Reinforcement Learning for Emergent Behavior Evolution in Complex System-of-Systems

Anitha Murugesan
Honeywell Aerospace
Plymouth, Minnesota, USA
email: anitha.murugesan@honeywell.com

Ramakrishnan Raman
Honeywell Technology Solutions Lab
Bangalore, Karnataka, India
email: ramakrishnan.raman@honeywell.com

***Abstract*—The ease of inter-connectivity among modern systems is permeating numerous System-Of-Systems (SoS), wherein multiple, independent systems interact and collaborate to achieve unparalleled levels of functionality that are otherwise unachievable by the constituent systems in isolation. This has resulted in exponential increase in complexity associated with modern systems and SoS. Complex SoS are characterized by emergent behavior which is very difficult, if not impossible, to anticipate just from knowledge of constituent systems. The emergent behavior manifests at the boundary of the SoS and impacts the Measures of Effectiveness (MOEs) of the SoS. In the context of SoS, each constituent system has its own MOEs, while the SoS has its own MOEs. Constituent systems collaborate and interact with each other, towards achieving the desired functionality and behavior at SoS level. Recently, there is an explosion in the adoption of Machine Learning techniques and models in various systems, and these techniques are increasingly being used to control many physical systems, such as cars and drones. Reinforcement Learning is a type of machine learning approach that allows agents to optimally learn strategies through interactions with its environment. This paper presents a novel approach towards using reinforcement learning models and techniques for evolving MOEs of the constituent systems and SoS towards addressing emergent behavior. The proposed approach, through SoS-Constituent System MOE Relationship, enables constituent systems to learn and adapt their behaviors in tandem with the evolution of emergent behavior at SoS level.**

Keywords*—*Systems of Systems; Emergent Behavior; Measures of Effectiveness; Reinforcement Learning; Complexity.

I. INTRODUCTION

Advances in machine learning have enabled the development of sophisticated autonomous systems such as self-driving vehicles, and drones. Though developed independently, these systems are expected to be brought together as System-of-Systems (SoS) and operate in a real-world environment. This demands integration of the heterogeneous and inter-operable systems to provide superior levels of functionality.However, to operate in SoS context the systems should be able to achieve their objectives as well as adapt based on SoS-level objectives– that are typically defined as system-level and SoS level Measures of Effectiveness (MOE). For example, consider a self-driving truck planning a shortest path (truck MOE) through the city using its own navigational aids. However, if certain roads in the city (SoS context) have to be used for other high-priority purposes (SoS MOE), the truck is expected to alter its path to respect the city-level constraint. While the new path may not be optimal for the truck, it is expected to near-optimally meet the MOEs at both SoS and system levels. Conventional approaches involve human intelligence in understanding such relationships between constituent systems MOEs and SoS MOEs, towards addressing emergent behavior and MOE evolution. However, with the recent evolution of autonomous systems, there is compelling need and interest to explore approaches that enable systems to automatically learn the implications at SoS level.

In this paper, we propose the use of Reinforcement Learning (RL) approaches to enable a system to learn optimal policies in SoS-context. Our approach trains non-collaborative, independent agents whose rewards are uniquely designed to balance system-level and SoS-level goals. Our approach leverages the SoS-Constituent System MOE Relationship to uniquely design the reward structure of each system (embedded with an intelligent agent). This enables constituent systems to learn policies respecting the SoS and system-level MOE towards addressing evolving emergent behavior.

The rest of the paper is organized as follows: Section II discusses complexity, emergence and MOE evolution in the context of SoS, and briefly introduces reinforcement learning. Section III illustrates the proposed approach, while Section IV describes the simulation of the proposed approach, and the experimental results. Finally, Section V has the conclusion.

II. BACKGROUND & RELATED WORK

A. Systems and SoS

A system can be considered as an integrated and interacting combination of elements and/or subsystems to accomplish a defined objective [1]. These elements may include hardware, software, firmware and other support. SoSs are systems of interest whose system elements are themselves systems [2]. SoS has evinced keen interest among the systems engineering community, and there has been significant research pertaining to principles and practices on the architecture design, development, deployment, operation and evolution of SoS [3]-[6]. Applications of SoS principles and practices span many domains, including electrical power distribution, and Internet-of-Things. SoS characteristics discussed in literature include operational/ managerial independence, emergent behavior and evolutionary development.

B. Complexity and Emergence

In a general sense, the adjective "complex" describes a system or component that by design or function or both is difficult to understand and verify. There are different types of complexity measures discussed from different perspectives [7]. Emergence, hierarchical organization and numerosity are some of the characteristics of complex systems [8]. Emergence refers to the ability of a system to produce a highly-structured collective behavior over time, from the interaction of individual subsystems [7]. Common examples include a flock of birds flying in a V-formation, and ants forming societies of different classes of individual ants, wherein these patterns are not induced by a central authority. For a system, emergent behavior refers to all that arises from the set of interactions among its subsystems and components. Complex systems and SoS are expressed by the emergence of global properties which are very difficult, if not impossible, to anticipate just from a complete knowledge of component or subsystem behaviors [9][10]. Emergent behavior can be characterized as positive or negative, depending on the impact on the MOEs. The challenge for complex SoS is that there is inadequate knowledge on combination of events that would result in a negative emergent behavior. Specifically, for complex SoS, the "stringing" together of the constituent systems results in unique functionality and emergent behavior being exhibited at the SoS level that is very difficult to envision and predict, and cannot be attributed to any of the constituent systems individually.

C. MOEs in SoS

MOEs are the operational measures of success that are closely related to the achievement of the objective of the system of interest, in the intended operational environment under a specified set of conditions [1]. MOEs are measures designed to correspond to accomplishment of mission objectives and achievement of desired results. MOEs provide quantifiable benchmarks against which the system concept and implementation can be compared. It reflects the overall customer and user satisfaction, and it manifests at the boundary of the system. MOEs are independent of the specific solution. Example of MOEs include service life of satellite, search area coverage and survivability. Failure of the system to meet an MOE implies that the system does not meet its purpose and objectives [11].

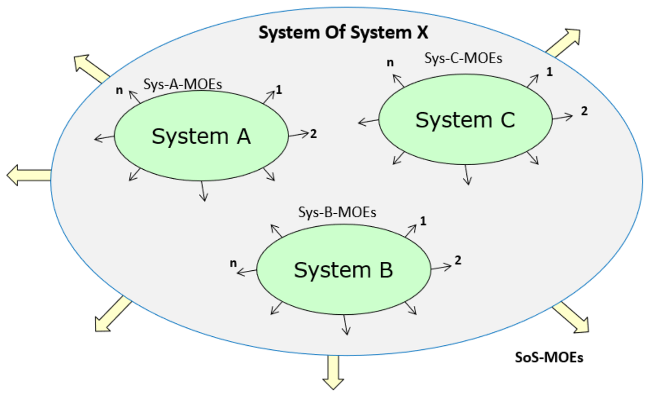


Figure 1. SoS-Constituent System

Understanding MOEs is critical to analyze the impact of the emergent behavior at SoS level. In the context of SoS, each constituent system of the SoS has its own MOEs. The MOEs for a constituent system can be independently measured to assess its success. MOEs of the SoS are the operational measures of success for the SoS as a whole. Figure 1 illustrates SoS MOEs versus constituent system MOEs. System A can have MOEs: SysA-MOE-1, SysA-MOE-2 and SysA-MOE-3. The MOEs of System-A represent the measures of success for System-A as an independent system, and the MOEs for System-A can be independently measured to assess the success of System-A. In addition to each constituent system having its own MOEs, MOEs are also relevant at the SoS level, i.e., SoSx would also have its own MOEs. The MOEs at the SoS level represent the measures of success for the SoS as a whole. Figure 2 further illustrates the impacts on MOEs at system level and at SoS level. The MOEs of the system are impacted by the behaviors exhibited by the system. Similarly, the MOEs of the SoS are impacted by the behaviors exhibited at SoS level. Further, the behaviors exhibited at constituent system level also impacts the SoS MOEs.

D. MOE Relationship Matrix

As discussed earlier, one of the characteristics of SoSs is that the stringing together of the constituent systems results in unique behavior and functionality that gets exhibited at the boundary of the

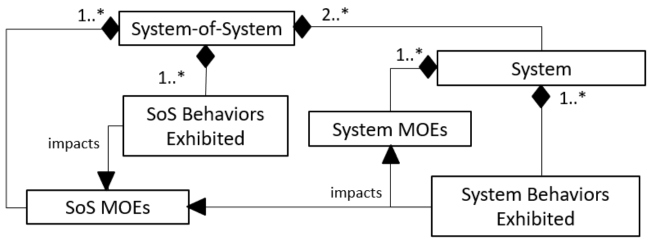
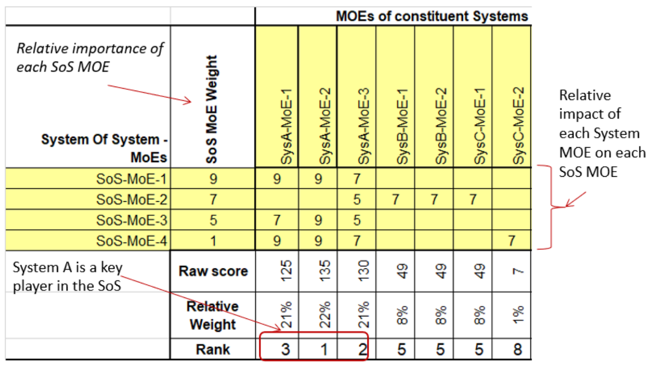


Figure 2. SoS-System Behaviors



		MOEs of constituent Systems						
Relative importance of each SoS MOE System Of System - MoEs	SoS MoE Weight	SysA-MoE-1	SysA-MoE-2	SysA-MoE-3	SysB-MoE-1	SysB-MoE-2	SysC-MoE-1	SysC-MoE-2
SoS-MoE-1	9	9	9	7				
SoS-MoE-2	7			5	7	7	7	
SoS-MoE-3	5	7	9	5				
SoS-MoE-4	1	9	9	7				7
System A is a key player in the SoS	Raw score	125	135	130	49	49	49	7
	Relative Weight	21%	22%	21%	8%	8%	8%	1%
	Rank	3	1	2	5	5	5	8

Relative impact of each System MOE on each SoS MOE

Figure 3. MOE Relationship Matrix

SoS, i.e., the behavior may not be attributed to any of the constituent systems functioning independently. With this being the case, the relationships between the MOEs of the SoS vis-à-vis the MOEs of the constituent systems might turn out to be complex and dynamic. There are different means to analyze the MOE relationships between the constituent systems and SoS. SoS-System MOE relationship matrix [6][12] is one of the means to analyze the relationships, as indicated in Figure 3. The impact of different system MOEs on the SoS MOEs could vary. There might be scenarios where a specific constituent system might be meeting all its MOEs, but the SoS MOEs might not be met. Similar scenarios will be discussed later in this paper.

E. SoS MOE Evolution

Evolution is often considered as a major challenge in system-of-systems, given the heterogeneity of constituent systems, hyper-connectivity of systems involved, the emergent behavior and the evolutionary development processes [13]. Architecture evolution deals with changes to the static SoS architecture - for instance, changes to how the constituent systems are networked to each other. On the other hand, behavior evolution pertains to evolution in emergent behavior, based on dynamic SoS architecture - for instance, in terms of changes in set of resources and environment parameters. As the SoS evolves, there might be changes in the impact of an existing constituent systems' MOEs on the MOEs of the SoS, or on the MOEs of other constituent systems. Further, there might also be changes in the relationships of an existing constituent system's MOEs on the MOEs of the SoS, or on the MOEs of other constituent systems. Many other such inherent dynamics would play a role in the SoS evolution. While the MOE relationship matrix (Figure 3) would provide a good sense of the intertwining relationships from the functional and behavioral properties of the SoS, factoring in these additional constraints would be a challenge for complex SoS.

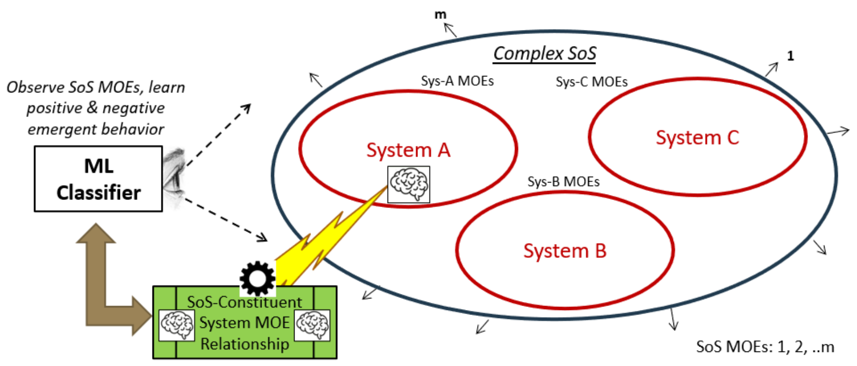

Complex SoS
Sys-A MOEs
Sys-C MOEs
Sys-B MOEs
System A
System C
System B
Observe SoS MOEs, learn positive & negative emergent behavior
ML Classifier
SoS-Constituent System MOE Relationship
SoS MOEs: 1, 2, ..m
m
1

Figure 4. Overview of Proposed Approach

F. Reinforcement Learning

Reinforcement learning is a type of machine learning approach that allows *agents* to automatically learn optimal control strategies through trial-and-error interactions with its environment [14]. As shown in Figure 5, a RL agent iteratively performs *actions* on the environment and in response, it receives the description of the environment (called *state*) and feedback (called *reward*) that indicates the impact of the action on the environment. While positive rewards indicate desired behaviour, negative rewards are penalties of bad actions. Based on the reward, the agent learns an optimal strategy or *policy* for choosing its next action that would receive higher reward. The training typically involves *exploration* in which random actions are selected, and *exploitation* which uses prior learned knowledge to select the best action. Striking a balance between exploration and exploitation is essential for maximizing rewards at minimal cost.

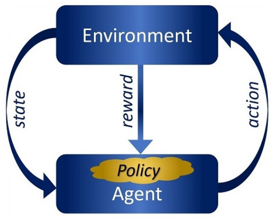


Figure 5. Reinforcement Learning

In the last decade, RL has become increasingly successful in solving complex systems in various fields such as robotics [15], gaming [16], and safety critical systems [17]-[19]. Typically, most of the existing RL literature concerns training single, or multiple agents that cooperate/compete [20][21] within the same environment, in which the agent observes a single scalar reward function and the goal is to find a policy that maximises the expected rewards [22]. However, since SoS is characterized by multiple constituent systems that have independent objectives and varying priorities, single agent or single objective training approaches are generally unsuitable in SoS context. Although there are theoretical discussions about multi-agent and multi-objective optimization approaches [23][24], to the best of our knowledge, the application of these techniques in a SoS context has not been previously explored. On the contrary, in this paper, we focus on training agents independently to achieve their own-goals, as well as the SoS-level goals by leveraging the relationships between SoS and constituent system MOEs for designing the rewards.

III. Proposed Framework

This section discusses the proposed framework towards application of reinforcement learning for constituent systems to learn the implications of its behaviors on the MOEs of the SoS and the emergent behavior witnessed at the SoS level.

A. Overview

Figure 4 provides an overview of the proposed approach. The complex SoS has a set of defined MOEs. The SoS comprises independent constituent systems, with each having their own corresponding system MOEs. A Machine Learning (ML) Classifier [25][26], that observes the various MOEs at SoS level and constituent system level and learns the positive and negative emergent behavior, is leveraged. In the proposed approach, the ML Classifier is used to advise the SoS-Constituent System MOE Relationship (SSMR) on positive and negative emergent behaviors. SSMR is built with the required intelligence to serve as the Environment (per Figure 5) for the constituent system, and provide the required feedback based on the positive and negative emergent behaviors witnessed at the SoS level. The constituent system is embedded with a reinforcement learning Agent (per Figure 5) to learn and evolve its behavior. Figure 6 provides details of the proposed approach.

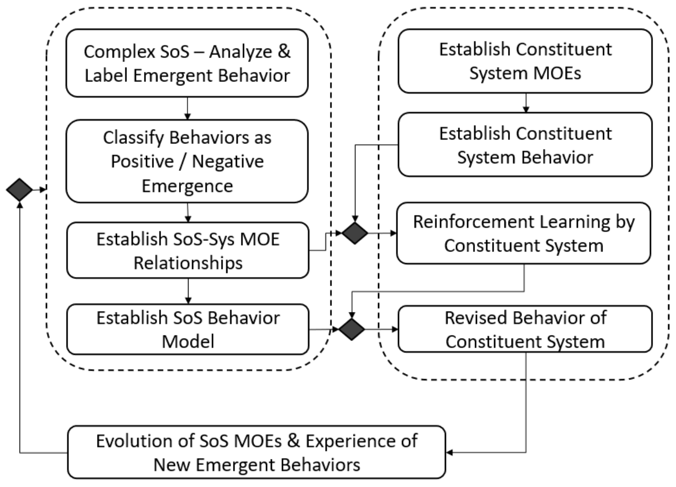
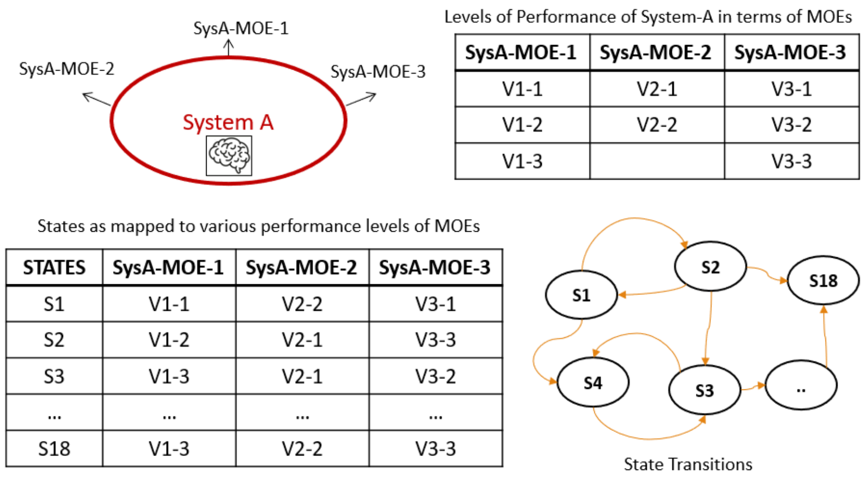

Complex SoS – Analyze & Label Emergent Behavior
Classify Behaviors as Positive / Negative Emergence
Establish SoS-Sys MOE Relationships
Establish SoS Behavior Model
Establish Constituent System MOEs
Establish Constituent System Behavior
Reinforcement Learning by Constituent System
Revised Behavior of Constituent System
Evolution of SoS MOEs & Experience of New Emergent Behaviors

Figure 6. Proposed Approach



Levels of Performance of System-A in terms of MOEs

SysA-MOE-1	SysA-MOE-2	SysA-MOE-3
V1-1	V2-1	V3-1
V1-2	V2-2	V3-2
V1-3		V3-3

States as mapped to various performance levels of MOEs

STATES	SysA-MOE-1	SysA-MOE-2	SysA-MOE-3
S1	V1-1	V2-2	V3-1
S2	V1-2	V2-1	V3-3
S3	V1-3	V2-1	V3-2
…	…	…	…
S18	V1-3	V2-2	V3-3

Figure 7. System-A: MOEs and State Transitions

B. *Framework Illustration*

Towards illustrating the adoption of proposed approach, the generic case of a constituent system, System-A, having a set of three MOEs: SysA-MOE-1, SysA-MOE-2 and SysA-MOE-3 is considered. The behavior exhibited by System-A is considered in terms of the specific levels of performance being exhibited in terms of its MOEs. The states of System-A, as mapped to the various performance levels of the MOEs are illustrated in Figure 7. For instance, System-A meets SysA-MOE-2 at two possible performance levels: V2-1 and V2-2. The figure also illustrates the various transitions permissible between the different states, as in state transition diagram. For instance, from state S2, the possible transitions are S1, S3 and S18. The System-A is a constituent system in SoSx. Let S^U denote the set of all systems. The definitions below illustrate the elements discussed (Z^+ is the set of positive integers) .

$$S^U = \{S_1, \ldots S_n\}, \text{ where } n \in Z^+ \tag{1}$$

$$SoSx \subset S^U; |SoSx| > 1 \tag{2}$$

$$MOE^{SoSx} = \{m_1^{SoSx}, \ldots m_x^{SoSx}\} \tag{3}$$

$$MOE^{S_A} = \{m_1^{S_A}, \ldots m_w^{S_A}\},\ S_A \in S^U \tag{4}$$

A subset of MOEs of S_A contribute towards a subset of the MOEs of SoSx, represented by $RMOE_{S_A}^{SoSx}$. It is to be noted that this relation has at least one element. Note that $RMOE_{S_A}^{SoSx}$ is defined iff $\exists\, m_w^{S_A} \in MOE^{S_A}$ contributing to $m_x^{SoSx} \in MOE^{SoSx}$.

$$Relation\ RMOE_{S_A}^{SoSx} = \{(m_w^{S_A}, m_x^{SoSx})\} \tag{5}$$

$$m_w^{S_A} \in MOE^{S_A} \text{ contributes to } m_x^{SoSx} \in MOE^{SoSx} \tag{6}$$

With various MOE performance levels being mapped to different states, the constituent system essentially has multiple means to transition from one state to another state. If given the narrow focus of achieving its own MOEs only, the system would perform at a level that maximizes its MOEs performance. In the proposed framework, the SSMR provides the required feedback to the constituent system on the impact of its MOE performance levels on the positive/ negative emergent behavior at the SoS level. This enables the constituent system to learn on the required MOE performance levels that balances its own mission along with the SoS objectives.

IV. Simulation & Experimental Results

In this section, we illustrate our approach using the Grid World example, in which the goal is to allow the agent learn the optimal set of transitions from an initial state to a terminal state. The Grid World serves as an abstract representation of a real world agent's state transition, where each state has implications both at system-level and SoS-level MOEs as described in the previous section. The goal for the agent is to navigate through states such that the SoS MOEs are met in addition to not unduly compromising the MOEs of the system.

A. *Experiment Details & Results*

We implemented our approach using MathWorks® MATLAB R2020b Reinforcement Learning toolbox [27]. For training, we used a Windows 10 OS with an Intel I-5 Core processor and 8 GB RAM.

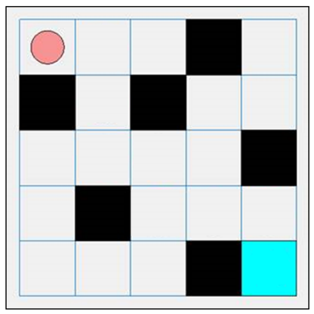


Figure 8. Grid World

We programmed a 2-dimensional 5 x 5 Grid World matrix [28], as shown in Figure 8. Each cell in the grid, that corresponds to each state, has an assigned reward value per the SSMR. The agent (visualized as a red circle in the Figure 8) begins from a programmed initial state and learns to traverse to the terminal state (colored blue). The agent has four possible actions in each grid state: west, north, south, and east. Further, certain states are defined as obstacles (black cells) to represent the infeasible states and transitions. The goal of the agent is to learn an optimal state transition through the grid, given the obstacles, such that the total rewards received is maximized. The parameters used for this training are shown in Table I.

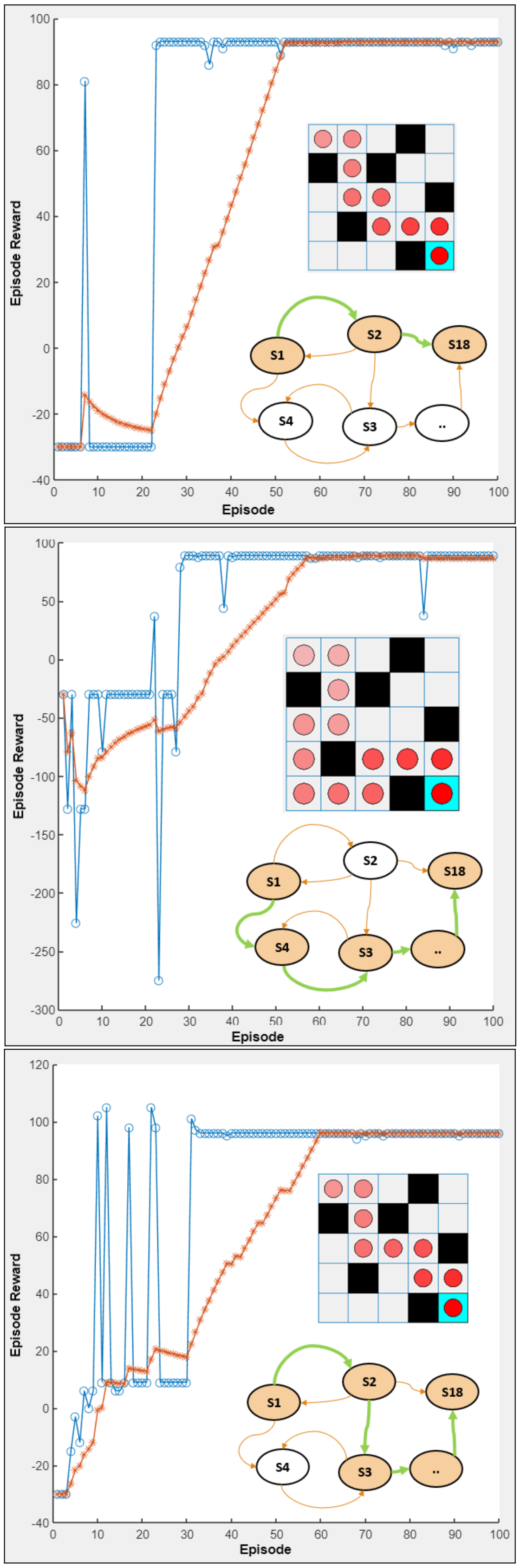


Figure 9. Training Visualization Plots and Learned Path

TABLE I. TRAINING PARAMETERS

Parameter Name	Description	Value
Epsilon	Probability threshold to select a random action or an existing action that maximizes the state-action value.	0.25 (Fig 9 a,b) 0.45 (Fig 9 c)
MaxEpisodes	Maximum number of episodes to train the agent.	100
MaxStepsPerEpisode	Maximum number of steps to run per episode	30
StopTrainingCriteria	Training termination condition	AverageReward
StopTrainingValue	Critical value of the training termination condition	100

First, to observe the learning in a non-SoS context, we trained the agent by assigning only system-level MOEs as rewards to each cell of the grid. We considered high MOE for the terminal state, a low positive reward for all intermediate states and a negative reward for the obstacles states. Figure 9(a) shows a plot of (i) rewards obtained by agent in each episode during the training (shown in blue line with circles),(ii) average rewards obtained after each training episode (shown in orange line with asterisks), (iii) the learned policy visualised in the grid environment, and (iv) the learned policy visualized in the state transition diagram (traversed states and transitions colored in beige and green respectively).

Next, to understand the emergent behaviours in a SoS context, we altered the reward for each state based on a predefined SSMR. We programmed the agent in a such a way that, at run-time if the learning context is set to SoS, it will adapt to learn a policy to traverse the grid to maximize the reward per the SSMR. For example, in one of the experiments, when an intermediate state ([3,3] in the grid) has low SoS-level MOE (i.e., negative SoS-level reward), the agent learnt to traverse avoiding that cell, as shown in Figure 9(b). Further, when we re-defined a higher SoS-level MOE for another intermediate state ([3,4] in the grid) in the SSMR arrangement and trained the agent again, it learnt another path that favors traversing that state, as shown in Figure 9(c).

The various paths learnt by the agent depending upon the SSMR, demostrates how the agent adapts to maximize SoS-level MOE. As one can observe, while the path taken by the agent could be sub-optimal at the system-level in some cases, it is optimal at the SoS-level, which is ultimately the desired behavior.

B. Limitations: Reward Hacking

While RL with appropriately designed rewards can help effectively train agents without expert supervision, it is not completely fool-proof. One of its well known failure modes is *reward hacking*, which happens when the agent attempts to learn to obtain high rewards in unexpected, rather undesired, ways. For example, when training the agent in the Grid World with higher SoS-level reward for a specific state transition (to [3,4] in Figure 9(c)), we found that the agent did not learn to reach the terminal state even after numerous training episodes. On examining the root cause, we found that the agent tried to accumulate rewards by just moving back and forth to accumulate more rewards, rather than moving past it to reach the terminal state with highest reward. While we addressed this problem in the case example by suitably adjusting the exploration factor as well as the SSMR values, we believe that RL designers of SoS should carefully evaluate the domain and design appropriate rewards to overcome this problem.

V. Conclusion

In this paper, we presented a novel approach towards using RL models and techniques for evolving MOEs of the constituent systems and SoS towards addressing emergent behavior. The proposed approach, through SoS-Constituent System MOE Relationship, enables constituent systems to learn and adapt their behaviors in tandem with the evolution of emergent behavior at SoS level. While the implementation and results described are specific to the case example considered, we believe that this serves as a promising proof of concept for a general, scalable and practical approach to train machine learning based systems in SoS context. In order to further realize the promise of this approach, we are currently working to make the approach applicable for dynamic MOE evolution scenarios in large complex systems and SoS. We are also exploring the use of deep RL approaches that incorporate deep neural networks for superior decision making capabilities and scalability.

References

[1] INCOSE, *Systems Engineering Handbook, 4th Edition*. INCOSE, 2015.

[2] M. Jamshidi, *Systems of Systems Engineering: Principles and Applications*. CRC Press, 2008.

[3] INCOSE, "International Council on Systems Engineering INSIGHT : Feature on System of Systems," vol. 19, pp. 3–78, Oct 2016.

[4] J. A. Lane and D. Epstein, "What is a system of systems and why should I care?," *University of Southern California*, 2013.

[5] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: Basic concepts, model-based techniques, and research directions," *ACM Computing Surveys*, vol. 48, pp. 18:1–18:41, Sept. 2015.

[6] R. Raman and M. D'Souza, "Decision learning framework for architecture design decisions of complex systems and system-of-systems," *Systems Engineering*, vol. 22, no. 6, pp. 538–560, 2019.

[7] W. Kinsner, "Complexity and its measures in cognitive and other complex systems," in *7th IEEE International Conference on Cognitive Informatics*, pp. 13–29, Aug 2008.

[8] J. Ladyman, J. Lambert, and K. Wiesner, "What is a complex system?," *European Journal for Philosophy of Science*, vol. 3, no. 1, pp. 33–67, 2013.

[9] M. Aiguier, P. L. Gall, and M. Mabrouki, "A formal definition of complex software," in *Proceedings of the 2008 3rd International Conference on Software Engineering Advances*, ICSEA '08, pp. 415–420, IEEE Computer Society, 2008.

[10] K. Giammarco, "Practical modeling concepts for engineering emergence in systems of systems," in *2017 12th System of Systems Engineering Conference (SoSE)*, pp. 1–6, June 2017.

[11] N. Smith and T. Clark, "A framework to model and measure system effectiveness," *11th ICCRTS Coalition Command and Control in the Network Era*, 2006.

[12] R. Raman and M. D'Souza, "Knowledge based decision model for architecting and evolving complex system-of-systems," in *Incose International Symposium*, vol. 27, pp. 30–44, Wiley Online Library, 2017.

[13] M. H. Fendali, D. Meslati, and I. Borne, "Understanding evolution in systems of systems," in *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–6, IEEE, 2017.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] J. Kober and J. Peters, *Reinforcement Learning in Robotics: A Survey*, pp. 9–67. Cham: Springer International Publishing, 2014.

[16] I. Szita, "Reinforcement learning in games," in *Reinforcement learning*, pp. 539–577, Springer, 2012.

[17] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Comput. Surv.*, vol. 50, pp. 1–38, June 2017.

[18] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial Intelligence in Medicine*, vol. 109, p. 101964, 2020.

[19] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *arXiv preprint arXiv:2002.00444*, 2020.

[20] R. Lowe et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[21] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, Apr. 2018.

[22] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[23] R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé, "Multi-objective multi-agent decision making: a utility-based analysis and survey," *Autonomous Agents and Multi-Agent Systems*, vol. 34, pp. 1–52, 2020.

[24] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, pp. 345–383, 2000.

[25] R. Raman and Y. Jeppu, "Formal validation of emergent behavior in a machine learning based collision avoidance system," in *IEEE International Systems Conference (SysCon)*, pp. 1–6, IEEE, 2020.

[26] R. Raman and Y. Jeppu, "An approach for formal verification of machine learning based complex systems," in *INCOSE International Symposium*, vol. 29, pp. 544–559, Wiley Online Library, 2019.

[27] "Reinforcement learning toolbox." https://www.mathworks.com/help/reinforcement-learning/index.html. [Accessed 1-Apr-2021].

[28] "Grid world." https://www.mathworks.com/help/reinforcement-learning/ref/creategridworld.html. [Accessed 1-Apr-2021].