to obtain a global solution.

There are several problems modeled as *MDPs* with an initial given state i_0, then the optimal action $f(i_0)$, and the optimal value $V^T(i_0)$ are computed by solving just the restricted *MDP* corresponding to the classes accessible from the class containing i_0 (one does not need to consider all states). This is also an advantage of this method, which reduces memory consumption and speeds up the computation time.

The remainder of the article is organized as follows: the second section introduces the fundamentals of finite-horizon *MDPs*. The third section focuses on the decomposition technique and describes the new finite-horizon restricted *MDP*. The fourth section presents the proposed hierarchical backward induction algorithm. The last section illustrates the advantages of this decomposition technique by its application to a racetrack problem. The paper concludes with conclusions and prospects for future work.

2. Markov Decision Process

Markov decision processes have been widely studied as an elegant mathematical formalism for many decision-making problems in a variety of fields of science and engineering [24]. The objective is to approximate the best decision policies (action selection) to achieve maximum expected rewards (minimum costs) in a given stochastic dynamic environment satisfying the Markov property [1]. In this section, we will present non-stationary finite-horizon *MDP* with a finite state and action spaces.

Formally, a non-stationary *MDP* with a finite-horizon is defined by five-tuple (S, A, T, P, and R), where S and A are the state and action spaces; T is the time horizon; P denotes the state transition probability function, where $P(S_{t+1} = j \mid S_t = i, A_t = a) = p^t_{iaj}$ is the probability of transition from state i to state *j* by taking action a at time *t*; S_t (A_t) is a random variable indicating the state (action) at time *t*; *R* supplies the reward function defined on state transitions, where r^t_{ia} indicates the reward gained if the action a is executed in the state at t period. Most solvers of *MDPs* attempt to find an optimal policy that specifies (optimal) action should be taken for each agent at each state. If the process will be considered a finite planning horizon *T*, an optimal policy π^* is given as the policy that maximizes the expected reward. π^* maximizes the value function of the Bellman equation [24]: $v_i^T(\pi^*) = \sup v_i^T(\pi), i \in E$, where $v_i^T(\pi)$ is the total expected reward in *T* periods, given that the process starts from initial state *i* and the policy π is used.

$$v_i^T(\pi) = \sum_{t=1}^{T} \sum_{j\in E} \sum_{a\in A(j)} P_\pi(X_t = j, Y_t = a \mid X_1 = i) r^t_{ja}, \tag{1}$$

where X_t and Y_t are the random variables representing, respectively, the state and action at time *t*. Besides, we define the optimal value vector V^T:

$$V_i^T = \sup_\pi v_i^T(\pi), i \in E. \tag{2}$$

It is well known that the backward induction algorithm is one of the most common iterative methods used to find an optimal policy. In the next section, we will discuss it in more detail.

3. Backward Induction Algorithm

In this section, we compute an optimal policy as well as the optimal value vector using the backward induction algorithm, its iterative process starting at the end of the planning horizon *T*, one computes the values for the previous periods. Then, after *T* iterations an optimal policy is found.

The following theorem introduced in [25] demonstrates the validity of the *BI* algorithm:

Theorem 1. *Let $x_i^{T+1} = 0$, $i \in S$. Define recursively for $t = T, T-1, \ldots, 1$, a deterministic decision rule f^t and the vector x^t as follows:*

(i) $\{r(f^t)\}_i + \{P(f^t)x^{t+1}\}_i = \max_{a\in A(i)} \left\{ r^t_{ia} + \sum_j p^t_{iaj} x_j^{t+1} \right\}, i \in S,$

(ii) $x^t = r(f^t) + P(f^t)x^{t+1},$

then $R^* = (f^1, f^2, \ldots, f^T)$ is an optimal policy and x^1 is the optimal value vector V^T.

Proof (see [25]).

To accelerate the execution time of the classical *BI*-algorithm, the authors used the proposal of [11], they introduced for each action *a* the list of state-action successors denoted by $\Gamma_a^+(i)$, where $\Gamma_a^+(i) = \{j \in E: P^t_{iaj} > 0, t = 1, \ldots, T\}$. This allowed to reduce the time complexity from $O(T|A|E^2)$ to $O(T|\Gamma_a^+|E^2)$ arithmetic operations, where $|\Gamma_a^+|$ denotes the average number of state-action successors; *T* designs the horizon and *E* is the number of

```
(1) $ABI$ (In $MDP$: $E, P, A, R, T$;  Out $(V^T, R^*)$
(2) $t \longleftarrow T + 1$;
(3) $\forall j \in E$ Take $v_j^{T+1} \longleftarrow 0$;
(4) Repeat
(5) $t \longleftarrow t - 1$:
(6) For each $i \in E$ Do
(7) $\{r(f^t)\}_i + \{P(f^t)v^{t+1}\}_i = \max_{a \in A(i)} \left\{ r_{ia}^t + \sum_{j \in \Gamma_a^+(i)} p_{iaj}^t v_j^{t+1} \right\}$ //The Deterministic Decision Rule $f^t$
(8) $v^t = r(f^t) + P(f^t)v^{t+1}$
(9) For each $i \in E$ Do
(10) $R^* = (f^1, f^2, \ldots, f^T)$ is an optimal policy and $v^1$ is the optimal value vector.
11 Return $V^T, R^*$
```

Algorithm 1: Ameliorated backward induction.

```
(i) $(\Omega) \longleftarrow E$; $n \longleftarrow 0$; $L_n \longleftarrow \{ C_i$: $C_i$ is closed $\}$
(ii) If $L_0 = E$ Stop.
(iii) Otherwise, unless $\Omega \neq \emptyset$; do
(iv) Delete $L_n$ (i.e., $\Omega \longleftarrow \Omega / L_n$ and eliminate all arcs coming into $L_n$);
(v) $L_{n+1} \longleftarrow \{ C_i$: $C_i$ is closed in the restricted MDP to $\Omega\}$;
(vi) $n \longleftarrow n + 1$.
```

Algorithm 2: Finding levels.

states. Algorithm 1 describes the ameliorated backward induction algorithm (*ABI*) as follows: □

4. Hierarchical Backward Induction Algorithm

The *BI* algorithm becomes quite impractical to compute an optimal policy for finite-horizon *MDPs* with large state space. For non-stationary finite-horizon *MDPs,* the computing load can increase further. To overcome this issue, we describe, in this section, a new decomposition technique for improving the performance and reducing the time running.

4.1. The Decomposition Technique. Let us consider an oriented graph $G = (S, U)$, associated with the original *MDP*, where S is a set of nodes that represents a state space and $U = \{(i, j) \in S\text{: } \exists a \in A(i) P_{iaj} > 0\}$ is a set of directed arcs. There exists a unique partition $S = C_1 \cup C_2 \ldots \cup C_p$ of the state space S into strongly connected classes. Note that the *SCCs* are defined to be the classes with respect to the relation on G defined by i is strongly connected to j if and only if $i = j$ or there exists a directed path from i to j and a directed path from j to i. There are many good algorithms in graph theory for the computation of such partition, e.g., see [11].

Now, we construct by induction the levels of the graph G. The level L_0 is formed by all closed classes C_i, that is for all $i \in C_i; a \in A(i)$: $P_{iaj} = 0$ for all $j \notin C_i$. The level L_p is formed by all classes C_i such that the end of any arc emanating from C_i is in some levels $L_{p-1}, L_{p-2}, \ldots, L_0$. After finding the *SCCs* using Tarjan's algorithm, their belonging levels are found by using the following algorithm (algorithm 2) introduced in [26].

For each level L_n, $n = 0, 1, 2, \ldots, L$. Let (C_{lk}), $k \in \{1, 2, \ldots, K(l)\}$ be the strongly connected classes corresponding to the

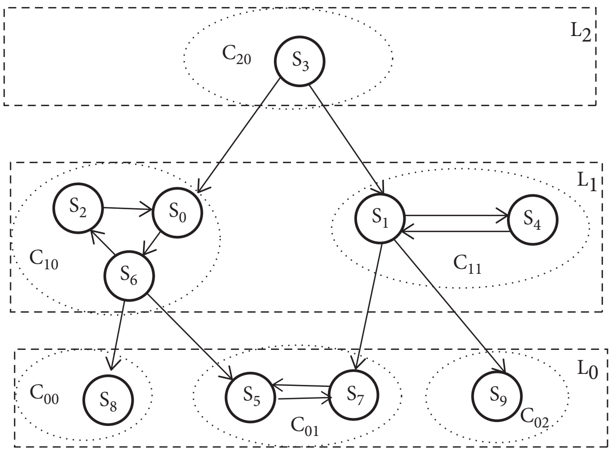

Figure 1: Example of SCCs and their levels.

nodes in level l (see Figure 1). Each class C_{lk} leads to a partial MDP_{lk} that is solved independently, the global solution is obtained by combining these partial solutions.

The hierarchical method used by several researchers for several categories of *MDPs,* addresses the "curse of dimensionality" of large *MDPs*, was described by [27] and later further developed by [11, 26]. It consists of breaking up the state space into small subsets, solving the restricted *MDPs* problems corresponding to these subsets, and combining these solutions to determine the solution of the global problem. Based on the above decomposition technique, the authors propose, a hierarchical backward induction (*HBI*) algorithm by decomposing the original finite-horizon *MDPs* into restricted *MDPs* corresponding to each *SCC*. These restricted *MDPs* are solved independently and according to their level.

The performance of the proposed algorithm is exposed after the search for the optimal policy of the known initial state, the algorithm solves only the restricted *MDP* corresponding to the reachable classes of the initial state. For example, in Figure 1, the initial state S_0 is in class C_{10}. Only the restricted *MDPs* corresponding to the *SCCs:* C_{10}, C_{00}, and C_{01} are solved.

In the next section, we will define new *MDPs* called the restricted finite-horizon *MDPs*.

4.2. The Restricted Finite-Horizon MDPs. In this work, we consider non-stationary finite-horizon *MDPs* with large finite state and action spaces. Now, we construct the restricted MDP_{nk} corresponding to each strongly connected class C_{nk}, $k \in \{1, 2, \ldots, K(n)\}$ in level L_n as follows:

(i) et $E_n = \bigcup\{i\varepsilon C_{pk}, \quad p = 0, 1, \ldots, \quad n-1 \text{ and } k\varepsilon \{1, 2, \ldots, k(p)\}\}$.

(ii) States space: $S_{nk} = C_{nk} \cup \{j\varepsilon E_n | \sum i\varepsilon C_{nk},\ a \in A(i);\ p^t_{iaj} > 0, \quad t = 1, \ldots, T\}$;

(iii) Actions space: For $i\varepsilon S_{nk}$, $A_{nk}(i) = \begin{cases} A(i), if\, i\varepsilon C_{nk} \\ \theta, \text{Otherwize} \end{cases}$;

(iv) Transition probabilities: For $t = 1, 2, \ldots, T$,

(v) for $i, j\varepsilon S_{nk}, p^t_{nk}(i, a, j) = \begin{cases} p^t_{iaj}\, if, i\varepsilon C_{nk},\ a \in A(i) \\ 1, if \quad i = j, i \notin C_{nk} \end{cases}$;

(vi) Reward function: For $t = 1, 2, \ldots, T$.

(vii) If $i \in C_{nk}, r^t_{pk}(i, a) = r^t_{ia}$;

(viii) If $i\varepsilon(S_{nk}/C_{nk})$, $\exists m\varepsilon\{0, 1, \ldots, n-1\}, \exists h\varepsilon \{1, 2, \ldots, k(m)\}$ {: $i\varepsilon C_{mh}, r^t_{nk}(i, \theta) = V^T_{mh}(i)/N$.

N is the horizon and V^T_{mh} is the optimal value vector of MDP_{mh} calculated in the previous levels.

According to the definition of the restricted finite-horizon *MDPs*, we remark that.

The restricted finite-horizon *MDPs* are solved according to the ascending order of levels and in the same level L_p. The restricted finite-horizon *MDPs* are independent, so they can be solved in parallel.

4.3. Hierarchical Backward Induction Algorithm. Based on the above restricted finite-horizon *MDPs*, the authors present in this section, a new algorithm called hierarchical backward induction (*HBI*) algorithm (algorithm 3). The main contribution is to show that the optimal value in the restricted MDP_{pk} is equal to the optimal value in the original *MDP* (Theorem 2).

Now, the corresponding restricted finite-horizon *MDPs* are constructed and immediately solved by using this procedure:

The following theorem shows the validity of the *HBI* algorithm.

Theorem 2. *Let $R^* = (f^1, f^2, \ldots, f^T)$ and V^T are, respectively, an optimal policy and the optimal value vector in the original MDP. If $R_{pk} = (f^1_{pk}, f^2_{pk}, \ldots, f^T_{pk})$ and V^T_{pk} are, respectively, an optimal policy and the optimal value vector in the restricted MDP_{pk}, then for all $i \in C_{pk}$, for $t = 1, 2, \ldots, T$, $f^t_{pk}(i) = f^t(i)$ is an optimal action in the original MDP and $V^T_{pk}(i) = V^T(i)$.*

Proof. The proof is by induction. For $p = 0$ (level L_0); $k \in \{1, \ldots, K(0)\}$. Let $R_{0k} = (f^1_{0k}, f^2_{0k}, \ldots, f^T_{0k})$ and V^T_{0k} are, respectively, an optimal policy and the optimal value vector in the restricted MDP_{0k}.

According to Theorem 1, we have for $t = T, T-1, \ldots, 1$:

$$\begin{cases} \forall i \in S_{0k}, f^t_{0k}(i) = \underset{a \in A_{0k}(i)}{\arg\max} \left\{ r^t_{0k}(i, a) + \sum_{j \in S_{0k}} p^t_{0k}(i, a, j) x^{t+1}_j \right\}, \\ x^t = r(f^t_{0k}) + P(f^t_{0k}) x^{t+1}, \end{cases} \tag{3}$$

from the definition of the restricted MDP_{0k}, the state space $S_{0k} = C_{0k}$, the action space $A_{0k}(i) = A(i)$ for $i \in S_{0k}$, for $t = T, T-1, \ldots, 1$ the transition probabilities $p^t_{0k}(i, a, j) = p^t(i, a, j)$ for all $i, j \in C_{0k}$, $a \in A_{0k}(i)$, the rewards $r^t_{0k}(i, a) = r^t_{ia}$.

Furthermore, the class C_{0k} is closed then for $t = 1, \ldots, T$. From (3), we have

$$\begin{cases} \forall i \in C_{0k}, f^t_{0k}(i) = \underset{a \in A(i)}{\arg\max} \left\{ r^t_{ia} + \sum_{j \in E} p^t(i, a, j) x^{t+1}_j \right\}, \\ x^t = r(f^t_{0k}) + P(f^t_{0k}) x^{t+1}, \end{cases} \tag{4}$$

therefore, for all $i \in C_{0k}$ and $t = 1, \ldots, T$, $f^t_{0k}(i) = f^t(i)$ is an optimal action for the global *MDP* and

$$V^T_{0k}(i) = x^1_i = V^T(i). \tag{5}$$

Suppose that the result is true until the level $p-1$. Now we shall show that the result is still true in the level p.

The state space of the restricted MDP_{pk} is $S_{pk} = C_{pk} \bigcup \{j \in E_p: p^t_{iaj} > 0 \text{ for all } i\varepsilon C_{pk}, \quad a\varepsilon A(i)\}$, where

$$E_p = \bigcup \{i\varepsilon C_{mk}, \quad m = 0, 1, \ldots, p-1; k\varepsilon\{1, 2, \ldots, K(m)\}\}. \tag{6}$$

Let $R_{pk} = (f^1_{pk}, f^2_{pk}, \ldots, f^T_{pk})$ and V^T_{pk} are, respectively, an optimal policy and the optimal value vector in the restricted MDP_{pk}.

According to Theorem 1, we have for $t = T, T-1, \ldots, 1$:

$$\begin{cases} \forall i \in S_{pk}, f^t_{pk}(i) = \underset{a \in A_{pk}(i)}{\arg\max} \left\{ r^t_{pk}(i, a) + \sum_{j \in C_{pk}} p^t_{pk}(i, a, j) x^{t+1}_j \right\}, \\ x^t = r(f^t_{pk}) + P(f^t_{pk}) x^{t+1}, \end{cases} \tag{7}$$

Based on the definition of the restricted MDP_{pk}, for $i \in S_{pk}$, if $i \in C_{pk}$, $A_{pk}(i) = A(i)$, for $a \in A_{pk}(i)$, the rewards $r^t_{pk}(i, a) = r^t_{ia}$ and the transition probabilities $p^t_{pk}(i, a, j) = p^t(i, a, j)$ for all $j \in S_{pk}$.

In fact, that and since $p^t_{pk}(i, a, j) = 0, \forall i \in S_{pk}$, $\forall j\varepsilon(E/S_{pk})$, then for $t = 1, \ldots, T, f^t_{pk}$ verifies

$$\begin{cases} \forall i \in C_{pk}, f^t_{pk}(i) = \underset{a \in A(i)}{\arg\max} \left\{ r^t_{ia} + \sum_{j \in E} p^t(i, a, j) x^{t+1}_j \right\}, \\ x^t = r(f^t_{pk}) + P(f^t_{pk}) x^{t+1}. \end{cases} \tag{8}$$

```
(1) HBI (In MDP: $E, P, A, R, T$, Out ($V^T$, $R^*$)
(2) Find SCCs and their belonging levels using Tarjan's algorithm
(3) For each level Lp, $p = 0,\ldots,$ Lp Do
(4) For each class $Cl_{pk}$, in level $L_p$
(5) Construct the restricted finite-horizon $MDP_{pk}$
(6) End For
(7) End For
(8) For each level Lp,$p = 0,\ldots, L_p$ Do
(9) For each class $C_{pk}$, in level $L_p$ over planning horizon $T$, Do
(10) ABI ($MDP_{pk}$)
(11) End For
(12) End For
(13) Return $V^T$, $R^*$
```

Algorithm 3: Hierarchical backward induction.

```
Step 1. Determine the class $C_{mk}$ such that $i_0 \in C_{mk}$.
Step 2. Determine the classes $C_{nh}$, $n\in\{0, 1, \ldots, m\}$; $h \in\{1, 2,.., k(n)\}$ such that the end of any arc emanating from $C_{mk}$ is in the classes $C_{nh}$
Step 3. Solve the restricted MDPnh found in Step 2.
```

Algorithm 4: Accelerated HBI algorithm.

By consequence, for $i \in C_{pk}$ and $t=1, \ldots, T$, $f^t_{pk}(i) = f^t(i)$ is an optimal action for the global *MDP*, and, $V^T_{pk}(i) = x^1_i = V^T(i)$.

Now, It remains to see the case where $i \in (S_{pk}/C_{pk}), \sum m \in \{0,1,\ldots,p-1\}$, $\sum h \in \{1,2,\ldots,K(m)\}$: $i \in C_{mh}$ and $r^t_{pk}(i, \theta) = V^t_{mh}(i)/N$.

From the recurrence hypothesis $f^t_{mh}(i) = f^t(i)$ is an optimal action for the global *MDP*, calculated in previous levels, it remains to verify that $V^T_{pk}(i) = V^T(i)$.

Since for $i \in (S_{pk}/C_{pk})$ and $t=1, \ldots, T$, $f^t_{pk}(i) = \theta$ then $P(f^t_{pk})_{ii} = 1$ and $r(f^t_{pk})_i = V^t_{mh}(i)/N$.

It follows from (5):

$$\begin{aligned} V^T_{pk}(i) = x^1_i = \ldots &= \sum_{s=1}^{N}\left\{P\left(f^1_{pk}\right)P\left(f^2_{pk}\right)\ldots P\left(f^{s-1}_{pk}\right)r\left(f^s_{pk}\right)\right\} \\ &= \sum_{s=1}^{N}\frac{V^t_{mh}(i)}{N} = V^T(i). \end{aligned} \tag{9}$$

□

Remark 1. If the initial state i_0 is known, its optimal action $f(i_0)$ and its optimal value $V^T(i_0)$ are computed by solving just few restricted *MDPs*: one does not need to consider all states. The following algorithm (algorithm 4) explains this issue.

It is clear that, f(i0) and VT(i0) are obtained by solving only MDPmk. ◆

To demonstrate the benefit of the proposed *HBI* algorithm, we consider a case study of the racetrack problem described in the following section.

5. Case Study and Experimental Results

To show the advantage of the proposed *HBI* algorithm, we consider a standard control racetrack problem described by Martin Gardner [28] and Barto [29]. The goal is to control the movement of a race car along a predefined racetrack so that the racer can get to the finish line from starting the line in the minimum amount of time.

At each time step, the state of the racer is given by the tuples (X_t, Y_t, $V_x(t)$, $V_y(t)$) that represent the position and speed of the car in the *x, y* dimensions at time *t*. The actions are pairs $a = (a_x, a_y)$ of instantaneous accelerations, where a_x, ay ϵ {−1, 0, 1}. We assume that the road is 'slippery' and the car may fail to accelerate. An action a = (a_x, a_y) has its intended effect 90% of the time; 10% of the time the action effects correspond to those of the action $a_0 = (0; 0)$. Also, when the car hits a wall, its velocity is set to zero and its position is left intact. When the car is state (X_t, Y_t, $V_x(t)$, $V_y(t)$) and the action taken is $a = (a_x, a_y)$, it transit with 90% probability to a state $(X_t + V_x(t)+a_x,\ Y_t + V_y(t) + a_y,\ V_x(t)+a_x,\ V_y(t)+a_y)$.

Let s = (X_t, Y_t, $V_x(t)$, $V_y(t)$), a = (a_x, a_y) and s''= $(X_t + V_x(t)+a_x$, $Y_t + V_y(t)+a_y$, $V_x(t)+a_x$, $V_y(t)+a_y)$, the transition probability is defined as follows:

$$P(s'|s,a) = \begin{cases} 1, & \text{if } s\prime = s \text{ and } s = s'', \\ 0.9, & \text{if } s \neq s' \text{ and } s' = s'', \\ 0.1, & \text{if } s = s' \text{ and } s' \neq s'', \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

To complete the formulation of the finite-horizon *MDP* problem, we need to define the reward function and the horizon. Independently of the action taken, the immediate reward for all non-goal states is equal to −1, i.e., $R_i = -1$ and it is equal to zero for any goal state reached, i.e., $R_g = 0$. The horizon is determined after the decomposition into levels; indeed, during this decomposition, the maximum level obtained will be considered as the horizon.

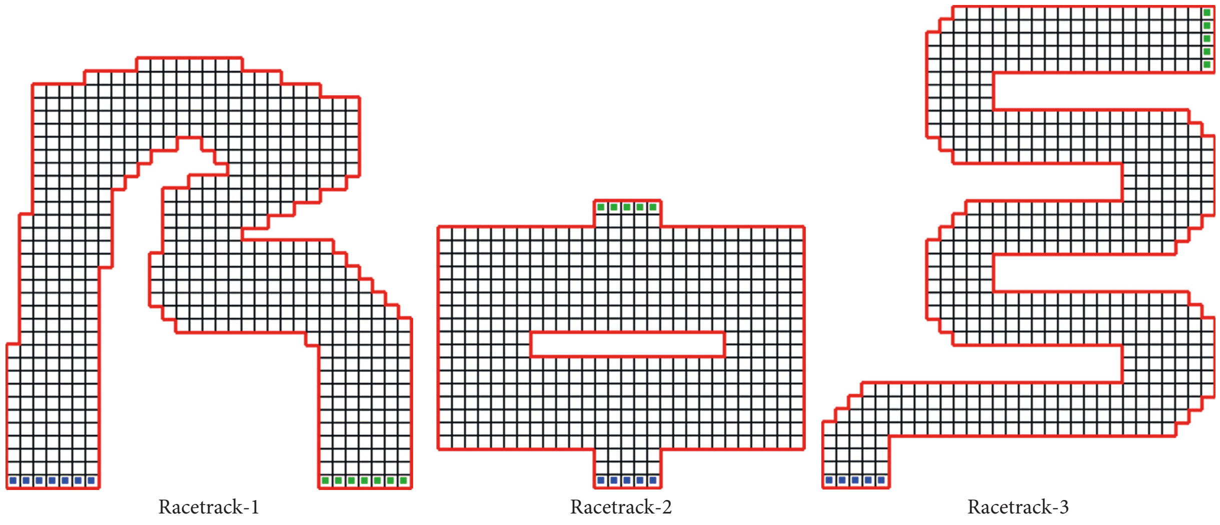

Figure 2: Racetrack's example.

Table 1: Characteristics of the three racetracks.

	MDP's horizon	Number of possible states	Number of SCC*s*
Race-1	31	132750	5120
Race-2	18	106200	5103
Race-3	48	123300	3554

Note. The value iteration (*VI*) algorithm under the infinite-horizon discounted *MDP* is used in [30] in order to solve racetrack problems. presents the comparison between *VI*, *BI*, and *HBI* algorithms. As it can be seen, the *BI* algorithm outperforms the *VI* algorithm, but the proposed *HBI* algorithm is more efficient than the *BI* algorithm.

Table 2: Performance of the *HBI* algorithm.

	Execution time (s)		
	VI algorithm	*BI* algorithm	*HBI* algorithm
Race-1	9, 7	4, 4	3
Race-2	6, 7	3, 6	2, 6
Race-3	8, 4	3, 9	2, 8

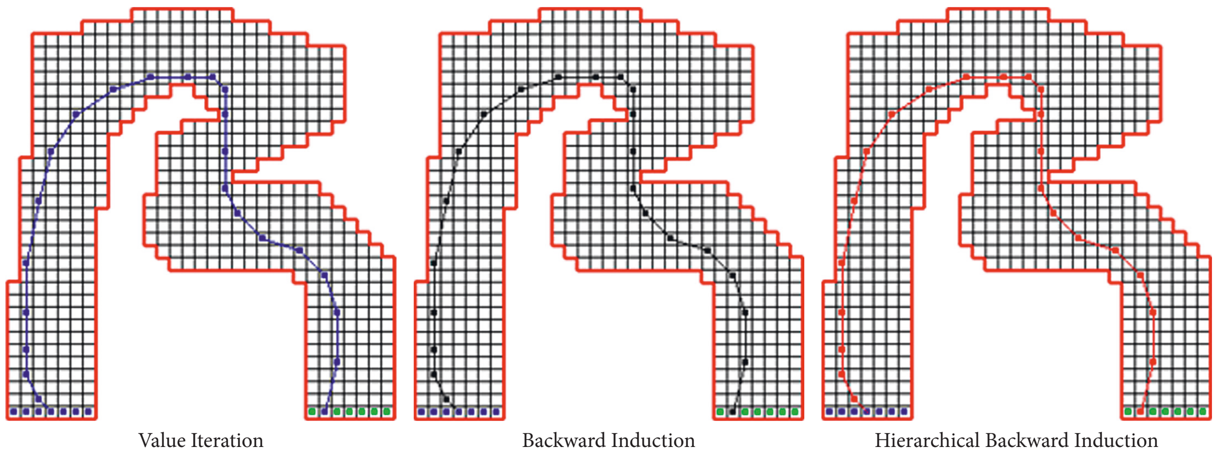

Figure 3: Road generated by VI, BI, and HBI algorithms for the racetrack-1.

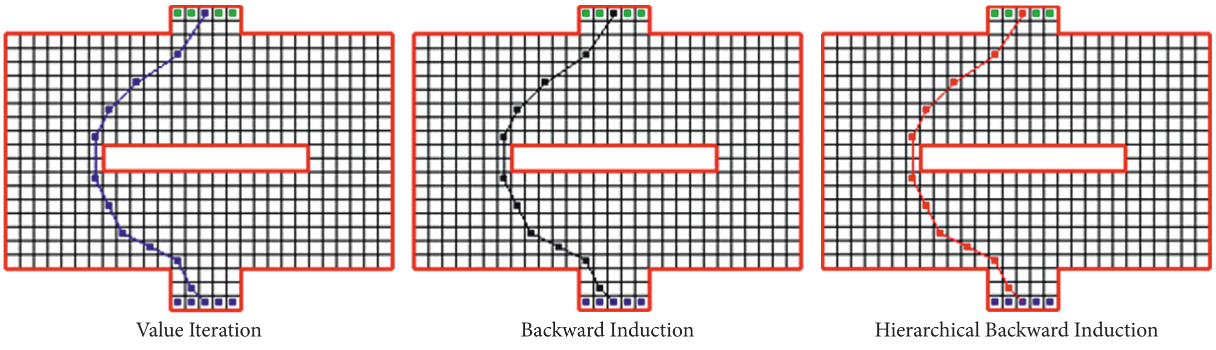

Figure 4: Road generated by VI, BI, and HBI algorithms for the racetrack-2.

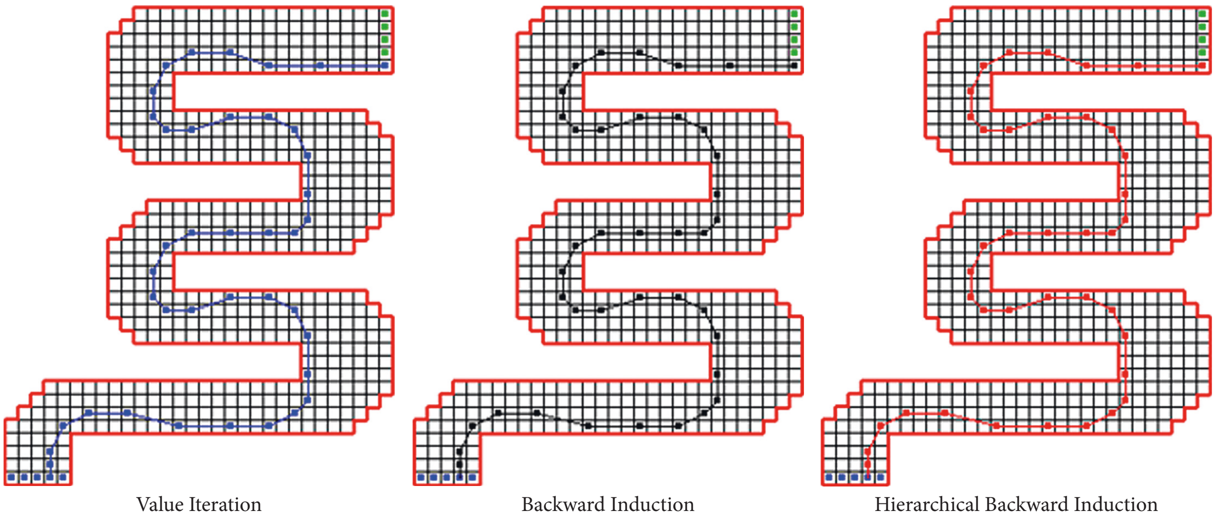

Figure 5: Road generated by VI, BI, and HBI algorithms for the racetrack-3.

To restrict the possible infinite state space, we assume that the speed of the car in the *x, y* dimensions are bounded in the range $[-7, +7]$, i.e., $V_x(t), V_y(t) \in [-7, +7]$. The speed will not change if the agent attempts to accelerate beyond these limits.

The proposed algorithms are tested using Intel(R) Core(TM) i7-6500 U (2.6 GHz), *C++* implementation, Windows 10 operating system (64 bits).

Figure 2 presents the three racetracks considered, blues cells represent the initial states, and green cells represent the goal states.

Table 1 presents the horizon, the number of *SCCs*, and the number of possible states obtained with the decomposition algorithm into levels for the three considered racetrack problems. As it can be seen, the number of states is reduced. Table 2.

Figures 3–5 show the policy obtained with *VI, BI*, and *HBI* algorithms for the three racetracks problems. As it can be seen, we obtain the same policy with the three algorithms.

6. Conclusion

In this paper, we have presented a new hierarchical backward induction algorithm for finite-horizon non-stationary *MDP* that is successful for large state spaces. It consists in decomposing the original problem into smaller restricted *MDPs*; indeed in each level and for each *SCC* a restricted finite-horizon *MDP* is constructed and solved independently from the other restricted *MDPs* of the same level. This proposed method accelerated the calculation time and reduces the memory requirement.

To show the advantage of the proposed *HBI* algorithm, we applied it to racetrack problems. The experimental results show that the *HBI* algorithm outperforms better the standard *BI* and value iteration algorithm. From a perspective, the use of parallelism techniques could further accelerate the convergence of the hierarchical finite-horizon *MDPs*.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. L. Puterman, *Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York USA, 1994.

[2] A. Larach and C. Daoui, "A new transformed stochastic shortest path with dead ends and energy constraint," *International Journal of Advanced Science and Technology*, vol. 129, pp. 43–58, 2019.

[3] D. Lefebvre and C. Daoui, "Control design for bounded partially controlled TPNs using timed extended reachability graphs and MDP," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 6, pp. 2273–2283, 2020.

[4] D. Lefebvre and C. Daoui, "Control design for untimed Petri nets using Markov decision processes," *Operations Research*, vol. 4, pp. 27–45, 2017.

[5] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the Complexity of Solving Markov Decision Problems," 2013, https://arxiv.org/abs/1302.4971.

[6] J. Bather, "Optimal decision procedures for finite Markov chains. Part II: communicating systems," *Advances in Applied Probability*, vol. 5, no. 3, pp. 521–540, 1973.

[7] K. W. Ross and R. Varadarajan, "Multichain Markov decision processes with a sample path constraint: a decomposition approach," *Mathematics of Operations Research*, vol. 16, no. 1, pp. 195–207, 1991.

[8] M. Abbad and H. Boustique, "A decomposition algorithm for limiting average Markov decision problems," *Operations Research Letters*, vol. 31, no. 6, pp. 473–476, 2003.

[9] C. Daoui, M. Abbad, and M. Tkiouat, "Exact decomposition approaches for Markov decision processes: a survey," *Advances in Operations Research*, vol. 2010, 2010.

[10] S. Chafik and C. Daoui, "A modified value iteration algorithm for discounted Markov decision processes," *Journal of Electronic Commerce in Organizations*, vol. 13, no. 3, pp. 47–57, 2015.

[11] A. Larach, S. Chafik, and C. Daoui, "Accelerated decomposition techniques for large discounted Markov decision processes," *Journal of Industrial Engineering International*, vol. 13, no. 4, pp. 417–426, 2017.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[13] F. Feng, B. Huang, K. Zhang, and S. Magliacane, "Factored adaptation for non-stationary reinforcement learning," 2022, https://arxiv.org/abs/2203.16582.

[14] C. E. Guestrin and G. Gordon, "Distributed planning in hierarchical factored MDPs," 2012, https://arxiv.org/abs/1301.0571.

[15] T. Tournaire, Y. Jin, A. Aghasaryan, H. Castel-Taleb, and E. Hyon, "Factored reinforcement learning for auto-scaling in tandem queues," in *Proceedings of the 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7, IEEE, Budapest, Hungary, 2022.

[16] C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artificial Intelligence*, vol. 121, no. 1-2, pp. 49–107, 2000.

[17] D. Kuinchtner, A. Sales, and F. Meneguzzi, "CP-MDP: A Candecomp-Parafac Decomposition Approach to Solve a Markov Decision Process Multidimensional Problem," 2021, https://arxiv.org/abs/2103.00331.

[18] D. Kuinchtner, F. Meneguzzi, and A. Sales, "A tensor-based Markov decision process representation," in *Proceedings of the Mexican International Conference on Artificial Intelligence*, pp. 313–324, Springer, Berlin, Germany, 2020.

[19] D. P. Smart, *Tensor Decomposition and Parallelization of Markov Decision Processes*, Massachusetts Institute of Technology, Massachusetts, USA, 2016.

[20] J. C. Gareau, É. Beaudry, and V. Makarenkov, "pcTVI: parallel MDP solver using a decomposition into independent chains," in *International Federation of Classification SocietiesIFCS 2022*IFCS, London, UK, 2022.

[21] A. Sapio, S. S. Bhattacharyya, and M. Wolf, "Efficient model solving for Markov decision processes," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–5, IEEE, Rennes, France, 2020.

[22] A. Sapio, S. S. Bhattacharyya, and M. Wolf, "Efficient solving of Markov decision processes on GPUs using parallelized sparse matrices," in *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pp. 13–18, IEEE, Porto, Portugal, 2018.

[23] F. Shang, Z. Zhang, Y. An, Y. Hu, and H. Liu, "Efficient parallel stochastic variance reduction algorithms for large-scale SVD," in *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*, pp. 172–179, IEEE, Beijing, China, 2019.

[24] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton USA, 1957.

[25] L. Kallenberg, *Lecture Notes Markov Decision Problems*, 2020.

[26] M. Abbad and C. Daoui, "Hierarchical algorithms for discounted and weighted Markov decision processes," *Mathematical Methods of Operations Research*, vol. 58, no. 2, pp. 237–245, 2003.

[27] A. R. Kristensen, "Hierarchic Markov processes and their applications in replacement models," *European Journal of Operational Research*, vol. 35, no. 2, pp. 207–215, 1988.

[28] M. Gardner, *Mathematical Games from Scientific American*, Simon Shuster, New York, USA, 1974.

[29] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, 1995.

[30] V. Mohan, *A Comparison of Various Reinforcement Learning Algorithms to Solve Racetrack Problem*, 2014.