The Fourth IEEE RAS/EMBS International Conference
on Biomedical Robotics and Biomechatronics
Roma, Italy. June 24-27, 2012

Robustness and Generalization of Model-Free Learning for Robot Kinematic Control using a Nested-Hierarchical Multi-Agent Topology

John N. Karigiannis and Costas S. Tzafestas

Abstract— **This paper focuses on evaluating the robustness and knowledge generalization properties of a model-free learning mechanism, applied for the kinematic control of robot manipulation chains based on a nested-hierarchical multi-agent architecture. In the proposed topology, the agents correspond to independent degrees-of-freedom (DOF) of the system, managing to gain experience over the task that they collaboratively perform by continuously exploring and exploiting their state-to-action mapping space. Each agent forms a local (partial) view of the global system state and task progress, through a recursive learning process. By organizing the agents in a nested topology, the goal is to facilitate modular scaling to more complex kinematic topologies, with loose control coupling among the agents. Reinforcement learning is applied within each agent, to evolve a local state-to-action mapping in a continuous domain, thus leading to a system that exhibits developmental properties. This work addresses problem settings in the domain of kinematic control of dexterous-redundant robot manipulation systems. The numerical experiments performed consider the case of a single-linkage open kinematic chain, presenting kinematic redundancies given the desired task-goal. The focal issue in these experiments is to assess the capacity of the proposed multi-agent system to progressively and autonomously acquire cooperative sensorimotor skills through a self-learning process, that is, without the use of any explicit model-based planning strategy. In this paper, generalization and robustness properties of the overall multi-agent system are explored. Furthermore, the proposed framework is evaluated in constrained motion tasks, both in static and non-static environments. The computational cost of the proposed multi-agent architecture is also assessed.**

I. INTRODUCTION

Understanding aspects of human cooperative behaviour constitutes a major research objective in the field of multi-agent systems. In robotics, related topics have been addressed by work done on cooperative mobile robots [10] [4], robotic hands, and multiple cooperating robot manipulators [11][16][22]. The common ground of all these approaches is that the motion of the object under cooperative manipulation (or pushing) actions is considered quasi-static, and that all the agents involved have predefined behavior models that are combined by employing specific control architectures.

Human behavior also demonstrates evolutionary characteristics and self-organizing abilities. In this context, the application of bio-inspired techniques such us Reinforcement Learning (RL) [20][2][6], evolutionary computation and fuzzy systems, constitutes an emerging research topic. Other approach to acquire robot manipulation skills is through Learning from Demonstration (LfD) [1], also referred to as Learning by Imitation [18]. By LfD, instead of learning by exploration, a policy is learned from examples, or demonstrations, provided by a teacher. These examples are defined as sequences of state-action pairs that are recorded during the teacher's demonstration of the desired robot behaviour. We note that a policy derived under LfD is necessarily defined only in those states encountered, and for those actions taken during the example executions.

John N. Karigiannis is Ph.D Candidate at the Faculty of Electrical and Computer Engineering, Division of Signals, Control and Robotics, National Technical University of Athens (NTUA), Zografou, Athens 15773, Greece johnkari@central.ntua.gr

Costas S. Tzafestas is with the Faculty of Electrical and Computer Engineering, Division of Signals, Control and Robotics, National Technical University of Athens (NTUA), Zografou, Athens 15773, Greece ktzaf@cs.ntua.gr

Inspired from the above research directions, this paper extends our previous work presented in [9][10], and addresses issues related to knowledge generalization and robustness achieved through model-free learning on multi-agent robot control and skill acquisition architectures. Learning is here approached not through demonstration and training, but as an autonomous exploration and self-learning (unsupervised) process, where each agent evolves a local sensori-motor behaviour by receiving information (reward signals) related to observations of task performance. With respect to our previous work, this paper applies multi-resolution goal training, aiming to explore how well the system manages to generalize the knowledge acquired during training, in order to subsequently reach new (untrained) goals. In addition, the robustness properties of the overall system are assessed, by considering a complex failure scenario where multiple agents fail at different time instants. A comparison of the proposed model-free approach to a typical model-based one is also performed, and the extensibility of our multi-agent system to constrained motion tasks is also evaluated. This paper also includes an analysis of the computational cost of our topology, which was not included in our previous work.

The paper is organized as follows. The following section outlines the proposed multi-agent framework and Section III describes more in detail the learning approach employed in this work. Section IV presents the numerical experiments performed and discusses the results obtained in the case of a simulated single kinematic chain, while Section V discusses issues related to the computational cost of the proposed multi-agent framework. Finally, concluding remarks and future work directions are presented in Section VI.

II. THE PROPOSED MULTI-AGENT FRAMEWORK

The proposed control framework has been described in detail in [9][10]. Fig. 1 presents the basic structure of the *multi-agent nested-hierarchical topology* for the proposed developmental robot manipulation control framework. In

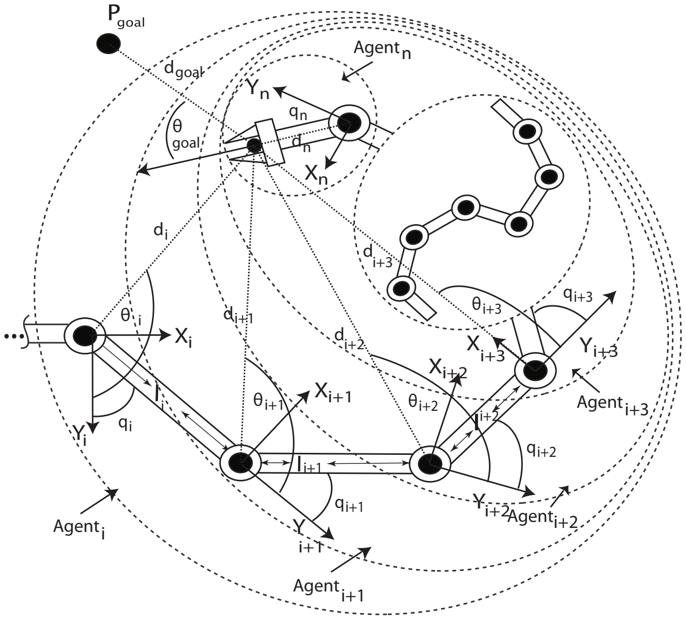

Fig. 1: Nested hierarchical topology of the multi-agent system, for the case of an n-DOF robot manipulator (open kinematic chain); positioning of the end-effector at a given goal position is, in this case, the considered manipulation task

order to facilitate understanding of the work presented in this paper, we highlight here the basic concepts of this multi-agent framework.

A. Mapping Agents to Degrees of Freedom

Each robot joint is assigned to an agent having as a function to govern local control at that joint level. The challenge here is to evolve global dexterity through progressive acquisition of local skills at each local agent level.

B. Nested-Hierarchical Multi-Agent Architecture

Each agent functions locally and selects an action independently of the rest, by performing an estimate of the actions that the rest of the agents could potentially perform. This is of course always performed as viewed in the local perspective of each agent, that is, with limited visibility only to those agents that are located below in the nested-hierarchical sense; in other words, the agent maintains a partial (local) view of the system topology, and does not know at any time how many agents constitute the whole system above or below and what is the exact underlying topology.

C. Continuous Problem Setting and State Definition

The learning process must also function in a continuous state-space, thus a fuzzification step is applied to the readings forming the system state. Learning is then accomplished in a discrete state-action mapping sense; a defuzzification step can subsequently be employed to perform action selection in a continuous domain. Let us consider a kinematic chain that is comprised of n dofs (agents $\mathrm{a}_i : i = 1, \ldots, n$), nested in the manner presented in Fig. 1. We define the state of every agent a_i as: $S_i =< q_i, \theta_i, d_i, \vec{g_i} >$, where: q_i is the current relative position of the i^{th} joint (i.e. the angular displacement of the i^{th} link with respect to the previous link in the kinematic chain), θ_i is the current angular position of the robot's end-effector with respect to the i^{th} agent, d_i is the current Euclidian distance of the robot's end-effector from the i^{th} agent, and $\vec{g_i}$ is a vector describing the position of the goal at the task space with respect to the end-effector ($\vec{g_i}$ is expressed in polar coordinates, comprising the distance d_{goal} and the angle θ_{goal} of the goal w.r.t. the reference frame of the end-effector agent). Therefore, the state vector of each agent is composed in total of five variables. Every agent a_i identifies certain variables of its state (constituting a partial -local- view of the full system topology) and forwards that information in a nested manner to the agents of the next layer in the hierarchy, in order to facilitate the process of identifying their own state variables. This whole process is composed of two subprocesses that evolve in a recursive manner. The first is evolving in a top-down manner, while the second one is evolving in a bottom-up way. Detailed description of this recursive process is presented in [10].

III. Continuous Reinforcement Learning

Let us assume a collection of n (homogeneous) agents, each agent a_i ($i \in [1, \ldots, n]$) having a finite set A_i of individual actions $a^i_j \in A_i$ available to it ($j = 1, \ldots, \mathrm{size}(A_i)$). Agents repeatedly operate within the framework of the environment posed, in which they each independently select an individual action to perform. In [8], RL is defined as the problem faced by an agent that must learn a behaviour through trial-and-error interactions with a dynamic environment. In terms of mathematical description, RL has been formalized as a Markov Decision Process (MDP). An MDP has four components: states, actions, transitions and reward distributions. More precisely, an MDP is a 4-tuple (S, A, T, r), where S denotes a finite set of states, A denotes the action space, T is a probabilistic transition function $T : S \times A \times S \rightarrow [0, 1]$, that denotes the probability for transition from a state s to a new state s' when a certain action a is applied, and $r : S \times A \rightarrow \Re$ is a reward function that denotes the reward for applying a certain action a to a certain state s.

At this stage, we need to provide a formal definition for the state of our system. Given each agent's structure formulated so far, the state of every individual agent a_i can be expressed as $< q_i, \theta_i, d_i, \vec{g_i} >$ and the entire multi-agent system state can be defined as: $S_t = \{< q_i, \theta_i, d_i, \vec{g_i} >, \; i = 1 \ldots n\}$, at a specific time instance t. For a 4-DOF manipulator, the corresponding state definition of each individual agent and subsequently the state definition of the entire system can then be written as: $S_t = \{< q_1, \theta_1, d_1, \vec{g_1} >, < q_2, \theta_2, d_2, \vec{g_2} >, < q_3, \theta_3, d_3, \vec{g_3} >, < q_4, \theta_4, d_4, \vec{g_4} >\}$. All agents wish to select actions that maximize the (expected) reward. Each agent contributes its own action component to the joint action that is eventually applied to the environment and determines the transition. The goal is to find a policy that maximizes the reward [2].

Before proceeding, let us adopt some standard game theory terminology in order to facilitate the discussion below [15]. A randomized policy for an agent a$_i$ is a distribution $\pi \in \Delta(A_i)$ (where $\Delta(A_i)$ is a set of distributions over the agent's action set A_i). Intuitively, $\pi(\alpha^i)$ denotes the probability of agent a$_i$ selecting an individual action α^i. A policy π is deterministic if $\pi(\alpha^i_j) = 1$ for some $\alpha^i_j \in A_i$. A collection of policies for all the agents is called policy profile, $\Pi = \{\pi_i : i \in [1, \ldots, n]\}$, where n is the number of agents. The expected value of acting according to a fixed profile can easily be determined. If each $\pi \in \Pi$ is deterministic, we can think of Π as a joint action. A reduced profile for agent a$_i$ is a policy profile for all agents but a$_i$ (denoted Π_{-i}). Given a profile Π_{-i} , a policy π_i is a best response for agent a$_i$ if the expected value of the policy profile $\Pi_{-i} \cup \{\pi_i\}$ is maximal for agent i; that is, agent a$_i$ could not do better using any other policy π'_i.

A. Action Selection and Reward Function

In our multi-agent system, each agent is able (independently from the others) to select its action among three discrete types of actions: *increase*, *decrease* or *maintain* its joint angular displacement. The action selection problem, however, is significantly difficult if there are multiple optimal joint actions. If the joint actions are chosen randomly, or in some way reflecting personal biases, then there is a risk to select a suboptimal or uncoordinated joint action. So, we have the general problem of equilibrium selection (or joint action selection), which can be addressed in several ways. One way is the communication among the agents [19]; another is to introduce conventions or rules that restrict behaviours and so to ensure coordination. In this paper, we apply a mechanism that results in a coordination among the agents' action through a repeated execution of the specific task by the same agents. In this action selection mode, each agent a$_i$ keeps a count of the number of times a specific action has been performed in the past by the same agent (as well as by its collaborative agents). This concept, although simple, is some times quite effective, and is known as fictitious play [7][3]. More precisely, each agent a$_i$ keeps a count $C^i(\alpha^j_k)$, for every agent a$_j$ that is visible by a$_i$, indicating the number of times agent a$_j$ has selected action $\alpha^j_k \in A_j$ in the past. When a task is assigned to our multi-agent system, each agent a$_i$ treats the relative frequencies of the moves of all other agents a$_j$ as indicative of their current policy. That is, agent a$_i$ assumes that agent a$_j$ performs action $\alpha^j_k \in A_j$ with probability:

$$P^i(\alpha^j_k) = \frac{C^i(\alpha^j_k)}{\sum_{b^j \in A_j} C^i(b^j)} \tag{1}$$

We note that most models (in game theory) assume that each agent can observe the actions executed by its counterparts with certainty. What we actually employ is something more general that allows each agent to obtain an observation that is related stochastically to the actual joint action selected. Action selection is more difficult when agents are not aware of the rewards associated with various joint actions, hence the expected reward associated with individual and joint actions has to be estimated based on previous experience. Q-learning algorithm developed by Watkins [21] is the most frequently used RL algorithm. An agent estimates the utility for doing each of its actions, chooses an action based on a selection function of the expected values, observes the reward, and then updates the Q-value or the estimate of the utility of that action. In the case of a stateless setting, we have an agent updating its estimated Q-values as follows: $Q(s, \alpha) \leftarrow Q(s, \alpha) + \lambda(r - Q(s, \alpha))$, where in state s action α was performed resulting in reward r. Here, λ is the learning rate ($0 \leq \lambda \leq 1$) governing to what extent the new sample replaces the current estimate.

The next issue that we have to address concerns the action selection function, which is particularly important since effective learning requires sufficient exploration. The action selection mechanism employed in this work is a variant of ε-greedy, called ε-decreasing, where the probability of an exploration action decreases as trials progress. This action selection mechanism starts with an exploration probability: $p_{\text{explore}}(t) = \varepsilon \cdot (T(t) - 1)/(T_{max} - 1)$. To compute the probability of choosing an action, we employ a Boltzmann distribution as explained below. On each trial, with probability: $1 - p_{\text{explore}}(t)$, each agent a$_i$ chooses the action α^i with the greatest estimated $\pi(\alpha^i)$:

$$\pi(\alpha^i) = \frac{e^{\frac{E(\alpha^i)}{T}}}{\sum\limits_{\alpha^i_j \in A_i} e^{\frac{E(\alpha^i_j)}{T}}} \tag{2}$$

where $E(\alpha^i)$ denotes the expected value of an action α^i and T is the temperature parameter that is controlled to diminish over time so that the exploitation probability is increased. Temperature determines the likelihood for an agent to explore other actions: when T is high, even when the $E(\alpha^i)$ of an action is high, an agent may still choose an action that appears to be less desirable. This exploration strategy is especially important in stochastic environments like the one we are examining, where payoffs received for the same action combination may vary. For effective exploration, high temperature is used at the early stages of a task. The temperature is then decreased over time to favour exploitation, as the agent is more likely to have discovered the true values of different actions. The temperature T as a function of time is given by: $T(t) = 1 + T_{max} \cdot e^{-st}$, where t here denotes time units (i.e. the iteration number), T_{max} is the initial temperature and s is the rate of decay (in our case, $s \in [0, 1]$, where a value close to 0 facilitates exploration, while a value of s close to 1 facilitates exploitation).

Now, let us elaborate on the definition of the expected value $E(\alpha^i)$. The presence of multiple agents, each one learning simultaneously with others, is a potential impediment to the successful employment of Q-learning (and RL in general) in multi-agent settings like the one considered in this paper. When an agent a$_i$ is learning the value of its actions in the presence of other agents, it is learning in a non-stationary environment. Thus, convergence of the Q-values is not guaranteed. What we need is each agent's policy to settle. This is a key issue and is discussed hereafter. In general, there are two distinct ways in which Q-learning could be applied in a multi-agent system; the *Independent* and the *Joint-Action Learner* algorithm [13][14][5]. In an Independent Learner algorithm, each agent learns its Q-values regardless of what the other agents are performing. This method is appropriate to be employed when an agent has no reason to believe that other agents are acting strategically. Joint action learner algorithm is the one where the agents do not learn Q-values of their individual actions but the Q-values of their joint actions. This implies that each agent can observe the actions of other agents. Each agent in such a system maintains beliefs about the policies of other agents. So, an agent a_i assesses the expected value $E(\alpha^i)$ of selecting an individual action α^i at a current state s to be:

$$E(\alpha^i) = \sum_{\alpha^{-i} \in A_{-i}} \left\{ Q\left(s, \alpha^{-i} \cup \{\alpha^i\}\right) \cdot \prod_{\alpha_k^j \in \alpha^{-i}} \left[P^i(\alpha_k^j)\right] \right\} \quad (3)$$

In the above equation, A_{-i} denotes the set of all possible joint actions that can performed by the group of agents that are considered "visible" by agent a_i, in the nested hierarchical sense (i.e., in our case, agents that are below in the hierarchy), $\alpha^{-i} \in A_{-i}$ denotes one such joint action performed by this group of agents, and $\alpha_k^j \in \alpha^{-i}$ is an individual action performed by a single agent a_j in this group. The reward that an agent receives at time instant t, after selecting certain action and moving to a new state, is defined by the reward function $R(t)$, which is formulated as follows:

$$\begin{cases} \texttt{if } (D_{goal}(t) \leq D_{\min}) \wedge (\Delta D_{goal}) \leq 0) \texttt{ then } R(t) = e^{-c \cdot D_{goal}(t)} \\ \texttt{if } (D_{goal}(t) > D_{\min}) \texttt{ then } R(t) = -2 \\ \texttt{if } (D_{goal}(t) < D_{\min}) \wedge (\Delta D_{goal}) > 0) \texttt{ then } R(t) = -1 \end{cases} \quad (4)$$

where $D_{goal}(t)$ is the distance from the goal at iteration time t, $D_{\min}$ is a threshold distance after which the agents start receiving rewards, and ΔD_{goal} is the rate of change of distance from the goal.

IV. Numerical Experiments

We have previously considered, in [9], a series of numerical experiments, on a single redundant (planar) kinematic chain, consisting of four links with 4 independent DOFs employing the proposed multi-agent architecture. Successful completion of the learning process has been demonstrated in [9], where experimental results show how the agents acquire knowledge to collaboratively reach their goal. In this paper, we focus on evaluating the generalization properties of the system; that is, how the agents use the knowledge acquired and how, without any additional training, they can explore and reach new targets different from the ones trained. Moreover, we evaluate robustness properties of the proposed multi-agent system and extensibility of the approach to more complex kinematic topologies, involving constrained manipulation tasks in both stationary and non-stationary environments.

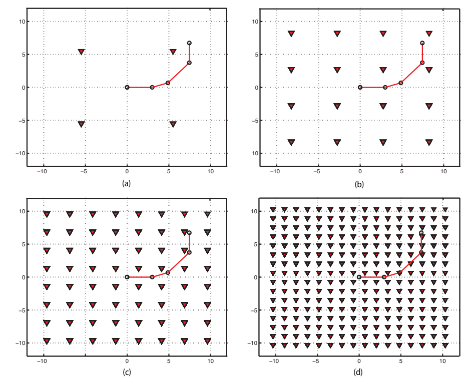

Fig. 2: Resolution levels and corresponding training targets

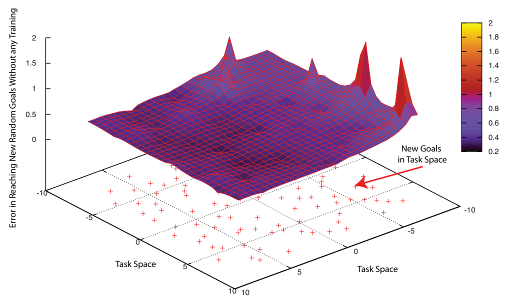

Fig. 3: Target-reaching error in the entire task space, with 4 training points

A. Multi-Resolution Goal Training and Generalization

In this section, we consider multiple layers of training data, each layer representing a different resolution at the target space. Initially, the task space is recursively subdivided into levels of increasing resolution, in a quad-tree structure as can be seen in Fig. 2. Starting from *Level*-1, containing four nodes (quadrants), we progressively move down to *Level*-4 that contains 256 nodes. Each node corresponds to a single target position, for which the multi-agent system receives training. Therefore, for each resolution level, the multi-agent system is trained on a different set of target-positions, increasing in size according to the training resolution *Level* (i.e.: at *Level*-1, the four nodes correspond to four goal positions on which the system is trained, for *Level*-2, 16 target positions are trained, etc).

For the purpose of the experiment presented in this Section, we thus investigate a range of resolutions starting from 4 up to 4^4 target training points, as shown in Fig. 2. For each resolution level, after completion of training on the corresponding target points (constituting the training-set at this resolution level), a test-set is generated consisting of 100 new goals randomly distributed in the entire task space (by employing Halton sequences to obtain a uniform distribution of points in the task space). Our aim is to explore the accuracy by which, at each resolution level, the multi-agent system manages to reach all the new (untrained), randomly generated goals, without performing any additional training.

The experimental process is the following: a) Select a

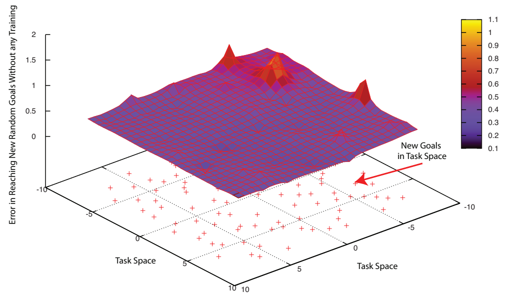

Fig. 4: Target-reaching error in the entire task space with 16 training points

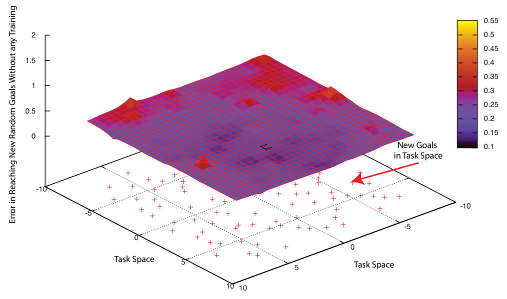

Fig. 5: Target-reaching error in task space with 256 training points

resolution level; b) Perform off-line training on the target points of the corresponding resolution level; c) After completion of training, issue successive on-line requests to reach all 100 randomly generated goal positions of the test set. The target-reaching results, for three different resolution levels, are depicted in Figures 3 to 5. By observing these plots, it is clear that even at the lowest resolution level (comprising only 4 training points in the entire task space), the proposed multi-agent topology manages to successfully reach almost all 100 randomly generated new goals assigned. It is also clear that the situation improves as the training resolution increases from 4 to 16 up to 256 points (both the height of the '*bumps*', representing the error for reaching specific goals, as well as their population, diminish).

Fig. 6 depicts the statistics of the target-reaching generalization results. We observe that even for the 4 training points (*Level*-1) resolution, the mean value of the error is quite low ($mean = 0.3814$), but with significant standard deviation value ($std = 0.5596$), showing again that most of the random target points at the test set are successfully reached, even using the lowest resolution level for training. The mean value of the error decays smoothly and reaches, at the 64 points resolution (*Level*-3), a value of $mean = 0.2366$, while the corresponding standard deviation drops to the value of $std = 0.2036$. We also observe that, in this experiment, increasing the resolution from *Level*-3 to *Level*-4 (i.e. from 64 to 256 training points) does not significantly improve

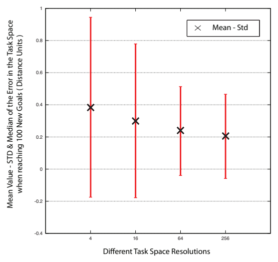

Fig. 6: Mean and STD values of the target-reaching error at the task space for different training resolution levels

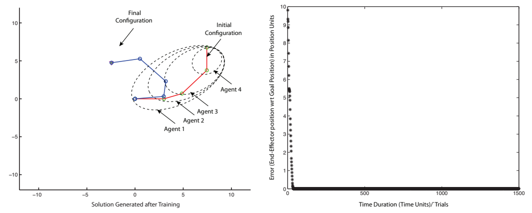

Fig. 7: Kinematic solution generated by the multi-agent system and position error over time (for the fully operational system, with all agents active).

the accuracy of the system. These findings clearly highlight the powerful generalization properties of the proposed multi-agent architecture. Furthermore, considering the results presented in Fig. 6, a resolution of level 2 to 3 (16 to 64 training points, in this case) is found to be sufficient to adequately cover the entire task-space.

B. Robustness to Changes in Kinematic Topology

A key advantage of a distributed multi-agent control strategy (with respect to a typical single-agent model-based architecture) is related to robustness. In this paragraph, we evaluate the capacity of the proposed multi-agent architecture to compensate, at run-time, for partial failures that may occur to the agents (degrees of freedom) comprising the system and, therefore, to adapt to unpredictable sudden changes in the kinematic topology.

We start by training the multi-agent system to reach a goal position, as described in the previous section. After

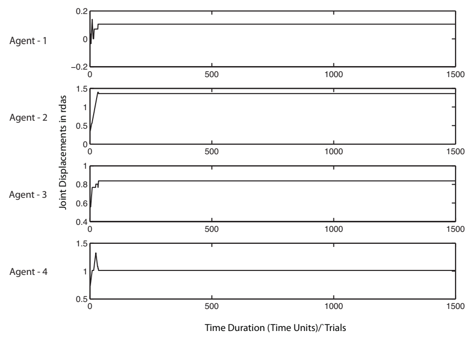

Fig. 8: Actions (joint displacements) of the agents. All four agents are active and cooperate to reach the goal position.

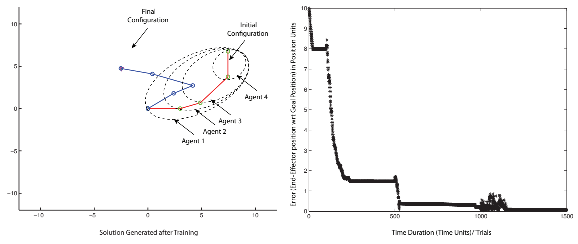

Fig. 9: Kinematic solution generated and tracking error after failure and progressive recovery of agents 1 to 3.

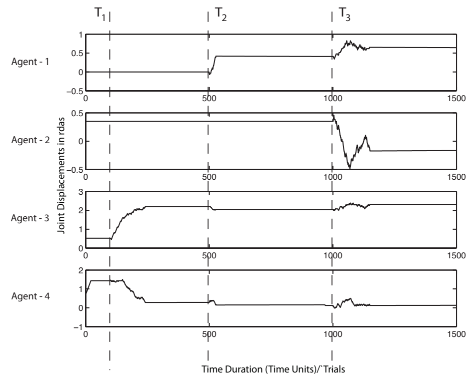

Fig. 10: Agents are adapting their actions dynamically to compensate for unpredictable failure-recovery events of agents 1 to 3.

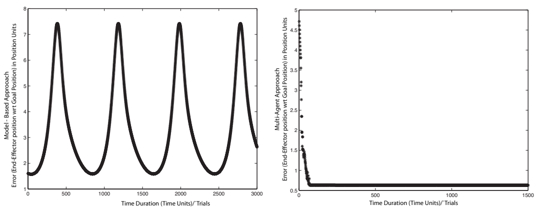

Fig. 11: Comparative tracking error results (model-based vs. multi-agent approach) in a failure scenario with agents 2 and 3 fully blocked.

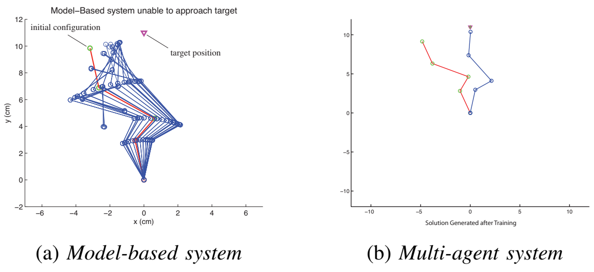

Fig. 12: Comparative target-tracking response for the model-based and the multi-agent systems in a failure scenario with agents 2 and 3 fully blocked.

completion of the training period, the fully operational multi-agent system is now able to find a solution and decrease the tracking error (end-effector w.r.t. goal position), as depicted in Fig. 7. Fig. 8 shows the actions (joint displacements) of individual agents.

In the sequel, we simulate a number of failure situations in the agents, in order to evaluate the robustness properties of the system. Firstly, we simulate an initial partial failure of the system, assumed to be progressively recovered. We consider that, initially (at $T = 0$), three of the agents (agents 1, 2 and 3) are non-operational and only agent 4 is responding. This situation corresponds to a failure of receiving and executing any action by these agents, which means that agents 1 to 3 stay blocked to their initial angular position. Subsequently, at $T_1 = 100$, agent 3 starts responding and later on, at $T_2 = 500$ and $T_3 = 1000$, agents 2 and 3, respectively, are also recovering from their failure. Our goal is to explore how, without any external assistance, the multi-agent system manages to compensate for the imposed failure situation. The results are quite interesting, demonstrating that the multi-agent system manages, at run-time, to find a new solution adapting to the unpredictable changes in the kinematic topology caused by the considered failure situation with some of the agents not responding during operation. Fig. 9 depicts the new kinematic solution generated by the multi-agent system, and the evolution of the tracking error. Fig. 10 shows the actions of the individual agents, and how they adapt to cope for the disturbance caused by the failure events.

To better highlight the superior robustness properties exhibited by the proposed (model-free) multi-agent system, as compared to a model-based (single-agent) control strategy, we assume the failure situation where joints 2 and 3 are completely blocked, and we perform the same experiment but this time applying a model-based (resolved motion rate) kinematic control at the joints (using the Jacobian matrix pseudo-inverse). The tracking error results obtained are depicted in Fig. 11, comparatively for the model-based and the multi-agent approach. The target-tracking response of the kinematic chain is also illustrated in Fig. 12. From these plots, it is evident that the kinematic chain under model-based kinematic control is unable to reach the designated target position, generating oscillatory motion patterns between specific configurations. This is obviously due to the fact that the kinematic model is in this case a-priori invalid and a typical model-based system does not have the mechanism (without any remodelling and replanning) to cope with such unpredictable errors, therefore potentially generating motions inconsistent with the target position at the task space. On the contrary, the proposed multi-agent system is still able to find a feasible solution (which seems, in fact, consistent with a minimum-energy configuration), as shown in Fig. 12(b). The above results clearly demonstrate the superior performance of the multi-agent approach in terms of its adaptability to changes in kinematic topology and its capacity to cope for unpredictable and complex failure situations.

C. Extensibility to Constrained Motion Tasks

To evaluate the extensibility of the proposed framework to a more complex topology, we assume the following constrained motion task. A seven DOF kinematic chain is considered and the corresponding multi-agent system must learn how to reach a target position that is located at the inside end of a narrow corridor. The goal is to generate

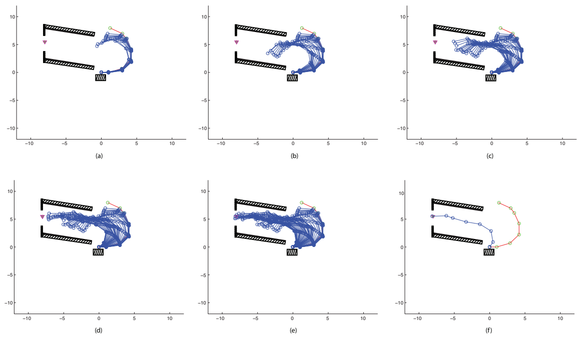

Fig. 13: Extensibility of the proposed multi-agent framework to a 7 DOF kinematic chain performing a constrained motion task within static environment, involving collision-free target reaching inside a narrow corridor

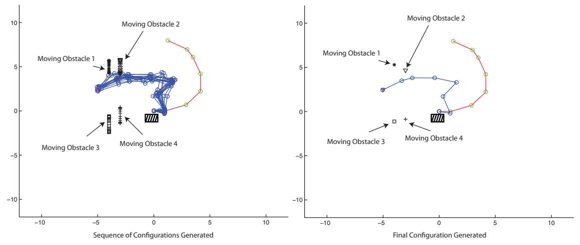

Fig. 14: Extensibility of the proposed multi-agent framework to a 7 DOF kinematic chain performing a constrained motion task within non-static environment (four oscillating obstacles) involving collision-free target reaching

a collision-free motion for the kinematic chain, enabling the end-effector to reach the target position without using any a-priori model of the environment. In such an experimental scenario, a typical model-based (single-agent) kinematic approach would have to introduce additional distance optimization criteria, which would make the analytic redundancy resolution problem more complex and more sensitive to modelling errors. The proposed distributed multi-agent architecture is naturally extended to cope with such a topology, considering only an additional term on the reward function that signals collision events during training. The results obtained are depicted in Fig. 13. These results show that the multi-agent system manages to evolve behaviors that effectively guide the kinematic chain to enter the opening of the narrow corridor and to successfully reach the target position, generating a natural collision-free path.

Subsequently, we consider a non-static environment comprising moving obstacles. In Fig. 14 and Fig. 15, we see that the multi-agent system learns to avoid collision with all four oscillating obstacles throughout the simulation, keeping a minimum safe distance from all obstacles (with a mean value of *mean* $= 1.0965$), while at the same time maintaining continuous contact with the target for the entire runtime, as can be seen in Fig. 15b (with a position error having a mean value of: *mean* $= 0.2060$).

V. Computational Cost Considerations

The state space S of the multi-agent system is composed of the local state spaces $S_1, S_2, \dots S_n$ of the individual

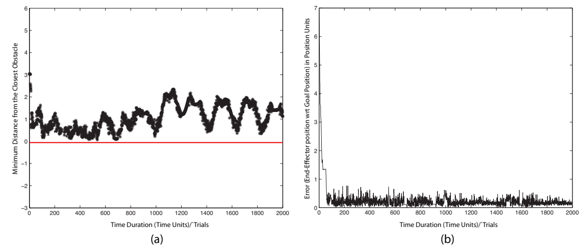

Fig. 15: The multi-agent system maintains contact with the target (Fig. 15b), while it avoids collision with all moving obstacles keeping a safe minimum distance for the entire duration of the run(Fig. 15a)

agents. Every local state-space is comprised of homogeneous state parameters. If we consider the system as a single agent then $S = S_1 \times S_2 \times \cdots \times S_n$, meaning that the cardinality of the state-space in a single-agent representation is: $|S| = |S_n|^n$. By adopting the proposed multi-agent (nested hierarchical) architecture along with a uniform state definition for every agent, the computational cost of the value iteration problem that we are solving is reduced, as compared to a single-agent approach.

The proposed multi-agent architecture is defined by homogeneous agents, meaning that all of them have the same number of state variables that uniquely define their state for all possible configurations. This implies that the cardinality $|\cdot|$ of every local state-space is the same: $|S_1| = |S_2| = |S_3| = \cdots = |S_i| = |S|$ for every agent i. According to the nested-hierarchical architecture, each agent is able to monitor only those agents that are below in the hierarchy, in order to formulate a joint action. Therefore, the corresponding action space of each agent is reduced, as we move from a higher to a lower level in the hierarchy. Eventually, the cardinality of the joint action space is $|A|^i$, where i is the number of agents that participate in the joint action at the specific level of the hierarchy, and $|A|$ is the number of distinct actions of each agent. Assuming that the state-space is finite, the number of state-action pairs to be updated at every iteration is: $|S| \cdot |A|^i$. In order to update the value for a given state-action pair, the maximization over the joint action space is solved by enumeration over $|A|^i$ elements. So, the cost per iteration is $|S| \cdot |A|^i \cdot |A|^i$ or $|S| \cdot (|A|^i)^2$. Assuming that our algorithm runs for L iterations and for n agents, the total computational cost can be computed as follows:

$$L \cdot \sum_{i=1}^{n} \left\{ |S| \cdot (|A|^i)^2 \right\} =$$

$$= L \cdot |S| \sum_{i=1}^{n} (|A|^2)^i = L \cdot |S| \cdot \frac{(|A|^2)^{n+1} - |A|^2}{|A|^2 - 1} =$$

$$= L \cdot |S| \, (|A|^2)^n \frac{|A|^2 - \frac{1}{(|A|^2)^{n-1}}}{|A|^2 - 1} =$$

$$\simeq L \cdot |S| \cdot (|A|^2)^n \cdot K \tag{5}$$

where, for a large value of n, we can assume that

$$K \cong \frac{|A|^2}{|A|^2 - 1} \tag{6}$$

Now comparing the above cost with the case of a single-agent system is trivial. In a single-agent representation, the state parameters, instead of being uniformly distributed among several agents, would be accumulated on a single agent, resulting to an exponential increase of the state-space cardinality. Thus, the computational cost for a single-agent architecture would be:

$$L \cdot |S|^n \cdot (|A|^2)^n \tag{7}$$

Comparing Eq. (5) to Eq. (7) it is clear that, when the number n of agents increases, the computational cost in the single-agent architecture increases exponentially, since the cardinality of the state space in (7) is raised to the power of n. In the case of the open kinematic chain considered in Section IV-C, the proposed multi-agent system comprises seven nested agents, so $n = 7$. The state-space of each agent, as defined in Section II-C, is composed of six state variables. Each state variable is fuzzified with 8 memberships functions. Therefore, the cardinality of the state-space of each agent is: $|S| = 8^6$. Furthermore, as defined in Section III-A, each agent has three distinct actions, so: $|A| = 3$. Each training epoch, within which every agent is allowed to operate, is assumed to have a duration of 1500 iterations, so: $L = 1500$. Based on these assumptions, the computational cost when employing the proposed multi-agent architecture, equals: $1500 \cdot 8^6 \cdot (3^2)^7 \cdot \frac{3^2}{3^2-1} = 2.1158 \times 10^{15}$ operations. In the case of a single-agent approach, this cost would be: 6.8663×10^{47}. Therefore, as the number of agents increases, it is evident that the computational benefit of the proposed multi-agent architecture (as compared to a single-agent approach) becomes significant.

VI. Conclusion and Future Work

This paper explores the applicability of a multi-agent nested-hierarchical architecture in the domain of dexterous robot kinematic control, evaluating key system properties related to knowledge generalization and robustness. Furthermore, the proposed multi-agent system, owing to its homogeneous characteristics (all agents obey the same modular internal structure) and to its hierarchical formation, facilitates scaling to more complex structures. Fig. 16 depicts a potential application of the proposed framework, where a more complex multi-agent topology could be envisaged. In addition, an analysis of the computational cost shows a significant gain of the proposed distributed multi-agent approach as compared to a typical centralized single-agent architecture.

By employing the proposed framework in the domain of dexterous manipulation, we believe that challenging problems can be tackled in a very elegant and powerful way (in the sense of modularity, robustness and extensibility). Similar (in some ways equivalent) problem settings, like grasp planning, locomotion control, or designing optimal climbing (and generally gaiting) patterns, could also be approached within the same framework, lending to the notions of evolving cooperative learning and developmental robot control skills.

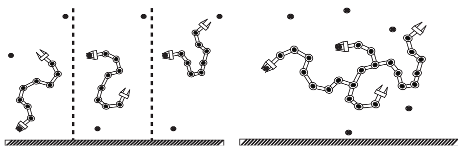

Fig. 16: Dexterous robotic chains performing hybrid locomotion / manipulation tasks (such as climbing)

References

[1] B. D. Argall, et al., "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, 57(5), 469-483, May 2009.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[3] G. W. Brown, "Iterative solution of games by fictitious play," In: *Activity Analysis of Production and Allocation*, T.C. Koopmans (editor), John Wiley, New York, 1951.

[4] S.S. Nestinger, H. H. Cheng, "Mobile-R: A reconfigurable cooperative control platform for rapid deployment of multi-robot systems," in Proc. *IEEE/RSJ Int. Conf. Robotics and Automation (ICRA'2011)*, pp. 52-57, Shanghai, 2011.

[5] C. Claus and C. Boutilier, "The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems," in Proc. *AAAI'98*, pp. 746-752, 1998.

[6] P. Dayan and L. F. Abbott, *Theoretical Neuroscience, Computational and Mathematical Modeling of Neural Systems*, MIT Press, 2001.

[7] D. Fundenberg and D. M. Kreps, "Lectures on Learning and Equilibrium in Strategic-Form Games," Technical Report, *CORE Lecture Series*, Louvain-La-Neuve, Belgium, 1990.

[8] L. P. Kaelbling, M. L. Littman and A. W.Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, 4, 237-285, 1996.

[9] J. N. Karigiannis, C.S. Tzafestas, "Multi-agent hierarchical architecture modeling kinematic chains employing continuous RL learning with fuzzified state space," *2nd IEEE RAS & EMBS Int. Conf. on Biomedical Robotics and Biomechatronics* (BioRob'2008), pp. 716-723, 19-22 Oct. 2008.

[10] J. N. Karigiannis, T.I. Rekatsinas, and C. S. Tzafestas, "Fuzzy rule based neuro-dynamic programming for mobile robot skill acquisition on the basis of a nested multi-agent architecture", in Proc. *IEEE/RAS Int. Conf. Robotics and Biomimetics (ROBIO)*, Tianjin, China pp. 312-319, 2010.

[11] O. Khatib et al., "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in Proc. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Osaka, Japan, vol. 2, pp. 546-553, 1996.

[12] J. R. Kok, N. Vlassis, "Sparse Tabular Multiagent Q-Learning," *Proceedings of Annual Machine Learning Conference of Benelearn* 2004.

[13] M. Lauer, M. Riedmiller, "Reinforcement Learning for Stochastic Cooperative Multi-Agent Systems," *Third International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS'04), vol. 3, pp. 1516-1517, 2004.

[14] M. McGlohon and S. Sen, "Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy," *International Journal on Lateral Computing*, vol. 1, pp. 58-64, August, 2005

[15] R. B. Myerson, *Game Theory: Analysis of conflict*, Harvard University Press, Cambridge 1991.

[16] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, 1990.

[17] D. Rus, "Coordinated Manipulation of Objects," *Algorithmica*, 19(1): 129-147, 1997.

[18] S. Schaal, "Is imitation learning the route to humanoid robots," *Trends Cogn. Sci.*, 3(6): 233-242, 1999.

[19] Y. Shoham and M. Tennenholtz, "On the synthesis of useful social laws for artificial agent societies," *Proc. AAAI-92*, pp. 276-281, San Jose, 1992.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[21] C. Watkins, "Learning from Delayed Rewards," *PhD Thesis*, University of Cambridge, England, 1989.

[22] Y. Yoshikawa and X. Zheng, "Coordinated dynamic hybrid position/force control for multiple robot manipulators handling one constrained object," *Int. J. Robot. Res.*, vol. 12, pp. 219-230, June 1993.