Systems & Control Letters xxx (xxxx) xxx

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle

Large-scale dynamic system optimization using dual decomposition method with approximate dynamic programming

Pegah Rokhforoz [a], Hamed Kebriaei [a,b,*], Majid Nili Ahmadabadi [a]

[a] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran, Iran

ARTICLE INFO

Article history:
Received 26 April 2020
Received in revised form 4 December 2020
Accepted 29 January 2021
Available online xxxx

Keywords:
Multi-agent system
Approximate dynamic programming
Duality theory
Coupling constraint

ABSTRACT

In this paper, multi-agent dynamic optimization with a coupling constraint is studied. The aim is to minimize a strongly convex social cost function, by considering a linear stochastic dynamics for each agent and also coupling constraints among the agents. In order to handle the coupling constraint and also, to avoid high computational cost imposed by a centralized method for large scale systems, the dual decomposition method is used to decompose the problem into multiple individual sub-problems, while the dual variable is adjusted by a coordinator. Nevertheless, since each sub-problem is not a linear–quadratic (LQ) optimal control problem, and hence its closed-form solution does not exist, approximate dynamic programming (ADP) is utilized to solve the sub-problems. The main contribution of the paper is to propose an algorithm by considering the interrelated iterations of dual variable adjustment and ADP, and to prove the convergence of the algorithm to the global optimal solution of the social cost function. Additionally, the implementation of the proposed algorithm using a neural network is presented. Also, the computational advantage of the proposed algorithm in comparison with other bench-marking methods is discussed in simulation results.

1. Introduction

Multi-agent optimization with coupling constraint has been studied extensively in a variety of systems including robotics [1,2], power systems [3], human operators [4], and wireless systems [5]. In some applications of multi-agent optimization, such as resource allocation to the robotic system [6], human operators system [4], and smart grid system [7], we face the dynamical agents and therefore, finding the optimal solution is regarded as a dynamic programming problem.

Centralized optimization method is one way to obtain a social optimal solution [8,9]. However, for large scale systems, a high computational cost is imposed on the central system. Also, the central system needs to know all the agents' cost functions and so, their privacy is not preserved. To address the mentioned challenges, dual decomposition is a useful method, where the dual problem can be separated in the subsystems despite the presence of coupling constraint. For instance, dual decomposition method is a common approach that can be utilized to solve multi-agent optimization with coupling constraints [10–14]. In this approach, the dual problem can be solved using an iterative algorithm, e.g. the sub-gradient method. Moreover, there are some papers in the literature [15,16] which propose a primal–dual method to solve the distributed optimization problem.

On the other side, the dynamic of agents makes the primal problem as an optimal control problem. In this instance, although we can handle the coupling constraints between agents using a dual decomposition approach, but each agent has to solve a dynamic optimization (optimal control) problem using Bellman's equation or dynamic programming method, which is a challenging problem for a general cost function (e.g. non-quadratic). In this case, a closed-form solution of Bellman's equation does not exist and the dynamic programming (DP) method imposes a high computational cost due to the "curse of dimensionality" [17].

An approximate dynamic programming (ADP) method was developed to address the computational challenges of a general optimal control problem. There are many research papers in the literature those have studied ADP method (e.g.[18–21]). The authors in [18,19], proposed an ADP method to solve an infinite horizon optimal control problem for discrete-time systems. The case for finite horizon has been also studied using ADP in [20,21]. Multi-agent systems are not considered in these works.

In the literature on ADP in multi-agent systems, most of the works have considered communication among the agents without considering any coupling constraints among them (which is a necessary part in some applications such as resource allocation problem). In [22,23], the authors consider multi-agent differential graphical games with continuous dynamics and suggest an ADP

* Corresponding author at: School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran.
E-mail address: Kebriaei@ut.ac.ir (M.N. Ahmadabadi).

https://doi.org/10.1016/j.sysconle.2021.104894
0167-6911/

approach to solve the problem. In these papers, agents interact with each other under a graphical topology, but there is no coupling constraint among them. In [24], a multi-agent system with discrete-time dynamics is studied. The agents communicate with each other through a graphical topology. Each agent solves a coupled HJB equation in real-time using the ADP method. The learning algorithm using ADP method for cooperative game theory with constraint input is proposed in [25]. In this paper, agents have the discrete-time dynamic and the state of the system coupled to each other, however there is not any coupling constraints among the control inputs.

Distributed Model predictive Control (DMPC) is another alternative approach to find a decentralized solution for a constrained optimal control problem. There are many researches which have investigated the DMPC approach for distributed optimization of multi-agent systems subject to coupling constraints, [26,27]. Among the advantages of DMPC in handling the state constraints [28], generally, they cannot guarantee the feasibility solution and the global optimality for all strongly convex problems [29] and [30]. There are some researches like [28,31,32] that prove the feasibility solution under some conditions, nevertheless, the obtained solutions are sub-optimal. The authors of [31], propose a DMPC using duality approach for the linear dynamic system with the quadratic cost function. They prove that the proposed solution is feasible but sub-optimal.

In this paper, we study the problem of multi-agent optimization where each agent has a general strongly convex cost function and a linear dynamic. We propose a decomposition algorithm using the combination of dual method and approximate dynamic programming (ADP). In our proposed algorithm, at each iteration, the central system updates and broadcasts the Lagrange variables (dual variables), which are associated with the coupling constraints, to the agents. Each agent performs a one-step ADP to update its value function (cost function) based on the Lagrange multipliers sent by the coordinator. Then, the updated control signals of all the agents are sent back to the coordinator to update the Lagrange multipliers for the next iteration. This producer continues until the convergence of the algorithm occurs. The convergence of the proposed algorithm to the global optimal solution has been proved. Regarding the computational advantages of the proposed method, since we employ ADP rather than DP, we do not face the "curse of dimensionality". Moreover, when the number of agents increases, the dual decomposition method makes the algorithm computationally tractable.

To the best of our knowledge, this is the first paper that proposes a decomposition algorithm based on the duality theory and ADP method to obtain the optimal solution of the multi-agent optimization with coupling constraint, linear dynamics, and general strongly convex cost function. Comparing with the literature of the concept, we study the optimization problem for dynamic agents, thus it is different from the literature of decomposition methods for static optimization problems [11,12,15,16]. Our proposed algorithm is also different from [22–24], since in these papers the coupling constraints among the agents have not been considered.

The main contributions of our paper are summarized in the following:

- We address the large-scale optimization problem with coupling constraint in a decentralized framework for dynamic agents.
- We propose a novel decentralized algorithm based on the combination of duality theory and ADP method where the agents perform a one-step ADP method in each iteration of the algorithm, hence it has a low computational cost.
- We prove the convergence of the proposed algorithm to the global optimal solution. We validate this theoretical proof in numerical test.

This paper is organized as follows. We represent the formulation of the optimization problem in Section 2. In Section 3, the dual decomposition method is demonstrated. Section 4 represents the algorithm based on the combination of dual method and ADP. The convergence and optimality of the proposed algorithm are proven in Section 5. The implementation of the algorithm using neural networks is provided in Section 6. The simulation results are shown in Section 7. The conclusion remarks are made in Section 8.

Notations: Given N vectors $y^i(n) \in \mathbb{R}^{n_i}$, $y^i(n)^\top$ represents the transpose of $y^i(n)$. $y^i = \mathbf{col}(y^i(1), \dots, y^i(N)) = [y^i(1)^\top, \dots, y^i(N)^\top]^\top \in \mathbb{R}^{Nn_i}$ and for given vectors $y^i \in \mathbb{R}^{Nn_i}$, $i = 1, \dots, M$, $y = \mathbf{col}(y^1, \dots, y^M) = [y^{1\top}, \dots, y^{M\top}]^\top \in \mathbb{R}^{NMn_i}$. Throughout the text, the index of agent is denoted by superscript as y^i and the power is denoted by parenthesis as $(y)^i$.

2. System model and problem formulation

We denote the set of agents by $\mathcal{I} = \{1, 2, \dots, I\}$. Let $x^i(n) \in \mathcal{X}^i(n) \subseteq \mathbb{R}^p$ be the random variable representing the state of agent i at time n, where $\mathcal{X}^i(n)$ is the state space. We assume that the states of the system evolve through a linear dynamic as follows

$$x^i(n+1) = A^i x^i(n) + B^i u^i(n) + w^i(n), \tag{1}$$

where, $u^i(n) \in \mathcal{U} \subseteq \mathbb{R}^q_+$ be an action of agent i at time n, where $\mathcal{U}$ is the action space. $w^i(n) \in \mathbb{R}^p$ is a random disturbance of agent i at time n. $A^i \in \mathbb{R}^{p\times p}$, and $B^i \in \mathbb{R}^{p\times q}$ are the system matrix, and input matrix, respectively.

Let $U^i : \mathbb{R}^p \times \mathbb{R}^q_+ \to \mathbb{R}_+$ and $U^i_N : \mathbb{R}^p \to \mathbb{R}_+$ be the cost function at time n and terminal cost of agent i, respectively. The central system seeks to find the control input u, which minimizes the expected sum of cost functions over the horizon N as follows

$$\begin{cases} \min_u \quad \mathbf{E}_w \left\{ \sum_{i=1}^{I} \left(U^i_N\big(x^i(N)\big) + \sum_{m=1}^{N-1} U^i\big(x^i(m), u^i(m)\big) \right) \right\}, \\ \text{s.t.} \qquad \mathrm{C}_1 : \ \sum_{i=1}^{I} u^i(n) \le b(n), n = 1, \dots, N-1, \\ \qquad\quad \mathrm{C}_2 : \ u^i(n) \in \mathcal{U}, i \in \mathcal{I}, n = 1, \dots, N-1, \end{cases} \tag{2}$$

where, $u = \mathbf{col}(u^1, \dots, u^I)$, $u^i = \mathbf{col}(u^i(1), \dots, u^i(N))$, C_1 is the coupling constraint among agents, and $b(n) \in \mathbb{R}^q_+$ is a constant vector. $\mathcal{U} \subseteq \mathbb{R}^q_+$ is the local constraint set.

We consider the following assumptions

Assumption 1. The disturbances $w^i(n) \in \mathcal{W}$ are independent random vectors with a known probability distribution which do not depend on $x^i(n)$ and $u^i(n)$, $\forall i \in \mathcal{I}$ and $n = 1, \dots, N-1$.

Assumption 2. The set $\mathcal{U} \subseteq \mathbb{R}^q_+$, $\mathcal{W} \subseteq \mathbb{R}^p$, and $\mathcal{X}^i(n) \subseteq \mathbb{R}^p$, $i \in \mathcal{I}$, $n = 1, \dots, N$, are compact and convex sets.

Remark 1. Since a linear transformation maps a convex and compact set into a convex and compact set (Chapter (3) of [33]), hence under a linear dynamic which is in fact a linear mapping, and by considering that the control input $(u^i(n))$, disturbance $(w^i(n))$, and initial state $(x^i(0))$ are from the convex and compact sets $\mathcal{U}$, $\mathcal{W}$ and $\mathcal{X}^i(0)$, respectively (according to Assumption 2), the state of the system (1) at time-step n belongs to a convex and compact set $\mathcal{X}^i(n)$, $i \in \mathcal{I}$.

Assumption 3. The function $U^i : \mathbb{R}^p \times \mathbb{R}_+^q \to \mathbb{R}_+$ and $U_N^i : \mathbb{R}^p \to \mathbb{R}_+$ are strongly convex and bounded, $i \in \mathcal{I}$.

In the case of complete information, (utility functions and dynamics are known), Problem (2) can be solved in a centralized manner. However, it has a high computational cost when the number of agents I is large. Dual decomposition decomposes problem (2) into I sub-problems, where each agent minimizes its own cost function subject to its local constraints, while only the coupling constraint is managed by a central coordinator who adjusts the corresponding dual variable. The global solution of (1) will be found by iterating between coordinator and agents which is explained in the next section.

3. Dual decomposition method

Let us define the Lagrangian function of problem (2) by considering the coupling constraints C_1 as follows:

$$L(x,u,\lambda) = \mathbf{E}_w\left\{\sum_{i=1}^{I}\left(U_N^i\left(x^i(N)\right) + \sum_{m=1}^{N-1} L^i\left(x^i(m),u^i(m),\lambda(m)\right)\right)\right\}$$
$$= \mathbf{E}_w\left\{\sum_{i=1}^{I}\left(U_N^i\left(x^i(N)\right) + \sum_{m=1}^{N-1}\left(U^i\left(x^i(m),u^i(m)\right) + \lambda(m)^\top u^i(m)\right)\right) - \sum_{m=1}^{N-1}\lambda(m)^\top b(m)\right\},$$

where $\lambda(m) \in \mathbb{R}_+^q$, $m = 1,\ldots,N-1$, is the non-negative Lagrange multiplier associated with the coupling constraint C_1.

Under Assumption 2 the coupling and local constraints C_1 and C_2 in (2) are convex. Assumption 3 guarantees that the objective function in (2) is strongly convex, and also the linear dynamic (1) is a convex equality constraint. Therefore, Problem (2) is strongly convex and hence, the strong duality holds (Section 5.2.3 of [34]). This means that problem (2) can be solved in the dual domain. Hence, we can alternatively solve the dual problem of (2) as follows

$$D: \max_{\lambda\geq 0}\min_{u} L(x,u,\lambda)$$
$$= \max_{\lambda\geq 0}\left\{-\sum_{m=1}^{N-1}\lambda(m)^\top b(m) + \sum_{i=1}^{I}\min_{u^i\in\mathcal{U}} \times \mathbf{E}_{w^i}\left\{\left(U_N^i\left(x^i(N)\right) + \sum_{m=1}^{N-1} L^i\left(x^i(m),u^i(m),\lambda(m)\right)\right)\right\}\right\}. \quad (3)$$

Dual problem D can be decomposed into I sub-problems, where each agent solves the following optimal control problem

$$\min_{u^i} \mathbf{E}_{w^i}\left\{U_N^i\left(x^i(N)\right) + \sum_{m=1}^{N-1} L^i\left(x^i(m),u^i(m),\lambda(m)\right)\right\}$$
$$\text{s.t.}\quad x^i(n+1) = A^i x^i(n) + B^i u^i(n) + w^i(n), \quad u^i(n)\in\mathcal{U},\quad n = 1,\ldots,N-1. \quad (4)$$

As we mentioned above, (3) can be solved in an iterative fashion, where the central system updates the dual variables (λ) using a sub-gradient method and broadcasts it to all agents. Agent i, $\forall i \in \mathcal{I}$, aims to solve (4) for each $x^i(0)$ by considering the value of dual variables in the last iteration. In the next section we explain the method which can solve (4).

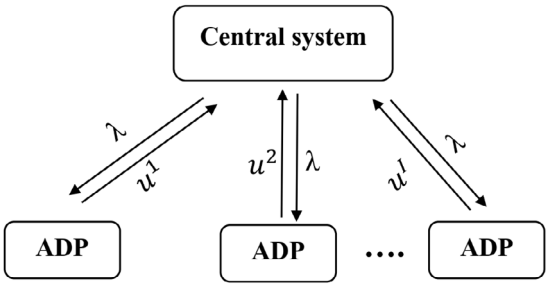

Fig. 1. The schematic of the decomposition algorithm.

4. Combination of ADP and dual decomposition method

Since the cost function of (4) is in the general form, Bellman's equation does not admit a closed-form solution. Moreover, the dynamic programming method results in "curse of dimensionality", when the number of horizon N increases. Hence, to face these challenges an ADP algorithm is employed, where the dynamic of an agent is handled by a local Bellman's equation and the cost functions and control inputs are updated using an ADP method.

In our proposed algorithm, the central system does not need to wait at each iteration, until the convergence of the local ADP of agents occurs. Instead, each agent only performs a one-step ADP update and sends the value of its control input to the central system. The central system updates the Lagrange multipliers based on those values and then broadcasts these updated values to the agents. This process continues until the algorithm converges. The schematic of this algorithm is shown in Fig. 1.

Our proposed algorithm has two main advantages. First, the central system does not need to know the agents' cost functions and dynamics, hence it preserves the privacy information of agents. Second, it has a low computational time in comparison with the case that the central system has to wait until the convergence of ADP occurs in each iteration and with using a dynamic programming method.

In what follows, we explain our proposed algorithm which is the combination of dual decomposition and ADP method.

In our proposed iterative solution, at iteration $k+1$ and step n, agent i aims to find u^i which minimizes the following cost-to-go function

$$V^i(x^i(n)) = \mathbf{E}_{w^i}\left\{U_N^i\left(x^i(N)\right) + \sum_{m=n}^{N-1} L^i\left(x^i(m),u^i(m),\lambda(m)\right)\right\}, \quad (5)$$

where $\lambda(m)$, is the Lagrange multiplier which is given to the agents by the central system.

In the rest of the paper, to simplify the notation we define $f^i(x^i(n),u^i(n),w^i(n)) := A^i x^i(n) + B^i u^i(n) + w^i(n)$.

Eq. (5) can be written as

$$V^i(x^i(n)) = L^i(x^i(n),u^i(n),\lambda(n)) + \mathbf{E}_{w^i}\left\{V^i(f^i(x^i(n),u^i(n),w^i(n)))\right\}, n = 1,\ldots,N-1, \quad (6)$$
$$V^i(x^i(N)) = U_N^i\left(x^i(N)\right).$$

The optimal cost-to-go function $V^{*i}(x^i(n))$ must satisfy the following Bellman's equation

$$V^{*i}(x^i(n)) = \min_{u^i(n)\in\mathcal{U}}\Big[L^i(x^i(n),u^i(n),\lambda(n)) + \mathbf{E}_{w^i}\left\{V^{*i}(f^i(x^i(n),u^i(n),w^i(n)))\right\}\Big], \quad (7)$$
$$V^{*i}(x^i(N)) = U_N^i\left(x^i(N)\right).$$

Since there is not any closed-form solution to (7), ADP is an efficient method to obtain the solution of (7). Here, we propose a novel ADP method which works with the coupling constraint (C_1). In an iterative procedure, the estimate of optimal control input and optimal cost-to-go function of agent i at iteration k + 1 denoted by $u^i_{k+1}(n)$ and $V^i_{k+1}(x^i(n))$, respectively, are updated as follows

$$u^i_{k+1}(n) = \arg\min_{u^i(n)\in\mathcal{U}} \Big[L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) + \mathbf{E}_{w^i}\Big\{ V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\}\Big], \tag{8}$$

$$\begin{cases} V^i_{k+1}(x^i(n)) = & \big(1-c_k(n)\big)V^i_k(x^i(n)) \\ & + \big(c_k(n)\big)^2\Big[L^i\big(x^i(n), u^i_{k+1}(n), \lambda_k(n)\big) \\ & + \mathbf{E}_{w^i}\Big\{ V^i_k\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\}\Big], \\ V^i_{k+1}(x^i(N)) = & \big(1-c_k(N)\big)V^i_k(x^i(N)) + \big(c_k(N)\big)^2 U^i_N\big(x^i(N)\big), \end{cases} \tag{9}$$

where $c_k(n)$ is the learning rate at iteration k and time-step n. In fact, (8) calculates the optimal control input at iteration k for agent i based on the Bellman's equation (7) by using the cost-to-go function at iteration k which is updated in (9). In other words, (8) and (9) represent a **recursive structure** for computing the optimal control strategy and optimal cost-to-go function in (7), respectively. Eqs. (8) and (9) cannot be solved explicitly, since we do not have any explicit expression for the cost-to-go function (V^i_k) in (8) and (9). To obtain the solution of (8) and (9), in Section 6, we propose an actor–critic method with two neural networks as the general function approximators to approximate both the control strategy and cost-to-go function, respectively. The weights of these neural networks are adjusted adaptively to obtain the optimal solution of control strategy and also optimal cost-to-go function. In Theorem 4, we show that for large enough number of basis functions in neural networks, the approximation errors tend to zero. Therefore, in our analysis we assume that Eqs. (8) and (9) can be solved. The details on computation of the solution to Eqs. (8) and (9) are given in Section 6.

Then, using the updated control signal in (8), the central system updates Lagrange multiplier at iteration $k+1$ using sub-gradient method as follows

$$\lambda_{k+1}(n) = \Big[\lambda_k(n) + c_k(n)\big(\sum_{i=1}^{I} u^i_{k+1}(n) - b(n)\big)\Big]^+, \tag{10}$$

where $[\cdot]^+ = max(0, \cdot)$.

The updated Lagrange multiplier of the coupling constraint (10) is used again in (8) and (9) for the next iteration and the above procedure repeats until the convergence occurs. In our analysis, since the weighted average of the sequence, $\{u^i_{r+1}(n)\}^k_{r=0}$ shows better convergence properties than $u^i_k(n)$ [13,14,35,36], the convergence of the following auxiliary variable to the optimal solution of problem D is examined.

$$\tilde{u}^i_{k+1}(n) = \frac{\sum^k_{r=0} c_r(n) u^i_{r+1}(n)}{\sum^k_{r=0} c_r(n)}. \tag{11}$$

Therefore, we can summarize the iterative solution in Algorithm 1.

Algorithm 1 Combination of ADP and dual decomposition method

1. **Initialize the parameters**: $x^i(0)$, $\lambda_0(n)$, $V^i_0(\cdot)$, $n = 1, \ldots, N$, $i \in \mathcal{I}$.
2. **For** $i \in \mathcal{I}$ **and** $n = 1, \ldots, N$ **repeat**
3. Compute the control input using (8).
4. Compute the cost-to-go function using (9).
5. Apply the control input to the system and obtain the next state using (1).
6. Update Lagrange multiplier using (10).
7. Compute $\tilde{u}^i_k(n)$ using (11)
8. $k = k + 1$.

5. Convergence and optimality

In what follows, the convergence of the proposed algorithm to the optimal solution has been proved. The first theorem shows the convergence of the dual variables. The feasibility and optimality of the solution have been proved through Theorems 2 and 3, respectively.

Assumption 4. $0 < c_k(n) \le 1$ and $\{c_k(n)\}_{k\ge 0}$ is a non increasing sequence such that $\sum^\infty_{k=0} c_k(n) = \infty$, $\sum^\infty_{k=0}(c_k(n))^2 < \infty$, and $c_\infty(n) = 0, n = 1, \ldots, N-1$.

One example for $\{c_k(n)\}_{k\ge 0}$ satisfying Assumption 4 is $c_k(n) = \frac{1}{k}$.

Theorem 1. *The dual variables converge to their optimal values.*

$$\lim_{k\to\infty} \|\lambda_k(n) - \lambda^*(n)\| = 0, n = 1, \ldots, N-1. \tag{12}$$

Proof. According to (10) and Assumption 4, the iteration of sub-gradient method converges to its optimal value and the proof is provided in Proposition 6.3.1 in [37]. □

Theorem 2. $\lim_{k\to\infty} \tilde{u}^i_k(n), i \in \mathcal{I}, n = 1, \ldots, N-1$, *is a feasible solution of problem* (2).

Proof. To prove the feasibility solution we have to show that $\lim_{k\to\infty} \sum^I_{i=1} \tilde{u}^i_k(n) \le b(n)$, $n = 1, \ldots, N-1$, for each $x^i(0)$, $i \in \mathcal{I}$. Based on (11), we have that

$$\begin{aligned} \sum_{i=1}^{I} \tilde{u}^i_{k+1}(n) &= \sum_{i=1}^{I}\Big(\frac{\sum^k_{r=0} c_r(n) u^i_{r+1}(n)}{\sum^k_{r=0} c_r(n)}\Big) \\ &\le \frac{\sum^k_{r=0}\sum^I_{i=1}\big(\lambda_{r+1}(n) - \lambda_r(n) + c_r(n)b(n)\big)}{\sum^k_{r=0} c_r(n)} \\ &= \frac{\sum^I_{i=1}\big(\lambda_{k+1}(n) - \lambda_0(n)\big) + \sum^k_{r=0} c_r(n) b(n)}{\sum^k_{r=0} c_r(n)}, n = 1, \ldots, N-1, \end{aligned} \tag{13}$$

where the first inequality is obtained by (10) which implies that

$$\lambda_{k+1}(n) \geq \lambda_k(n) + c_k(n)\sum_{i=1}^{I}\big(u_{k+1}^i(n) - b(n)\big), \quad n = 1, \ldots, N-1. \tag{14}$$

Theorem 1 implies that the sequence $\{\lambda_k(n)\}_{k\geq 0}$ converges to $\lambda^*(n)$. Hence, $\{\lambda_k(n)\}_{k\geq 0}$ is a bounded sequence, and based on Assumption 4, we can deduce that

$$\lim_{k\to\infty}\sup\sum_{i=1}^{I}\tilde{u}_{k+1}^i(n) \leq b(n)\frac{\sum_{r=0}^{k} c_r(n)}{\sum_{r=0}^{k} c_r(n)}, \; n = 1, \ldots, N-1, \tag{15}$$

so we have

$$\lim_{k\to\infty}\sup\sum_{i=1}^{I}\tilde{u}_{k+1}^i(n) \leq b(n), \; n = 1, \ldots, N-1. \quad \square \tag{16}$$

We present the proof of following Lemmas in Appendix.

Lemma 1. *Let $u^{*i}(n)$ and $\lambda^*(n)$, $i \in \mathcal{I}$, $n = 1, \ldots, N-1$ denote the optimal primal and Lagrange multiplier of (2). Then, we have*

$$\begin{aligned}&\tilde{V}\big(x^i(n), u^{*i}(n), \lambda(n)\big) \leq \tilde{V}\big(x^i(n), u^{*i}(n), \lambda^*(n)\big)\\ &\leq \tilde{V}\big(x^i(n), u^i(n), \lambda^*(n)\big),\\ &i \in \mathcal{I}, n = 1, \ldots, N-1,\end{aligned} \tag{17}$$

where

$$\begin{aligned}&\tilde{V}\big(x^i(n), u^i(n), \lambda(n)\big)\\ &= L^i\big(x^i(n), u^i(n), \lambda(n)\big) + \mathbf{E}_{w^i}\big\{V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\big\}.\end{aligned} \tag{18}$$

Lemma 2. *The sequence $\{V_k^i(x^i(n))\}_{k\geq 0}$, $i \in \mathcal{I}$, $n = 1, \ldots, N$, which is defined in (9), is bounded.*

Lemma 3. *The sequence $\{\mathbf{E}_{w^i}\big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\big\}\}_{k=0}^{\infty}$, $i \in \mathcal{I}$, $n = 1, \ldots, N$, is a bounded sequence.*

Lemma 4. *Let $V^{*i}(x^i(n))$ be the optimal cost-to-go function and $\{V_k^i(x^i(n))\}_{k=0}^{\infty}$ be the iterative cost-to-go function which is updated as (9), then we have*

$$\sum_{k=0}^{\infty} c_k(n)\big(V_k^i(x^i(n)) - V^{*i}(x^i(n))\big) < \infty. \tag{19}$$

Corollary 1. *The sequence $\{V_k^i(x^i(n))\}_{k\geq 0}$, converges to its optimal value, for all initial value of $x^i(0)$, $i \in \mathcal{I}$.*

$$\lim_{k\to\infty} V_k^i(x^i(n)) = V^{*i}(x^i(n)), \quad i \in \mathcal{I}, n = 1, \ldots, N, \tag{20}$$

Proof. Assumption 4 ($\sum_{k=0}^{\infty} c_k(n) = \infty$) together with Lemma 4 and $V_k^i(x^i(n)) - V^{*i}(x^i(n)) \geq 0$ imply that

$$\lim_{k\to\infty} V_k^i(x^i(n)) = V^{*i}(x^i(n)), i \in \mathcal{I}, n = 1, \ldots, N-1. \quad \square \tag{21}$$

Theorem 3. *The sequence $\{\tilde{u}_k^i(n)\}_{k\geq 0}$ converges to its optimal value, $i \in \mathcal{I}$.*

$$\lim_{k\to\infty} \|\tilde{u}_k^i(n) - u^{*i}(n)\| = 0, i \in \mathcal{I}, n = 1, \ldots, N-1, \tag{22}$$

where $u^{*i}(n)$ is the optimal solution of problem (2).

Proof. Notice that, by using (10), we have

$$\begin{aligned}&\|\lambda_{k+1}(n) - \lambda(n)\|^2 \leq \|\lambda_k(n) - \lambda(n) + c_k(n)G_{k+1}^n\|^2\\ &= \|\lambda_{k+1}(n) - \lambda(n)\|^2 + 2c_k(n)G_{k+1}^n\big(\lambda_k(n) - \lambda(n)\big)\\ &\quad + \|c_k(n)G_{k+1}(n)\|^2,\\ &n = 1, \ldots, N-1,\end{aligned} \tag{23}$$

where $\lambda(n)$ has a non-negative value and $G_{k+1}(n) = \sum_{i=1}^{I}\big(u_{k+1}^i(n) - b(n)\big)$.

Adding and subtracting

$$\begin{aligned}&2c_k(n)\sum_{i=1}^{I}\Bigg(U^i\big(x^i(n), u_{k+1}^i(n)\big) + \mathbf{E}_{w^i}\Big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\\ &+V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big) + V^{*i}\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\Big\}\Bigg),\end{aligned} \tag{24}$$

on the right hand side of (23), we get that

$$\begin{aligned}&\|\lambda_{k+1}(n) - \lambda(n)\|^2 \leq \|\lambda_k(n) - \lambda(n)\|^2 + \|c_k(n)G_{k+1}(n)\|^2\\ &- 2c_k(n)(\lambda_k(n) - \lambda(n))b(n)\\ &+ 2c_k(n)\sum_{i=1}^{I}\Bigg(U^i\big(x^i(n), u_{k+1}^i(n)\big) + \lambda_k(n)u_{k+1}^i(n)\\ &+ \mathbf{E}_{w^i}\Big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big) + V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\\ &+ V^{*i}\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\Big\}\Bigg)\\ &- 2c_k(n)\sum_{i=1}^{I}\Bigg(U^i\big(x^i(n), u_{k+1}^i(n)\big) + \lambda(n)u_{k+1}^i(n)\\ &+ \mathbf{E}_{w^i}\Big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big) + V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\\ &+ V^{*i}\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\Big\}\Bigg)\\ &= \|\lambda_k(n) - \lambda(n)\|^2 + \|c_k(n)G_{k+1}(n)\|^2 - 2c_k(n)(\lambda_k(n) - \lambda(n))b(n)\\ &+ 2c_k(n)\sum_{i=1}^{I}\Bigg(L^i\big(x^i(n), u_{k+1}^i(n), \lambda_k(n)\big)\\ &+ \mathbf{E}_{w^i}\Big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big) + V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\\ &+ V^{*i}\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\Big\}\Bigg)\\ &- 2c_k(n)\sum_{i=1}^{I}\Bigg(L^i\big(x^i(n+1), u_{k+1}^i(n), \lambda(n)\big)\\ &+ \mathbf{E}_{w^i}\Big\{V_k^i\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big) + V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\\ &+ V^{*i}\big(f^i(x^i(n), u_{k+1}^i(n), w^i(n))\big)\Big\}\Bigg).\end{aligned} \tag{25}$$

Consider now (8). From the optimality of $u^i_{k+1}(n)$, it follows that

$$L^i\big(x^i(n+1), u^i_{k+1}(n), \lambda_k(n)\big) + \mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\} \le L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) + \mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\}. \tag{26}$$

In view of inequality (26), the inequality (25) becomes

$$\begin{aligned}
\|\lambda_{k+1}(n) - \lambda(n)\|^2 \le{}& \|\lambda_k(n) - \lambda(n)\|^2 + \|c_k(n)G_{k+1}(n)\|^2 \\
&- 2c_k(n)(\lambda_k(n) - \lambda(n))b(n) \\
&+ 2c_k(n)\sum_{i=1}^{I}\Big(L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^i(n), w^i(n))) \\
&+ V^{*i}(f^i(x^i(n), u^i(n), w^i(n))) + V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\}\Big) \\
&- 2c_k(n)\sum_{i=1}^{I}\Big(L^i\big(x^i(n), u^i_{k+1}(n), \lambda^n\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big) + V^{*i}(f^i(x^i(n), u^i(n), w^i(n))) \\
&+ V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\}\Big),
\end{aligned} \tag{27}$$

which in turn can be reformulated into

$$\begin{aligned}
&\sum_{i=1}^{I}2c_k(n)\Big(L^i\big(x^i(n), u^i_{k+1}(n), \lambda(n)\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\}\Big) \\
&\le \|\lambda_k(n) - \lambda(n)\|^2 - \|\lambda_{k+1}(n) - \lambda(n)\|^2 + \|c_k(n)G_{k+1}(n)\|^2 \\
&- 2c_k(n)(\lambda_k(n) - \lambda(n))b(n) \\
&+ 2c_k(n)\sum_{i=1}^{I}\Big(L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^{*i}\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\} \\
&+ \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^i(n), w^i(n))) + V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\} \\
&- \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n))) + V^{*i}(f^i(x^i(n), u^i(n), w^i(n)))\Big\}\Big).
\end{aligned} \tag{28}$$

Consider the above equation with $\lambda(n) = \lambda^*(n)$ and $u^i(n) = u^{*i}(n)$, $i \in \mathcal{I}$. Recalling the definition of $\tilde{V}\big(x^i(n), u^i(n), \lambda(n)\big)$, we have

$$\begin{aligned}
\sum_{i=1}^{I}2c_k(n)\Big(\tilde{V}\big(x^i(n), u^i_{k+1}(n), \lambda^*(n)\big)\Big) \le{}& \|\lambda_k(n) - \lambda^*(n)\|^2 \\
&- \|\lambda_{k+1}(n) - \lambda^*(n)\|^2 + \|c_k(n)G_{k+1}(n)\|^2 \\
&- 2c_k(n)(\lambda_k(n) - \lambda(n))b(n) + 2c_k(n)\sum_{i=1}^{I}\Big(\tilde{V}\big(x^i(n), u^{*i}(n), \lambda_k(n)\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^{*i}(n), w^i(n))) \\
&+ V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\} - \mathbf{E}_{w^i}\Big\{V^i_k \\
&\times (f^i(x^i(n), u^i_{k+1}(n), w^i(n))) + V^{*i}(f^i(x^i(n), u^{*i}(n), w^i(n)))\Big\}\Big).
\end{aligned} \tag{29}$$

Consider now (11), we obtain

$$\sum_{i=1}^{I}\tilde{V}\big(x^i(n), \tilde{u}^i_{k+1}(n), \lambda^*(n)\big) \le \frac{\sum_{r=0}^{k}\sum_{i=1}^{I}c_r(n)\tilde{V}\big(x^i(n), u^i_{r+1}(n), \lambda^*(n)\big)}{\sum_{r=0}^{k}c_r(n)}. \tag{30}$$

Substituting (29) into (30), we have

$$\begin{aligned}
&\sum_{i=1}^{I}2\tilde{V}\big(x^i(n), \tilde{u}^i_{k+1}(n), \lambda^*(n)\big) \\
&\le \frac{1}{\sum_{r=0}^{k}c_r(n)}\sum_{r=0}^{k}\Big(\|\lambda_r(n) - \lambda^*(n)\|^2 - \|\lambda_{r+1}(n) - \lambda^*(n)\|^2 \\
&+ \|c_r(n)G_{r+1}(n)\|^2 - 2c_r(n)(\lambda_r(n) - \lambda^*(n))b(n)\Big) \\
&+ \frac{2}{\sum_{r=0}^{k}c_r(n)}\sum_{r=0}^{k}\sum_{i=1}^{I}c_r(n)\Big(\tilde{V}\big(x^i(n), u^{*i}(n), \lambda_r(n)\big) \\
&+ \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^{*i}(n), w^i(n))) + V^{*i}\big(f^i(x^i(n), u^i_{k+1}(n), w^i(n))\big)\Big\} \\
&- \mathbf{E}_{w^i}\Big\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n))) \\
&+ V^{*i}(f^i(x^i(n), u^{*i}(n), w^i(n)))\Big\}\Big), n = 1, \dots, N-1,
\end{aligned} \tag{31}$$

where

$$\sum_{r=0}^{k}\big(\|\lambda_r(n) - \lambda^*(n)\|^2 - \|\lambda_{r+1}(n) - \lambda^*(n)\|^2\big) = \|\lambda_0(n) - \lambda^*(n)\|^2 - \|\lambda_{k+1}(n) - \lambda^*(n)\|^2. \tag{32}$$

Assumption 4 implies that $\sum_{r=0}^{\infty}c_r(n) = \infty$. Since $\{\lambda_k(n)\}_{k\ge0}$, $n = 1, \dots, N-1$, is a bounded sequence, so $\|\lambda_0(n) - \lambda^*(n)\|^2 - \|\lambda_{k+1}(n) - \lambda^*(n)\|^2$ is finite. Taking the $\lim_{k\to\infty}$ in (32) leads to

$$\lim_{k\to\infty}\frac{1}{\sum_{r=0}^{k}c_r(n)}\big(\|\lambda_0(n) - \lambda^*(n)\|^2 - \|\lambda_{k+1}(n) - \lambda^*(n)\|^2\big) = 0. \tag{33}$$

Under Assumption 2, we can infer that $\|G_{k+1}(n)\| \leq G$, $n = 1, \ldots, N$, thus using Assumption 4 we have

$$\lim_{k\to\infty} \frac{1}{\sum_{r=0}^{k} c_r(n)} \sum_{r=0}^{k} \|c_r(n) G_{k+1}(n)\|^2 = 0. \tag{34}$$

The convergence rate of $(\lambda_k(n) - \lambda^*(n))$ is in the order $\frac{1}{\sqrt{k}}$, [37]. Thus, by choosing $c_k(n) = \frac{1}{k}$, we have

$$\sum_{k=0}^{\infty} c_k(n)(\lambda^*(n) - \lambda_k(n)) < \infty, \tag{35}$$

Hence, using Assumption 4 we have

$$\lim_{k\to\infty} \frac{1}{\sum_{r=0}^{k} c_r(n)} \sum_{r=0}^{k} c_r(n)(\lambda^*(n) - \lambda_r(n)) = 0. \tag{36}$$

According to Lemmas 2 and 3, and under Assumption 4, we can infer that

$$\begin{aligned} &\lim_{k\to\infty} \frac{1}{\sum_{r=0}^{k} c_r(n)} \sum_{r=0}^{k} \sum_{i=1}^{I} c_r(n) \Bigg(\mathbf{E}_{w^i} \Big\{ V_r^i(f^i(x^i(n), u^{*i}(n), w^i(n))) \\ &- V^{*i}(f^i(x^i(n), u^{*i}(n), w^i(n))) \\ &+ V^{*i}\big(f^i(x^i(n), u_{r+1}^i(n), w^i(n))\big) - V_r^i(f^i(x^i(n), u_{r+1}^i(n), w^i(n))) \Big\} \Bigg) \\ &= 0. \end{aligned} \tag{37}$$

Based on the above discussion, we can get

$$\begin{aligned} &\lim_{k\to\infty} \sum_{i=1}^{I} \tilde{V}\big(x^i(n), \tilde{u}_{k+1}^i(n), \lambda^*(n)\big) \\ &\leq \frac{1}{\sum_{r=0}^{k} c_r(n)} \sum_{r=0}^{k} \sum_{i=1}^{I} c_r(n) \tilde{V}\big(x^i(n), u^{*i}(n), \lambda_r(n)\big) \\ &\leq \frac{1}{\sum_{r=0}^{k} c_r(n)} \sum_{r=0}^{k} \sum_{i=1}^{I} c_r(n) \tilde{V}\big(x^i(n), u^{*i}(n), \lambda^*(n)\big) \\ &\leq \sum_{i=1}^{I} \tilde{V}\big(x^i(n), u^{*i}(n), \lambda^*(n)\big), \end{aligned} \tag{38}$$

where the first inequality follows from (31), (33), (34), (36), and (37). The second inequality is obtained by using Lemma 1 (saddle point theory).

Moreover, by using Lemma (39), we have

$$\lim_{k\to\infty} \sum_{i=1}^{I} \tilde{V}\big(x^i(n), \tilde{u}_{k+1}^i(n), \lambda^*(n)\big) \geq \sum_{i=1}^{I} \tilde{V}\big(x^i(n), u^{*i}(n), \lambda^*(n)\big), \tag{39}$$

Hence, we can deduce from (38) and (39) that

$$\lim_{k\to\infty} \sum_{i=1}^{I} \tilde{V}\big(x^i(n), \tilde{u}_{k+1}^i(n), \lambda^*(n)\big) = \sum_{i=1}^{I} \tilde{V}\big(x^i(n), u^{*i}(n), \lambda^*(n)\big). \tag{40}$$

The above equation holds for $n = 1, \ldots, N-1$. Moreover, by virtue of the fact that $\tilde{V}(\cdot, \cdot, \lambda^*(n))$, $i \in \mathcal{I}$ and $n = 1, \ldots, N-1$, is continuous and strongly convex, we can infer that all limit points of $\{\tilde{u}_{k+1}^i(n)\}_{k\geq 0}$, converge to $u^{*i}(n)$, $i \in \mathcal{I}$ and $n = 1, \ldots, N-1$. Hence the claim of Theorem 3 follows. □

6. Implementation of the proposed algorithm using neural network

In the above sections, we proved that the updated cost-to-go function (9) and control input (8) converge to their optimal value. We assumed that (9) and (8) can be solved exactly. However, it is hard to find the exact solutions for these equations for the general cost function. Hence, inspired by [38], we can approximate these equations using neural networks. In what follows, we explain the structure and learning process of the neural network.

6.1. Critic and action neural network

A critic and action neural network are used to approximate the cost-to-go function (9) and the control input (8), respectively. The control input is approximated as

$$\hat{u}_k^i(n) = {\nu_{a,k}^i}^{\top} \sigma(x^i(n)), \tag{41}$$

where $\nu_{a,k}^i$ is the weight matrix of the action network of agent i at iteration k. $\sigma : \mathbb{R}^p \to \mathbb{R}_+$ is the general function approximation.

The cost-to-go function is approximated at each iteration as

$$\hat{V}_k^i(x^i(n)) = {\nu_{c,k}^i}^{\top} \phi(x^i(n)), \tag{42}$$

where $\nu_{c,k}^{iT}$ is the weight matrix of the critic network of agent i at iteration k. $\phi(\cdot)$ is the radial basis function.

In order to satisfy the Bellman equation in (7), the loss function of the critic network is

$$\begin{aligned} E_{c,k+1}^i(n) = \frac{1}{2} \Bigg(&(1 - c_k(n)) \hat{V}_k^i(x^i(n)) + \big(c_k(n)\big)^2 \\ &\times \Big(L^i\big(x^i(n), \hat{u}_{k+1}^i(n), \lambda_k(n)\big) \\ &+ \mathbf{E}_{w^i} \big\{ \hat{V}_k^i\big(f^i(x^i(n), \hat{u}_{k+1}^i(n), w^i(n))\big) \big\} \Big) - \hat{V}_{k+1}^i(x^i(n)) \Bigg)^2. \end{aligned} \tag{43}$$

The weights of the critic are updated using the gradient-based algorithm, such that the minimum loss function (43) is attained. Hence, we have

$$\nu_{c,k+1}^i(j+1) = \nu_{c,k+1}^i(j) - \beta_c \frac{\partial E_{c,k+1}^i(n)}{\partial \hat{V}_{k+1}^i(x^i(n))} \frac{\partial \hat{V}_{k+1}^i(x^i(n))}{\partial \nu_{c,k+1}^i(j)}, \tag{44}$$

where j is the iteration step for updating the neural network's weights. β_c is the learning rate parameter.

Based on (8) the lost function of the action network is given by

$$\begin{aligned} E_{a,k+1}^i(n) = \frac{1}{2} \Bigg(&\arg \min_{u^i(n) \in \mathcal{U}} \Big[L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) \\ &+ \mathbf{E}_{w^i} \Big\{ \hat{V}_k^i\big(f^i(x^i(n), u^i(n), w^i(n))\big) \Big\} \Big] - \hat{u}_{k+1}^i(n) \Bigg)^2, \end{aligned} \tag{45}$$

The weights of the action network are updated as

$$\nu_{a,k+1}^i(j+1) = \nu_{a,k+1}^i(j) - \beta_a \frac{\partial E_{a,k+1}^i(n)}{\partial \nu_{a,k+1}^i(j)}, \tag{46}$$

where β_a is the learning rate parameter.

Now, we present the implementation of the proposed iterative method in Algorithm 2 in which, the solution of Eqs. (9) and (8) in Steps 3 and 4 of Algorithm 1 are obtained using neural networks from Steps 3–9 of Algorithm 2.

Theorem 4. *Consider (41), and (42). Under update rules in (44), and (46), the weights of actor and critic neural networks converge to their optimal values at each iteration k of Algorithm 2, if the learning rates in (44) and (46) are chosen as follows:*

The estimation error of ideal actor and critic neural network weight at iteration k of Algorithm 2, which is denoted by $\hat{\nu}^i_{a,k}$ and $\hat{\nu}^i_{c,k}$, $i \in \mathcal{I}$, respectively, is zero, if we choose the learning rate of (44) and (46) as follows:

$$\beta_c \leq \frac{2}{\|\phi(x^i(n))\|^2}, \quad \beta_a \leq \frac{2}{\|\sigma(x^i(n))\|^2}. \tag{47}$$

Proof. Let the target control input and the cost-to-go function be obtained as follows:

$$\begin{aligned} u^i_k(n) &= \sum_{l=1}^{L_a} \nu^{*i}_{a,k,l}\sigma(x^i(n)) = {\nu^{*i}_{a,k}}^\top \sigma(x^i(n)), \\ V^i_k(x^i(n)) &= \sum_{l=1}^{L_c} \nu^{*i}_{c,k,l}\phi(x^i(n)) = {\nu^{*i}_{c,k}}^\top \phi(x^i(n)), \end{aligned} \tag{48}$$

where L_a and L_c are the neurons number of the neural networks which are sufficiently large enough. $\nu^{*i}_{a,k}$ and $\nu^{*i}_{c,k}$ are the optimal weights of actor and critic networks, respectively.

Let us define the error weights of critic and actor networks as follows:

$$\begin{aligned} \hat{\nu}^i_{c,k}(j) &= \nu^i_{c,k}(j) - \nu^{*i}_{c,k}, \\ \hat{\nu}^i_{a,k}(j) &= \nu^i_{a,k}(j) - \nu^{*i}_{a,k}. \end{aligned} \tag{49}$$

Consider the following Lyapunov function:

$$F(\hat{\nu}^i_{c,k}(j), \hat{\nu}^i_{a,k}(j)) = \mathrm{tr}\{\hat{\nu}^i_{c,k}(j)^T \hat{\nu}^i_{c,k}(j) + \hat{\nu}^i_{a,k}(j)^T \hat{\nu}^i_{a,k}(j)\}. \tag{50}$$

Then, using (46) and (44), the difference of (50) is obtained as follows:

$$\begin{aligned} &\Delta F(\hat{\nu}^i_{c,k}(j), \hat{\nu}^i_{a,k}(j)) \\ &= \mathrm{tr}\{\hat{\nu}^i_{c,k}(j+1)^T \hat{\nu}^i_{c,k}(j+1) + \hat{\nu}^i_{a,k}(j+1)^T \hat{\nu}^i_{a,k}(j+1)\} \\ &\quad - \mathrm{tr}\{\hat{\nu}^i_{c,k}(j)^T \hat{\nu}^i_{c,k}(j) + \hat{\nu}^i_{a,k}(j)^T \hat{\nu}^i_{a,k}(j)\} \\ &= \beta_c \|E^i_{c,k}(n)\|^2(-2 + \beta_c\|\phi(x^i(n))\|^2) + \beta_a\|E^i_{a,k}(n)\|^2 \\ &\quad \times (-2 + \beta_a\|\sigma(x^i(n))\|^2), \end{aligned} \tag{51}$$

where the second equality is obtained by the following formulations:

$$\begin{aligned} \hat{\nu}^i_{a,k}(j+1) &= \hat{\nu}^i_{a,k}(j) - \beta_a E^i_{a,k}(n)\sigma(x^i(n)), \\ \hat{\nu}^i_{c,k}(j+1) &= \hat{\nu}^i_{c,k}(j) - \beta_c E^i_{c,k}(n)\phi(x^i(n)). \end{aligned} \tag{52}$$

Hence, by choosing $\beta_c \leq \frac{2}{\|\phi(x^i(n))\|^2}$ and $\beta_a \leq \frac{2}{\|\sigma(x^i(n))\|^2}$, the difference of Lyapunov candidate (52) is negative and the weights of actor and critic neural networks converge to their optimal values. □

7. Simulation results

Resource allocation to the human operators is one of the potential applications of our proposed algorithm [39] which can be modeled as a multi agent stochastic dynamical system. The human operators are considered as the agents, and the state of system is considered as the workload of them. The resources are considered as the input to the system in which, the workload of agents decreases with the allocated resources to them. The objective function of the system consists of the terms associated with the cost of workload and the cost of resources allocated to the human operators which is modeled as follows

$$\begin{cases} \min_u \quad \mathbf{E}_w \left\{ \sum_{i=1}^{I} \Big(x^i(N)^\top x^i(N) + \sum_{m=1}^{N-1} \big(x^i(m)^\top Q_i x^i(m) \\ \qquad + e^{(u^i(m)^\top R_i u^i(m))} \big) \Big) \right\}, \\ \text{s.t.} \qquad \mathrm{C}_1: \ \sum_{i=1}^{I} u^i(n) \leq b(n), \quad n = 1, \ldots, N-1, \\ \qquad\quad \mathrm{C}_2: \ u^i(n) \geq 0, i \in \mathcal{I}, n = 1, \ldots, N-1, \end{cases} \tag{53}$$

The control input $u^i(m)$ of agent i is the resource allocated to the operators at time-step m. $b(n)$ is the maximum available resource that can be allocated to the agents. Q_i and R_i are positive definite matrices which depend on the agents' model.

We consider 10 agents and the dynamic system and cost function parameters of the agents are shown in Table 1. The initial value of the dynamic states is $x^1(0) = 10$, $x^2(0) = 50$, $x^3(0) = 40$, $x^4(0) = 20$, $x^5(0) = 20$, $x^6(0) = 30$, $x^7(0) = 10$, $x^8(0) = 40$, $x^9(0) = 50$, and $x^{10}(0) = 20$.

Since, in 7, we face a non-quadratic cost function, the Bellman's equation does not have a closed-form solution and

Algorithm 2 Implementation of ADP and dual decomposition method

Initialize: N, $x^i(0)$, λ_0, β_c, β_a, $\phi(\cdot)$, ϵ, $k = 1$, $i \in \mathcal{I}$.
procedure for $i \in \mathcal{I}$ **and** $n = 1, \ldots, N$
  Initialize: $j = 0$, $\nu^i_{c,k}(0)$, $\nu^i_{a,k}(0)$, $i \in \mathcal{I}$
  while $\|\nu^i_{c,k}(j+1) - \nu^i_{c,k}(j)\| > \epsilon$ or $\|\nu^i_{a,k}(j+1) - \nu^i_{a,k}(j)\| > \epsilon$ for $i \in \mathcal{I}$, **do**
    Update the weights of critic and actor neural networks using (44) and (46), respectively.
    $\nu^i_{c,k}(j) \leftarrow \nu^i_{c,k}(j+1)$, $\nu^i_{a,k}(j) \leftarrow \nu^i_{a,k}(j+1)$.
    $j = j + 1$.
  Compute the control input using (41).
  Compute the cost-to-go function using (42).
  Apply the control input to the system and obtain the next state.
  Update Lagrange multiplier using (10).
  Compute $\tilde{u}^i_k(n)$ using (11).
  $k = k + 1$.

Table 1
Agent's model parameters.

	Agent									
	1	2	3	4	5	6	7	8	9	10
Q	50	30	20	10	10	20	30	40	30	50
R	45	27	18	13.5	13.5	22.5	27	36	18	45
A	0.8	0.78	0.85	0.8	0.8	0.65	0.75	0.9	0.8	0.95
B	−0.02	−0.02	−0.015	−0.01	−0.01	−0.02	−0.017	−0.02	−0.01	−0.02

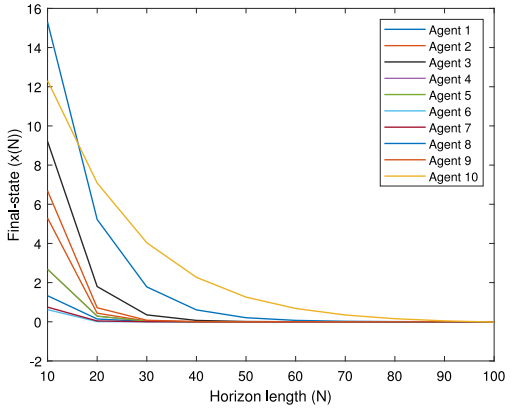

Fig. 2. Final-state of all the agents with respect to horizon length N.

hence, we apply our proposed algorithm to this model. In order to implement the ADP method, we consider two RBF neural networks with one hidden layer for each agent which estimate the cost-to-go and control inputs. The initial value for the weights of neural network are chosen randomly between 1 and 3. In what follows, we show our simulation results.

In order to choose an appropriate value for the number of horizons "N", we implemented our proposed algorithm for different values of N. Then, we chose the value of N so that $x^i(N) < \epsilon, i \in \mathcal{I}$. Fig. 2 shows the final-state of all the agents for different horizon length N. By setting $\epsilon = 0.001$, we can see that for the horizon length larger than 90, the final-state of all the agents reach to ϵ neighborhood of the origin. Hence, we choose the horizon length larger than 90 and implement our proposed algorithm with $N = 100$.

Fig. 3 compares the computation time of different methods. The computation time is recorded for different optimality levels of solution. The results provided by MATLAB R2019a and are performed on a Desktop PC with 8 GB of RAM and a 2.40 GHz processor. As it was expected, the computation time of the centralized approach is more than that of the decentralized methods. In addition, we can see that the DMPC requires more computation time than decentralized ADP to reach the solutions with different levels of optimality. In any case, decentralized DP has more computation time than two other decentralized methods. It should be emphasized that since MPC solves the optimization problem by considering the receding horizon-length and then applies only the first-step control input to the system and go one-step ahead until the end of the horizon length (N), hence the computation time of MPC depends on the selection of receding-horizon length. In this simulation, by making a trad-off between optimality gap, computation time, and accuracy of the model prediction, the value of receding-horizon length is chosen equal to 5. However, in decomposition method we need many data communication between the agents and central system to compute the optimal solution, whereas in centralized method, no communication is required for optimization process. Nevertheless, in centralized method, we assume that the central system is already informed about the utility function and the local constraints of the agents and hence, if the structure of those functions are not predetermined, it is not straightforward to send such information [40].

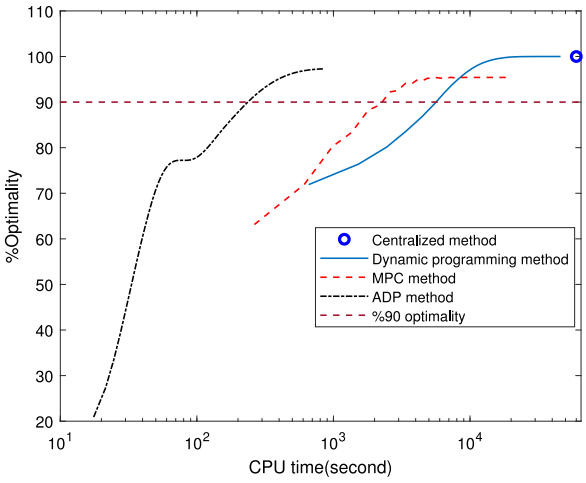

Fig. 3. Computation time of centralized, DP, MPC, ADP methods.

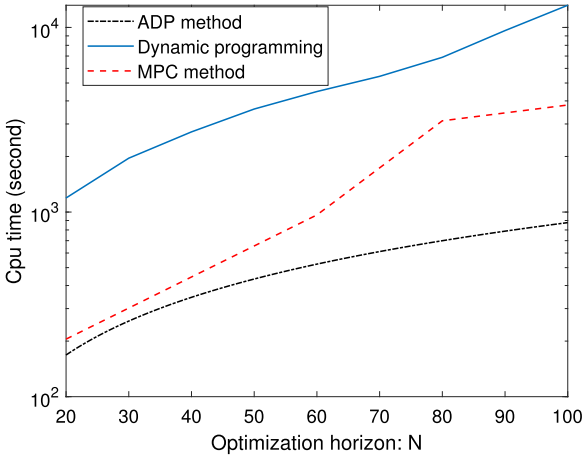

Fig. 4. The comparison of computation time of decentralized DP, MPC, and ADP concerning the optimization horizon N.

Fig. 4 compares the computation time of DP, MPC and ADP concerning the optimization horizon N. As we can see the difference between the computation time of decentralized ADP and MPC increases for larger values of the horizon length N. Moreover, as we expected, the computation time of the centralized approach is more than that of the decentralized methods. In addition, we can see that the DMPC requires more computation time than decentralized ADP. In any case, decentralized DP has more computation time than two other decentralized methods. Fig. 5 shows the convergence of the agents' expected cost functions (total cost of all the agents) for decentralized ADP, DP and MPC methods. As we can see, DP and MPC converge in a lesser number of iterations than ADP. Specifically, DMPC reaches to a solution with %90 of optimality in about 15 iterations, but decentralized ADP reaches to the same level of optimality in about 150 iterations. However, since each iteration of DP and MPC takes more CPU time than the ADP method, based on Fig. 4, the computation time of decentralized ADP is lesser than two

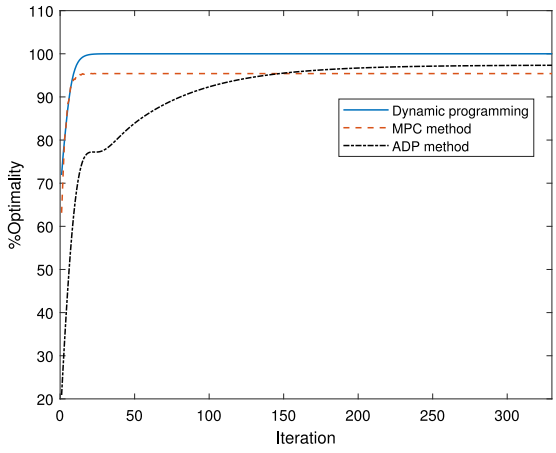

Fig. 5. The convergence of the expected cost functions of DP, MPC, and ADP method.

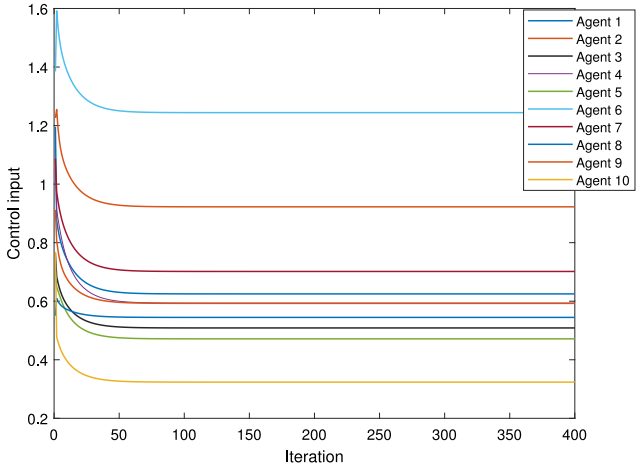

Fig. 6. The convergence of the control input at $N = 20$ using adaptive dynamic programming method.

other methods. Moreover, according to Fig. 5, the optimality gap of decentralized ADP is lesser than DMPC.

The convergence of the optimal solution (calculates using (11)) at $N = 20$ for all agents is shown in Fig. 6. According to this figure the average control inputs converge to their optimal value, so that at the convergence points the coupling constraint is satisfied. In the other words $u^1(20) + u^2(20) + u^3(20) + u^4(20) + u^5(20) + u^6(20) + u^7(20) + u^8(20) + u^9(20) + u^{10}(20) < 10.41$, where 10.41 is the coupling constraint at $N = 20$. The convergence of the expected cost function is shown in Fig. 7. This figure shows that the agents' expected cost function converges and according to Fig. 3 we can deduce that they converge to their near optimal values.

8. Conclusion

In this paper, we proposed a novel decomposition algorithm which works based on the duality theory and ADP method to solve the multi-agent optimal control problem with general strictly convex cost function and coupling constraints among the agents. In our proposed algorithm, the central system does not need to know the agents' dynamics and cost functions. As a result, the privacy of agents is preserved. Moreover, convergence of the proposed algorithm to global optimal solution was proven.

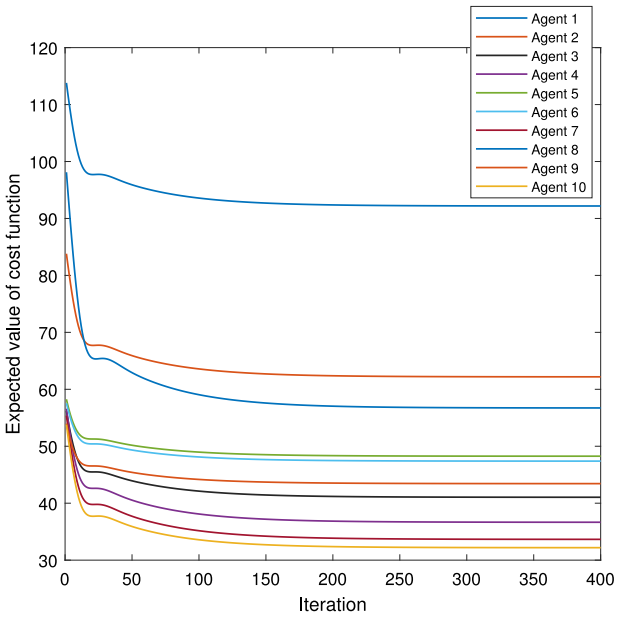

Fig. 7. The convergence of the expected cost functions.

Our simulation results demonstrated that the proposed algorithm converges to the optimal value with lower computational cost in comparison with the dynamic programming and distributed MPC methods.

As a future research, a nonlinear dynamic can be considered for each agent which makes the problem non-convex. It would be challenging to solve the problem in a decomposition way, and to compute the duality gap. Furthermore, one can consider that there is not any central system and the agents communicate with each other through the graph.

CRediT authorship contribution statement

Pegah Rokhforoz: Conceptualization, Writing - original draft, Methodology, Programming and simulating the results. **Hamed Kebriaei:** Conceptualization, Writing - original draft, Methodology, Supervision. **Majid Nili Ahmadabadi:** Conceptualization, Methodology, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the Institute for Research in Fundamental Sciences under Grant CS 1398-4-256.

Appendix

Proof of Lemma 1. As mentioned above the strong duality holds. Hence, an optimal primal–dual pair (u^*, λ^*) exists and saddle-point theorem holds (Proposition 5.1.6 [37]). □

Proof of Lemma 2. Let $\eta^i(n)$ be any arbitrary admissible control input which stabilizes (1), $i \in \mathcal{I}$, $n = 1, \ldots, N-1$. Let us define a new sequence $\Lambda^i_k(x^i(n))$ as

$$\begin{aligned}\Lambda^i_{k+1}(x^i(n)) &= (1-c_k(n))\Lambda^i_k(x^i(n)) + \big(c_k(n)\big)^2 \\ &\times \left[L^i\big(x^i(n), \eta^i(n), \lambda_k(n)\big) + \mathbf{E}_{w^i}\Big\{\Lambda^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big)\Big\}\right], \\ \Lambda^i_{k+1}(x^i(N)) &= (1-c_k(N))\Lambda^i_k(x^i(N)) + \big(c_k(N)\big)^2 U^i_N\big(x^i(N)\big).\end{aligned} \tag{54}$$

Let us assume that $\Lambda^i_0(\cdot) = 0$, $i \in \mathcal{I}$. Thus we have

$$\begin{aligned}\Lambda^i_{k+1}(x^i(n)) &= \sum_{j=0}^{k}\Big(\prod_{O=1}^{j}\big(1-c_{k-O+1}(n)\big)\Big)\big(c_{k-j}(n)\big)^2 \\ &\times L^i\big(x^i(n), \eta^i(n), \lambda_{k-j}(n)\big) \\ &+ \big(c_k(n)\big)^2\mathbf{E}_{w^i}\Big\{\Lambda^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big)\Big\}, \\ &n = 1, \ldots, N-1, \\ \Lambda^i_{k+1}(x^i(N)) &= \sum_{j=0}^{k}\Big(\prod_{O=1}^{j}\big(1-c_{k-O+1}(N)\big)\Big)\big(c_{k-j}(N)\big)^2 U^i_N\big(x^i(N)\big),\end{aligned} \tag{55}$$

where

$$\begin{aligned}&\Lambda^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big) \\ &= \sum_{j=0}^{k} f(c_j(n+j))\mathbf{E}_{w^i}\big\{L^i(x^i(n+j), \eta^i(n+j), \lambda_{k-j}(n+j))\big\}, \\ &n = 1, \ldots, N-1.\end{aligned} \tag{56}$$

Since, $0 < c_k(n) \le 1$, so we have

$$\begin{aligned}&\sum_{j=0}^{k}\Big(\prod_{O=1}^{j}\big(1-c_{k-O+1}(n)\big)\Big)\big(c_{k-j}(n)\big)^2 L^i\big(x^i(n), \eta^i(n), \lambda_{k-j}(n)\big) \\ &\le \sum_{j=0}^{k}\big(c_{k-j}(n)\big)^2 L^i\big(x^i(n), \eta^i(n), \lambda_{k-j}(n)\big) < \infty, \\ &\sum_{j=0}^{k}\Big(\prod_{O=1}^{j}\big(1-c_{k-O+1}(N)\big)\Big)\big(c_{k-j}(N)\big)^2 U^i_N\big(x^i(N)\big) \\ &\le \sum_{j=0}^{k}\big(c_{k-j}(N)\big)^2 U^i_N\big(x^i(N)\big) < \infty,\end{aligned} \tag{57}$$

where the second inequality follows from Assumption 4 along with the fact that $\{\lambda_k(n)\}_{k\ge 0}$ and $\{L^i(x^i(n), \eta^i(n), \lambda_k(n))\}_{k\ge 0}$ are a bounded sequence (Theorem 2, Assumptions 2 and 3) and $U^i_N\big(x^i(N)\big)$ has a finite value.

Moreover, since $\eta^i(n)$ is an admissible stabilizing controller, using (56), we can deduce that

$$\Lambda^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big) < \infty, n = 1, \ldots, N-1. \tag{58}$$

Hence (57) and (58) lead to

$$\Lambda^i_{k+1}(x^i(n)) \le Y < \infty, n = 1, \ldots, N. \tag{59}$$

Now, we prove by the mathematical induction that if $V^i_0(\cdot) = \Lambda^i_0(\cdot) = 0$, then we have $V^i_{k+1}(x^i(n)) \le \Lambda^i_{k+1}(x^i(n))$. Since $\eta^i(n)$ is any arbitrary admissible control law, we can deduce that

$$\begin{aligned}V^i_1(x^i(n)) &= \big(c_k(n)\big)^2 \min_{u^i(n)} L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) \\ &\le \big(c_k(n)\big)^2 L^i\big(x^i(n), \eta^i(n), \lambda_k(n)\big) = \Lambda^i_1(x^i(n)).\end{aligned} \tag{60}$$

Suppose k is the instant at which $V^i_k(x^i(n)) \le \Lambda^i_k(x^i(n))$, $n = 1, \ldots, N-1$. Then, we have to show that at instant $k+1$, $V^i_{k+1}(x^i(n)) \le \Lambda^i_{k+1}(x^i(n))$. Eq. (9) leads to

$$\begin{aligned}V^i_{k+1}(x^i(n)) &= (1-c_k(n))V^i_k(x^i(n)) \\ &+ \big(c_k(n)\big)^2 \min_{u^i(n)}\Big[L^i\big(x^i(n), u^i(n), \lambda_k(n)\big) \\ &+ \mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\}\Big] \\ &\le (1-c_k(n))V^i_k(x^i(n)) + \big(c_k(n)\big)^2\Big[L^i\big(x^i(n), \eta^i(n), \lambda_k(n)\big) \\ &+ \mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big)\Big\}\Big] \\ &\le (1-c_k(n))\Lambda^i_k(x^i(n)) + \big(c_k(n)\big)^2\Big[L^i\big(x^i(n), \eta^i(n), \lambda_k(n)\big) \\ &+ \mathbf{E}_{w^i}\Big\{\Lambda^i_k\big(f^i(x^i(n), \eta^i(n), w^i(n))\big)\Big\}\Big] \\ &= \Lambda^i_{k+1}(x^i(n)),\end{aligned} \tag{61}$$

where the second inequality follows from the assumption at instant k, and the equality is obtained by (54).

Similarly, for $n = N$, we can use the mathematical induction. Suppose at instant k we have $V^i_k(x^i(N)) \le \Lambda^i_k(x^i(N))$. Then at instant $k+1$ we have

$$\begin{aligned}V^i_{k+1}(x^i(N)) &= \big(1-c_k(N)\big)V^i_k(x^i(N)) + \big(c_k(N)\big)^2 U^i_N\big(x^i(N)\big) \\ &\le \big(1-c_k(N)\big)\Lambda^i_k(x^i(N)) + \big(c_k(N)\big)^2 U^i_N\big(x^i(N)\big) \\ &= \Lambda^i_{k+1}(x^i(N)).\end{aligned} \tag{62}$$

In view of (58), (61), and (62), we can deduce that

$$V^i_{k+1}(x^i(n)) \le \Lambda^i_{k+1}(x^i(n)) \le Y, n = 1, \ldots, N. \quad \square \tag{63}$$

Proof of Lemma 3. Let us define $\mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\}$ as follows

$$\begin{aligned}&\mathbf{E}_{w^i}\Big\{V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big)\Big\} \\ &= \sum_{x^i(n+1)\in\mathcal{X}^i(n+1)} p_{k+1}\big(f^i(x^i(n), u^i(n), w^i(n))\big) \\ &\times V^i_k\big(f^i(x^i(n), u^i(n), w^i(n))\big) \\ &\le Y \sum_{x^i(n+1)\in\mathcal{X}^i(n+1)} p_{k+1}\big(f^i(x^i(n), u^i(n), w^i(n))\big) \le Y,\end{aligned} \tag{64}$$

where $\mathcal{X}^i(n)$, $i \in \mathcal{I}$, $n = 1, \ldots, N$, is the set of all possible states. The first inequality is obtained by Lemma 2. The second inequality follows by the fact that $\sum_{x^i(n+1)\in\mathcal{X}^i(n+1)} p_{k+1}\big(f^i(x^i(n), u^i(n), w^i(n))\big)$ 1 (using the features of transition probabilities). $\square$

Proof of Lemma 4. Consider (9). Adding and subtracting $V^{*i}(x^i(n))$ on the right hand side of (9) and squaring both sides

of the equation, we get

$$\begin{aligned}
&\left(V^i_{k+1}(x^i(n)) - V^{*i}(x^i(n))\right)^2 = \left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right)^2 \\
&+ \left(c_k(n)\right)^2\left(V^i_k(x^i(n))\right)^2 \\
&- 2c_k(n)V^i_k(x^i(n))\left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right) \\
&+ \left(c_k(n)\right)^4(L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right) \\
&+ \mathbf{E}_{w^i}\left\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\right\})^2 \\
&+ 2(c_k(n))^2(L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right) \\
&+ \mathbf{E}_{w^i}\left\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\right\})(V^i_k(x^i(n)) - V^{*i}(x^i(n))) \\
&- 2(c_k(n))^3(L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right) \\
&+ \mathbf{E}_{w^i}\left\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\right\})(V^i_k(x^i(n)) - V^{*i}(x^i(n))).
\end{aligned} \tag{65}$$

In view of (10) and under Assumptions 2 and 3, we can deduce that $\{L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right)\}^\infty_{k=0} \geq 0$ and $\{V^i_k(x^i(n))\}^\infty_{k=0} \geq 0$, $n = 1, \ldots, N-1$. Then, after some simplification and neglecting some negative terms we obtain

$$\begin{aligned}
&2c_k(n)V^i_k(x^i(n))\left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right) \\
&\leq \left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right)^2 - \left(V^i_{k+1}(x^i(n)) - V^{*i}(x^i(n))\right)^2 \\
&+ \left(c_k(n)\right)^2\left(V^i_k(x^i(n))\right)^2 + \left(c_k(n)\right)^4(L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right) \\
&+ \mathbf{E}_{w^i}\left\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\right\})^2 \\
&+ 2(c_k(n))^2(L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right) \\
&+ \mathbf{E}_{w^i}\left\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\right\})(V^i_k(x^i(n)) - V^{*i}(x^i(n))),
\end{aligned} \tag{66}$$

where the above inequality is obtained by the fact that $V^{*i}(x^i(n))$ is the optimal cost-to-go function and $V^i_k(x^i(n)) - V^{*i}(x^i(n)) \geq 0$.

Summing (66) over k by the fact that $\sum^\infty_{k=0}\left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right)^2 - \left(V^i_{k+1}(x^i(n)) - V^{*i}(x^i(n))\right)^2 = \left(V^i_0(x^i(n)) - V^{*i}(x^i(n))\right)^2 - \left(V^i_\infty(x^i(n)) - V^{*i}(x^i(n))\right)^2$, and by using Lemmas 2 and 3, and under Assumptions 2 and 3, $\{V^i_k(\cdot)\}^\infty_{k=0} < \infty$, $\{L^i\left(x^i(n), u^i_{k+1}(n), \lambda_k(n)\right)\}^\infty_{k=0} < \infty$, $\{E_{w^i}\{V^i_k(f^i(x^i(n), u^i_{k+1}(n), w^i(n)))\}^\infty_{k=0}\} < \infty$, and also under Assumption 4, we have

$$\sum^\infty_{k=0} c_k(n)\left(V^i_k(x^i(n)) - V^{*i}(x^i(n))\right) < \infty, n = 1, \ldots, N-1. \tag{67}$$

Similarly, for $n = N$, we have

$$\begin{aligned}
&2c_k(n)V^i_k(x^i(N))\left(V^i_k(x^i(N)) - V^{*i}(x^i(N))\right) \\
&\leq \left(V^i_k(x^i(N)) - V^{*i}(x^i(N))\right)^2 - \left(V^i_{k+1}(x^i(N)) - V^{*i}(x^i(N))\right)^2 \\
&+ \left(c_k(N)\right)^2\left(V^i_k(x^i(N))\right)^2 + \left(c_k(N)\right)^4(U^i_N(x^i(N)))^2 \\
&+ 2\left(c_k(N)\right)^2U^i_N(x^i(N))\left(V^i_k(x^i(N)) - V^{*i}(x^i(N))\right).
\end{aligned} \tag{68}$$

Using the same reasons of $n = 1, \ldots, N-1$, we can deduce that

$$\sum^\infty_{k=0} c_k(N)\left(V^i_k(x^i(N)) - V^{*i}(x^i(N))\right) < \infty. \quad \square \tag{69}$$

References

[1] D.H. Lee, Resource-based task allocation for multi-robot systems, Robot. Auton. Syst. 103 (2018) 151–161.
[2] B.P. Gerkey, J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, Int. J. Robot. Res. 23 (9) (2004) 939–954.
[3] Y. Zhang, G.B. Giannakis, Distributed stochastic market clearing with high-penetration wind power, IEEE Trans. Power Syst. 31 (2) (2016) 895–906.
[4] V. Srivastava, R. Carli, C. B. Langbort, F. Bullo, Attention allocation for decision making queues, Automatica 50 (2) (2014) 378–388.
[5] Z. Shen, J.G. Andrews, B. Evans, L. Brian, Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints, IEEE Trans. Wirel. Commun. 4 (6) (2005) 2726–2737.
[6] S. Sariel-Talay, T.R. Balch, N. Erdogan, Multiple traveling robot problem: A solution based on dynamic task selection and robust execution, IEEE/ASME Trans. Mechatron. 14 (2) (2009) 198–206.
[7] R. Singh, P. Kumar, L. Xie, Decentralized control via dynamic stochastic prices: The independent system operator problem, IEEE Trans. Automat. Control 63 (10) (2018) 3206–3220.
[8] S. Lozano, G. Villa, Centralized resource allocation using data envelopment analysis, J. Product. Anal. 22 (1–2) (2004) 143–161.
[9] F.L. Lotfi, A. Noora, Centralized resource allocation for enhanced russell models, J. Comput. Appl. Math. 235 (1) (2010) 1–10.
[10] D.P. Bertsekas, Nonlinear programming, J. Oper. Res. Soc. 48 (3) (1997) 334.
[11] D. Palomar, M. Chiang, A tutorial on decomposition methods for network utility maximization, IEEE J. Sel. Areas Commun. 24 (8) (2006) 1439–1451.
[12] G. Cohen, Optimization by decomposition and coordination: A unified approach, IEEE Trans. Automat. Control 23 (2) (1978) 222–232.
[13] A. Falsone, K. Margellos, S. Garatti, M. Prandini, Dual decomposition for multi-agent distributed optimization with coupling constraints, Automatica 84 (2017) 149–158.
[14] M. Zhu, S. Martínez, On distributed convex optimization under inequality and equality constraints, IEEE Trans. Automat. Control 57 (1) (2012) 151–164.
[15] J. Lei, H. Chen, H. Fang, Primal–dual algorithm for distributed constrained optimization, Systems Control Lett. 96 (2016) 110–117.
[16] A. Cherukuri, E. Mallada, J. Cortés, Asymptotic convergence of constrained primal–dual dynamics, Systems Control Lett. 87 (2016) 10–15.
[17] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, Athena scientific Belmont, MA, 2005.
[18] D. Liu, D. Wang, X. Yang, An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs, Inform. Sci. 220 (2013) 331–342.
[19] T. Bian, Y. Jiang, Z. Jiang, Adaptive dynamic programming and optimal control of nonlinear nonaffine systems, Automatica 50 (10) (2014) 2624–2632.
[20] F.Y. Wang, N. Jin, D. Liu, Q. Wei, Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ϵ-error bound, IEEE Trans. Neural Netw. 22 (1) (2011) 24–36.
[21] C. Mu, D. Wang, H. He, Data-driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach, IEEE Trans. Cybern. (2017).
[22] T.L. Nguyen, Adaptive dynamic programming-based design of integrated neural network structure for cooperative control of multiple MIMO nonlinear systems, Neurocomputing 237 (2017) 12–24.
[23] K.G. Vamvoudakis, F.L. Lewis, G.R. Hudas, Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality, Automatica 48 (8) (2012) 1598–1611.
[24] M. Abouheaf, F.L. Lewis, K. Vamvoudakis, S. Haesaert, R. Babuska, Multi-agent discrete-time graphical games and reinforcement learning solutions, Automatica 50 (12) (2014) 3038–3053.
[25] H. Jiang, H. Zhang, X. Xie, J. Han, Neural-network-based learning algorithms for cooperative games of discrete-time multi-player systems with control constraints via adaptive dynamic programming, Neurocomputing 344 (2019) 13–19.
[26] R. Negenborn, J. Maestre, Distributed model predictive control: An overview and roadmap of future research opportunities, IEEE Control Syst. Mag. 34 (4) (2014) 87–97.
[27] E. Camponogara, D. Jia, B. Krogh, S. Talukdar, Distributed model predictive control, IEEE Control Syst. Mag. 22 (1) (2002) 44–52.
[28] P. Trodden, A. Richards, Cooperative distributed MPC of linear systems with coupled constraints, Automatica 49 (2) (2013) 479–487.
[29] J. Löfberg, Oops! i cannot do it again: Testing for recursive feasibility in MPC, Automatica 48 (3) (2012) 550–555.
[30] D. Bertsekas, Dynamic programming and suboptimal control: A survey from ADP to MPC, Eur. J. Control 11 (4–5) (2005) 310–334.
[31] M. Doan, T. Keviczky, B. B. Schutter, A hierarchical MPC approach with guaranteed feasibility for dynamically coupled linear systems, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 393–406.
[32] R. Bourdais, J. Buisson, D. Dumur, H. Guéguen, P. Moroşan, Distributed MPC under coupled constraints based on dantzig-wolfe decomposition, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 101–114.
[33] A. Ioffe, V. Tihomirov, Theory of Extremal Problems, Elsevier, 2009.

[34] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge university press, 2004.
[35] A. Nedić, A. Ozdaglar, Approximate primal solutions and rate analysis for dual sub-gradient methods, SIAM J. Optim. 19 (4) (2009) 1757–1780.
[36] D. Yuan, S. Xu, H. Zhao, L. Rong, Distributed dual averaging method for multi-agent optimization with quantized communication, Systems Control Lett. 61 (11) (2012) 1053–1061.
[37] D.P. Bertsekas, Nonlinear Programming, Athena scientific Belmont, 1999.
[38] A. Al-Tamimi, F. Lewis, M. Abu-Khalaf, Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof, IEEE Trans. Syst. Man Cybern. B 38 (4) (2008) 943–949.
[39] K. Savla, E. Frazzoli, A dynamical queue approach to intelligent task management for human operators, Proc. IEEE 100 (3) (2011) 672–686.
[40] S. Koziel, D. Ciaurri, L. Leifsson, Surrogate-based methods, in: Computational Optimization, Methods and Algorithms, Springer, 2011, pp. 33–59.