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 Cha/f_ik 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 , then the optimal action , and the optimal value are computed by solving just the restricted MDP corresponding to the classes accessible from the class containing (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

\[ P(S_{t+1} = j | S_t = i, A_t = a) = p_{iaj}^t, \]

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 \( r_{ij}^t \) 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 \( \pi^* \) is given as the policy that maximizes the expected reward. \( \pi^* \) maximizes the value function of the Bellman equation [24]:

\[
V_t^f (\pi^*) = \sup_{\pi} V_t^f (\pi), i \in E, \text{ where } V_t^f (\pi) \text{ is the total expected reward in } T \text{ periods, given that the process starts from initial state } i \text{ and the policy } \pi \text{ is used.}
\]

\[
V_t^f (\pi) = \sum_{i=1}^{T} \sum_{j \in E} P_t (X_t = j, Y_t = a | X_1 = i) r_t^j, \tag{1}
\]

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 \( V_t^f \):

\[
V_t^f = \sup_{\pi} V_t^f (\pi), i \in E. \tag{2}
\]

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 \( x_t^{T+1} = 0 \), \( i \in S \). Define recursively for \( t = T, T - 1, \ldots, 1 \), a deterministic decision rule \( f_t \) and the vector \( x_t \) as follows:

\[
(i) \quad \{r(f_t^j)i_t + P(f_t^j)x_{t+1}^j\}_{i} = \max_{\pi} \left\{ r_{iaj}^t + \sum_{j} p_{iaj}^t x_{t+1}^j \right\}, i \in S,
\]

\[
(ii) \quad x_{i} = r(f_t^j) + P(f_t^j)x_{t+1}^j, \quad \text{then } R = (f_1, f_2, \ldots, f_T) \text{ is an optimal policy and } x_t \text{ is the optimal value vector } V_t^f.
\]

**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 \( \Gamma^a \), where \( \Gamma^a (i) = \{ j | E: p_{iaj} > 0, t = 1, \ldots, T \} \). This allowed to reduce the time complexity from \( O(T|A|E^2) \) to \( O(T^a|A|E^2) \) arithmetic operations, where \( |A| \) 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 ← t + 1;
(3) ∀ j ∈ E Take v_j^{t+1} ← 0;
(4) Repeat
(5) t ← t − 1:
(6) For each i ∈ E Do
(7) r(f'(f))_i + [P(f')_i]_i^{t+1} = \max_{a∈A(i)} \left\{ r'^a + \sum_{j∈Γ(i)} p^j_a v_j^{t+1} \right\} //The Deterministic Decision Rule f^t
(8) v' = r(f') + P(f') v^{t+1}
(9) For each i ∈ E Do
(10) R^* = (f^1, f^2, ..., f^T) is an optimal policy and v^1 is the optimal value vector.
11 Return V^T, R^*

Algorithm 1: Ameliorated backward induction.

Algorithm 2: Finding levels.

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

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) ∈ S: \exists a ∈ A(i) : P_{aij} > 0\} is a set of directed arcs. There exists a unique partition \( S = C_1 \cup C_2 \cup ... \cup C_p \) 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 \( i ∈ C_i; a ∈ A(i) : P_{aij} = 0 \) for all \( j ⫇ C_i \). 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 \( L_{p-1}, L_{p-2}, ..., L_0 \). 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 \( L_n, n = 0, 1, 2, ..., L \). Let (C_ik), \( k ∈ \{1, 2, ..., K(l)\} \) be the strongly connected classes corresponding to the nodes in level l (see Figure 1). Each class C_ik leads to a partial MDP_ik 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: $C_{10}$, $C_{30}$ and $C_{61}$ 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) $E_n = \bigcup \{iC_{nk}\}$, $p = 0, 1, \ldots, n - 1$ and $ke \{1, 2, \ldots, k(p)\}$.

(ii) States space: $S_{nk} = C_{nk} \cup \{jE_n\}$, $a \in A(i)$; $p_{i,j}^{nk} > 0, \ t = 1, \ldots , T$;

(iii) Actions space: For $i \in S_{nk}$, $A_{nk}(i) = \{A(i), i \text{ if } i \in C_{nk}; \theta, \text{ Otherwise}\}$;

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

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

(vii) If $i \in C_{nk}$, $r_{i,a}^{nk} = r_{i,a}^{nk}$.

N is the horizon and $V_{nk}^{T}$ is the optimal value vector of $MDP_{nk}$ 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, and 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 $V_{pk}^T$ are, respectively, an optimal policy and the optimal value vector in the restricted $MDP_{pk}$, then for all $i \in C_{pk}$ for $t = 1, 2, \ldots , T$, $f_t^{pk}(i)$ is an optimal action in the original MDP and $V_{pk}^T(i) = V^T(i)$.

**Proof.** The proof is by induction. For $p = 0$ (level $L_0$); $k \in \{1, 2, \ldots, K(0)\}$. Let $R_{ok} = (f_1^{ok}, f_2^{ok}, \ldots, f_T^{ok})$ and $V_{ok}^T$ are, respectively, an optimal policy and the optimal value vector in the restricted $MDP_{ok}$.

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

$$V_{ok}^T(i) = \max_{a \in A(i)} \left\{ r_{i,a}^{ok} + \sum_{j \in C_{nk}} p_{i,j}^{ok}(i,a) V_{nk}^{T+1}(j) \right\},$$

$$x^* = r\left(f_1^{ok}\right) + P\left(f_1^{ok}\right)x^{T+1},$$

from the definition of the restricted $MDP_{ok}$, the state space $S_{ok} = C_{ok}$, the action space $A_{ok}(i) = A(i)$ for $i \in S_{ok}$ for $T = T, T - 1, \ldots , 1$ the transition probabilities $p_{i,j}^{ok}(i,a,j) = p(i,a,j)$ for all $i,j \in C_{ok}$, $a \in A_{ok}(i)$, the rewards $r_{i,a}^{ok}(i,a) = r_{i,a}^{T+1}$.

Furthermore, the class $C_{ok}$ is closed then for $t = 1, 2, \ldots , T$.

From (3), we have

$$\forall i \in C_{ok}, f_{i,Spk}^{T}(i) = \max_{a \in A(i)} \left\{ r_{i,a}^{T} + \sum_{j \in E} p_{i,j}^{T}(i,a,j)x_{j}^{T+1} \right\},$$

$$x^* = r\left(f_0^{T}\right) + P\left(f_0^{T}\right)x^{T+1},$$

therefore, for all $i \in C_{ok}$ and $t = 1, 2, \ldots , T$, $f_{i,Spk}^{T}(i)$ is an optimal action for the global MDP and

$$V_{ok}^{T}(i) = x_{i}^{T} = V^{T}(i).$$

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 $S_{pk} = C_{pk} \cup \{jE_p\}$; $p_{i,j}^{pk} > 0$ for all $iC_{pk}$, $aC_{pk}(i)$, where $E_p = \bigcup \{iC_{mk}, \ m = 0, 1, \ldots , p - 1; k(1, 2, \ldots , K(m))\}$.

Let $R_{pk} = (f_1^{pk}, f_2^{pk}, \ldots, f_T^{pk})$ and $V_{pk}^T$ are, respectively, an optimal policy and the optimal value vector in the restricted $MDP_{pk}$.

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

$$\forall i \in C_{pk}, \ f_{i,Spk}^{T}(i) = \max_{a \in A(i)} \left\{ r_{i,a}^{T} + \sum_{j \in C_{nk}} p_{i,j}^{T}(i,a,j)x_{j}^{T+1} \right\},$$

$$x^* = r\left(f_1^{pk}\right) + P\left(f_1^{pk}\right)x^{T+1},$$

Based on the definition of the restricted $MDP_{pk}$, for $i \in S_{pk}$, if $iC_{pk}$, $A_{pk}(i) = A(i)$, for $a \in A_{pk}(i)$, the rewards $r_{i,a}^{T}(i,a) = r_{i,a}^{T+1}$ and the transition probabilities $p_{i,j}^{T}(i,a,j) = p(i,a,j)$ for all $j \in S_{pk}$.

In fact, that and since $p_{i,a}^{T}(i,a,j) = 0, \forall i \in S_{pk}, \forall j \in E(S_{pk})$, then for $t = 1, \ldots , T$, $f_{i,Spk}^{T}$ verifies

$$\forall i \in C_{pk}, \ f_{i,Spk}^{T}(i) = \max_{a \in A(i)} \left\{ r_{i,a}^{T} + \sum_{j \in C_{nk}} p_{i,j}^{T}(i,a,j)x_{j}^{T+1} \right\},$$

$$x^* = r\left(f_1^{pk}\right) + P\left(f_1^{pk}\right)x^{T+1}.$$
By consequence, for \( i \in C_{mk} \) and \( t = 1, \ldots, T \), \( f^t_{mk}(i) = f^t(i) \) is an optimal action for the global \( MDP \), and, \( V^T_{mk}(i) = x^1_i = V^T(i) \).

Now, it remains to see the case where \( i \in (S_{pk}/C_{pk}) \sum m \in \{0,1,\ldots,p-1\} \), \( \sum h \in \{1,2,\ldots,K(m)\} \) such that \( i \in C_{mh} \) and \( r^1_{pk}(i, \emptyset) = V^*_{mh}(i)/N \).

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 \( t = 1, \ldots, T \), \( f^t_{pk}(i) = \emptyset \) then \( P(f^t_{pk})_i = 1 \) and \( r(f^t_{pk})_i = V^*_{mh}(i)/N \).

It follows from (5):

\[
V^T_{pk}(i) = x^1_i = \ldots = \sum_{s=1}^{N} P(f^1_{pk})P(f^2_{pk}) \ldots P(f^{t-1}_{pk})r(f^t_{pk})
\]

(9)

\[
= \frac{\sum_{s=1}^{N} V^s_{mh}(i)}{N} = V^T(i).
\]

\[\Box\]

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, \( V(0) \) and \( V(T) \) are obtained by solving only \( MDP_{mk} \).

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 race car 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 \( a_x, a_y \in \{-1, 0, 1\} \). 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:

\[
P(s' | s, a) = \begin{cases} 
1, & \text{if } s' = s \text{ and } s = s'' \\
0.9, & \text{if } s \neq s' \text{ and } s' = s'' \\
0.1, & \text{if } s = s' \text{ and } s' \neq s'' \\
0, & \text{otherwise.}
\end{cases}
\] (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.
Table 1: Characteristics of the three racetracks.

<table>
<thead>
<tr>
<th></th>
<th>MDP’s horizon</th>
<th>Number of possible states</th>
<th>Number of SCCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Race-1</td>
<td>31</td>
<td>132750</td>
<td>5120</td>
</tr>
<tr>
<td>Race-2</td>
<td>18</td>
<td>106200</td>
<td>5103</td>
</tr>
<tr>
<td>Race-3</td>
<td>48</td>
<td>123300</td>
<td>3554</td>
</tr>
</tbody>
</table>

Note. The value iteration (VI) algorithm under the infinite-horizon discounted MDP is used in [30] in order to solve racetrack problems. It 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.

<table>
<thead>
<tr>
<th></th>
<th>VI algorithm</th>
<th>BI algorithm</th>
<th>HBI algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Race-1</td>
<td>9, 7</td>
<td>4, 4</td>
<td>3</td>
</tr>
<tr>
<td>Race-2</td>
<td>6, 7</td>
<td>3, 6</td>
<td>2, 6</td>
</tr>
<tr>
<td>Race-3</td>
<td>8, 4</td>
<td>3, 9</td>
<td>2, 8</td>
</tr>
</tbody>
</table>

Figure 2: Racetrack’s example.

Figure 3: Road generated by VI, BI, and HBI algorithms for the racetrack-1.
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) i7-6500U (2.6GHz), C++ implementation, Windows 10 operating system (64 bits).

Figure 2 presents the three racetracks considered, blue 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 racetrack 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


