Formal Modeling and Strategy Synthesis for Resource Optimization in System of Systems

Nazakat Ali, Muhammad Naeem, Julieth Patricia Castellanos-Ardila, and Sasikumar Punnekkat
School of Innovation, Design and Engineering
Mälardalen University - Västerås, Sweden
{nazakat.ali, muhammad.naeem, julieth.castellanos, sasikumar.punnekkat}@mdu.se

***Abstract*—Systems of Systems (SoS) face challenges related to coordinated management of the various tasks performed by constituent systems (CS), resource allocation, and SoS-level decision-making to achieve optimal performance related to costs and energy consumption. Addressing these challenges requires rigorous modeling and verification methods that accurately represent CS, capturing their interactions and synchronization. This becomes paramount as the complexity of the SoS grows with the increasing autonomy of individual CS. In this paper, we propose a methodology for efficient resource optimization in battery-powered autonomous CS within an SoS. We begin by modeling the CS and their respective controllers within a constellation of an SoS using UPPAAL STRATEGO. Then, we synthesize a strategy for mission planning and efficient resource utilization to achieve the mission. We also apply our proposed approach to an industrial case study focused on mass removal in the construction site modeled as an SoS to validate our proposed approach. The simulation results show that the synthesized strategies significantly improved resource optimization and reduced mission completion times compared to the ones without the synthesized strategies. Our approach, based on a synergetic combination of formal model modeling and reinforcement learning, provides a viable approach to achieve efficiency in SoS contexts.**

***Index Terms*—Formal modeling, System of Systems, resource optimization.**

I. INTRODUCTION

A system of systems (SoS) comprises multiple independent systems, known as constituent systems (CS) that interact to achieve a common mission [1]. For instance, in the construction domain, an SoS of autonomous CS can reduce costs and enhance productivity by minimizing human intervention. However, their high variability [2], such as differing levels of autonomy, presents significant challenges in mission execution, resource optimization, and dynamic task management. In SoS, the inherent complexities arise from distributed decision-making, dynamic task allocation, and potential conflicts (e.g., timing inconsistencies and resource contention) between CS. To address these challenges, formal methods provide a rigorous foundation for modeling, analyzing, and verifying the behavior of these systems. Formal methods ensure that SoS satisfies critical safety and timing properties while maintaining operational efficiency [3]. They also enable optimization of resource allocation and decision-making.

This work is supported by the SIMCON project funded by Vinnova and the DAISY project funded by SSF.

Recent advances in models and strategies for collaborative systems have been particularly significant, especially in collaborative and probabilistic approaches [4]. Fraser et al. [5] developed a concrete simulation model for Unmanned Aerial Vehicle (UAV) missions and used the PRISM model checker for probabilistic verification. Experimental results demonstrate the effectiveness of the synthesized strategies, highlighting their practical applicability. Ali et al. [3] proposed an approach using colored Petri nets to model collaborative systems. However, their focus was primarily on fail-safe and fail-operational aspects rather than resource optimization. In general, various formal modeling tools are available [6], but UPPAAL [7] stands out as a powerful model-checking framework for verifying real-time systems [8]. Its capability to model timed automata and analyze system behaviors under temporal constraints makes it particularly well-suited for modeling safety-critical SoS (see, for example, the work done in [9], [10]).

The collaborative approach to model development and strategy synthesis is particularly relevant in SoS contexts, where multiple autonomous CS must coordinate. However, a key challenge is productivity, especially in SoS consisting of battery-powered CS, where limited operating range and duration directly impact efficiency [11]. Thus, efficient resource planning is necessary to increase productivity. In this regard, Q-learning [12], a model-free reinforcement learning algorithm, is widely used for mission planning and resource optimization due to its ability to learn optimal strategies without requiring prior knowledge of the environment [13].

In this paper, we propose a methodology for efficient resource optimization in battery-powered autonomous CS within an SoS. First, we model the CS and their respective controllers within an SoS constellation using UPPAAL STRATEGO. Next, we synthesize a strategy for mission planning and resource optimization using Q-learning, a built-in feature of UPPAAL STRATEGO, to achieve the desired goal. To validate our methodology, we apply it to an industrial case study focused on mass removal in the construction domain. The simulation results demonstrate that the synthesized strategy significantly improves resource optimization and reduces mission completion times compared to scenarios without it.

This paper is structured as follows. Section II introduces the necessary preliminaries. Section III presents the proposed approach. Section IV describes the use case. Section V presents the SoS models in UPPAAL. Section VI discusses the simulation results. Finally, Section VII concludes the paper.

II. Preliminaries

A. System of Systems (SoS)

A SoS [1] is a collection of constituent systems (CS) that, by forming constellations, collaborate to achieve common missions. A constellation is a subset of CS linked together to exchange information and provide a specific capability [14]. Unlike single systems, SoS exhibits emergent behaviors, i.e., the ability to deliver new functionalities arising from the real-time collaboration of its CS [15]. However, this behavior can also lead to unexpected situations, even if the individual CS are well understood [16]. For example, during a ground collapse on a construction site, autonomous machines may attempt uncoordinated evasive (and unsafe) actions based on individual hazard avoidance algorithms.

CSs are also characterized by their operational and managerial independence. This independence promotes operational self-sufficiency, a diverse range of stakeholders, and a distributed distribution. Moreover, an SoS commonly evolves over time as new CS are added, existing CS are upgraded, or decommissioned. Different arrangements, called topologies, have been identified to facilitate CS collaboration in an SoS. In particular, Maier [17] describes three of them, i.e., directed SoS, where all CS operate under the direct control of a central authority imposing goals, and collaborative and virtual SoS, which loosely collaborate. Dahmann and Baldwin [18] also included the acknowledged SoS topology, which is an arrangement of CS with central authority as the directed SoS. However, such CS can influence general SoS objectives.

B. Modelling in Uppaal Stratego

Uppaal Stratego is a tool for modeling and strategy synthesis in stochastic hybrid games (SHGs) [8]. It allows the representation of systems as networks of timed automata, where states are represented by locations and system evolution is defined by transitions. A location represents a distinct state of the system, while transitions define how the system moves between locations. Each transition can have guards, which are conditions that must be met for the transition to occur. Additionally, invariants impose constraints on how long the system can remain in a particular location. Uppaal Stratego distinguishes between controllable (solid) transitions, which are decisions made by the system, and uncontrollable (doted) transitions, which represent environmental uncertainties.

A SHG is a two-player game played on a stochastic hybrid automaton, where one player represents the system and the other represents the environment [13]. The system aims to synthesize an optimal strategy to reach a goal while accounting for uncertainties. For example, consider a vehicle routing scenario where a vehicle must travel from a start location to an end location (see Fig. 1). The driver chooses between high roads and low roads, each with different travel times and fuel consumption. The SHG model includes locations representing different waypoints, transitions indicating road choices, and clock constraints defining travel time. Controllable transitions allow the driver to choose roads, while uncontrollable transitions model unpredictable travel times due to varying traffic conditions. The objective is to find a strategy that minimizes fuel consumption while ensuring the vehicle reaches its destination within a given time limit.

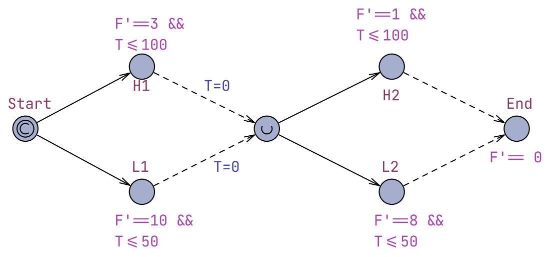


Fig. 1. Stochastic hybrid automaton for a vehicle [19]

In Uppaal Stratego, a strategy can be synthesized to guide the system player (e.g., CS) towards its goal using the following query:

$$\texttt{strategy s} = \texttt{minE(cost)}\,[\leq \texttt{T}]\,\{\texttt{ExpList1}\} \;\rightarrow \{\texttt{ExpList2}\} :<> \texttt{goal} \quad (1)$$

In this query, `minE(exp)` indicates that the strategy `s` aims to minimize the value of the expression `cost` within `T` time units or until the goal predicate becomes true. `ExpList1` and `ExpList2` are lists of observable state expressions utilized in Q-learning. `ExpList1` consists of discrete variables, while `ExpList2` contains continuous variables. Uppaal Stratego simulates the model and generates trace samples for the learning algorithm, with the <> goal specifying the criteria for selecting these traces. The learning algorithm receives controllable actions from these traces, which represent partial observations of the state space.

III. Proposed Approach

A SoS is a complex, dynamic collection of independent CS that exhibit emergent behaviors, enabling new capabilities or potentially leading to unexpected outcomes, while evolving over time and operating under different topological arrangements (see Section II-A). To better navigate this complexity in terms of efficiency, we introduce a methodology based on Uppaal Stratego (see Section II-B), that permits planning and optimizing resources in an SoS that needs to achieve a common goal in a minimal period of time.

We model the SoS in two layers as shown in Fig.2. First, we model controllers in the orchestrator that assign tasks to CS of respective constellations and control their task execution. The controllers in the orchestrators are dynamically created based on the number of constellations determined to achieve the common goal (In this paper, we assumed one constellation). Secondly, we modeled the potential CS (refer to the case study) for the assumed constellation. After SoS modeling, we synthesize a strategy to achieve the goal in an optimal time. The Strategy Synthesis module uses Q-learning, a machine

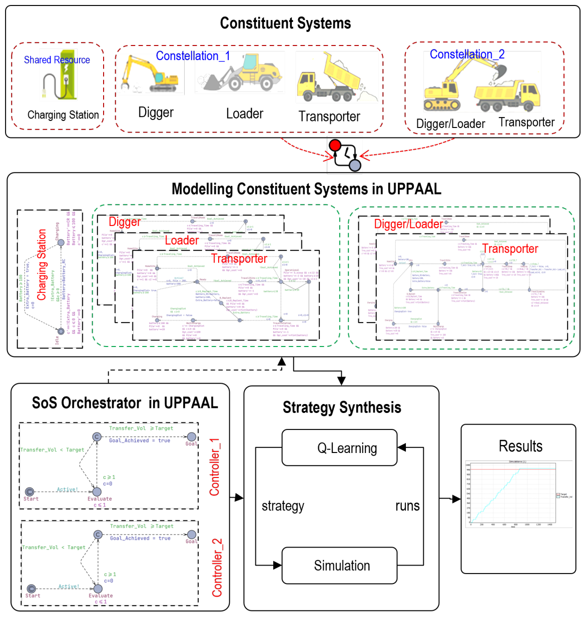


Fig. 2. Proposed Approach

learning technique, along with simulation to optimize strategies for improved system performance. The strategy synthesis module initially explores the state space of the SoS model by conducting random simulations and sampling multiple runs. These runs go as input for the Q-Learning algorithm to determine the best strategy.

IV. Use Case Description

Construction sites are heterogeneous environments with multiple machines and humans collaborating on tasks in an ad-hoc manner. Construction tasks are highly interdependent yet often managed with operational and managerial autonomy, requiring careful coordination to maximize productivity. Typically, site orchestrations are done manually in spite of the growing digitalization trends. There is growing interest in electrification and usage of semi-autonomous machinery collaborating over a digital infrastructure. Due to the interdependency of different systems managing these tasks, there is a need for an unambiguous interpretation of tasks, their precedences, preconditions, and synchronization. From a safety perspective, avoiding cascading failures or missing deadlines during various connected tasks is also paramount.

In this section, we present an industrial use case, that is, mass removal on a construction site, as illustrated in the middle of Fig. 2 (Constellation_1). The use case consists of multiple CS, including human operators, autonomous digging machines, loaders, transporters, and other relevant actors, all collaborating within a constellation to accomplish a mission: to transport material from the digging site to the dumping site. The orchestrator is responsible for receiving the mission and determining the number of required constellations [20]. In our scenario, the orchestrator is assigned the task of transferring 1,000 tons of material. To achieve this, a single constellation is deployed comprising a digger, a loader, and a transporter. The task execution controller (`Executor`) within the orchestrator assigns specific tasks to each CS and monitors their progress. Upon receiving commands, the digger moves to the designated location, begins excavation, and makes a pile. The loader then loads the excavated material to the transporter, which then transports it to the designated dumping site. This cycle repeats until the mission is completed. During operation, machines must be returned for charging if their battery level falls below the minimum threshold. Once the mission is accomplished, the task execution controller issues a stop command to each CS, bringing the operation to a halt.

V. Modeling in Uppaal Stratego

To ensure the efficient execution of construction tasks within an SoS, we developed a formal model using Uppaal Stratego. The model includes autonomous `Digger`, `Loader`, `Transporter`, `Charging_System`, and `Executor`, all working collaboratively to complete the mass removal operation. The design incorporates real-time constraints, energy management, and task synchronization to optimize resource utilization. Each CS is represented as a timed automaton, with defined locations, transitions, guards, and invariants that regulate system behavior. Below, we describe the modeling and behavior of each CS model.

A. Digger Model

The model for `Digger` is shown in Fig. 3. This model represents an autonomous `Digger` operating within a construction system, which captures its workflow from travel to and from the excavation site, digging process and, charging management. The digger is considered to be at `HomeSite` location, where key parameters such as `Pile`, `battery`, and `Dgr_cost` are initialized.

When `Digger` receives an activation signal from the `Executer`, it proceeds to the `Travel2Site` location for traveling to the excavation site while consuming power at the rate of $\texttt{Battery}' = -PCT$. `Battery` is a clock variable that evolves with the rate of PCT (power consumption on the travel). Upon arrival at the excavation site, the `Digger` enters the `Operational` location and starts digging. At this location, the model evolves a clock variable `Pile` with the rate of scope volume (`Pile' = D_scoup`) and `Battery` clock with the rate of `-PCD` (power consumption on the digging). The excavation process continues while the `Digger` monitors its battery level. If the battery level falls below 20 percent, the digger travels to the charging station through `Travel2station` location. The digger waits at `Wait2Charge` location if the `ChargingSlot` is already occupied. When the `ChargingSlot` becomes true, it goes for `Charging` and remains there until fully recharged (`Battery` $\geq$ 100). Once charged, it returns to the `Ready` state and begins a new task cycle if the goal is not achieved. We introduce flexibility in the charging mechanism by adding controllable (learning) transitions. Through this the `Digger` can choose whether to continue digging at the excavation site or go early to charge to avoid waiting at the `Charging` location. Another feature introduced in this model

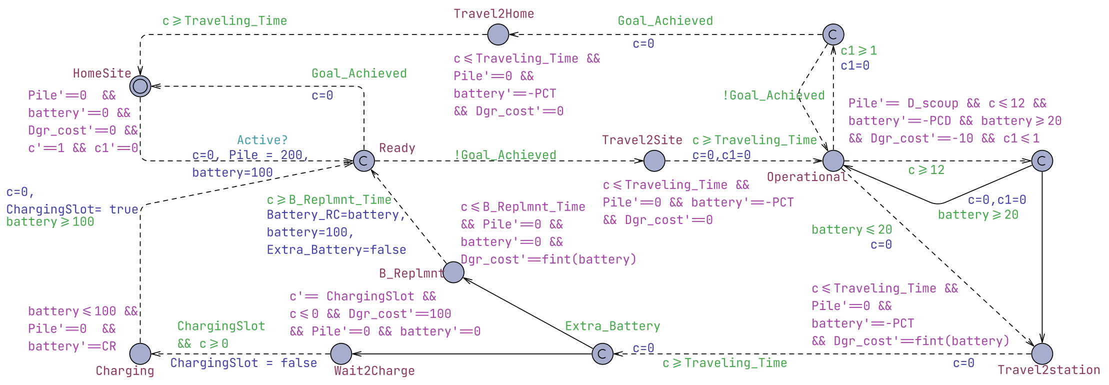

c ⩾ Traveling_Time
Travel2Home
Goal_Achieved
c=0
c ⩽ Traveling_Time && Pile'==0 && battery'==-PCT && Dgr_cost'==0
HomeSite
Pile'==0 && battery'==0 && Dgr_cost'==0 && c'==1 && c1'==0
Goal_Achieved
c=0
C
c1 ⩾ 1
c1=0
!Goal_Achieved
Pile'== D_scoup && c ⩽ 12 && battery'==-PCD && battery ⩾ 20 && Dgr_cost'==-10 && c1 ⩽ 1
Active?
c=0, Pile = 200, battery=100
Ready
!Goal_Achieved
Travel2Site
c ⩾ Traveling_Time
c=0,c1=0
Operational
c ⩾ 12
c=0,c1=0
battery ⩾ 20
c=0, ChargingSlot= true
battery ⩾ 100
c ⩾ B_Replmnt_Time
Battery_RC=battery, battery=100, Extra_Battery=false
c ⩽ B_Replmnt_Time && Pile'==0 && battery'==0 && Dgr_cost'==fint(battery)
c ⩽ Traveling_Time && Pile'==0 && battery'==-PCT && Dgr_cost'==0
battery ⩽ 20
c=0
B_Replmnt
c'== ChargingSlot && c ⩽ 0 && Dgr_cost'==100 && Pile'==0 && battery'==0
c ⩽ Traveling_Time && Pile'==0 && battery'==-PCT && Dgr_cost'==fint(battery)
battery ⩽ 100 && Pile'==0 && battery'==CR
ChargingSlot && c ⩾ 0
Extra_Battery
c=0
Charging
ChargingSlot = false
Wait2Charge
c ⩾ Traveling_Time
c=0
Travel2station

Fig. 3. UPPAAL Model for Digger Machine

is the choice-based charging mechanism, where a Loader can either charge its battery at a station or opt for a faster battery replacement if an extra battery is available. The model controls this selection through the controllable transitions. That allows dynamic adaptation based on extra battery availability. Lastly, the Digger returns to HomeSite if the task is finished (Goal_Achieved) through the Travel2Home location.

B. Loader Model

The model for Loader is illustrated in Fig. 4. This model depicts an autonomous Loader responsible for loading material from the excavated pile to the Transporter. The Loader includes a clock variable (Battery) to represent power consumption, which varies at different rates depending on the actions being performed. Similar to the Digger, the Loader also starts from the HomeSite and proceeds to the Loading location through the Travel2Site when received activation signal from Executor. Upon arrival at the excavation site, the Loader enters the Wait location, where it checks if there is enough Pile ($\texttt{Pile} \geq 2 * \texttt{L\_scoup}$) and the transport vehicle is available. If these conditions hold, it proceeds to Loading location and starts loading material to the Transporter (Load_Vol' == L_scoup). It also updates the Pile volume. The process continues until the Transporter reaches its maximum load capacity (Load_vol >= Max_LV). The charging mechanism of Loader is also similar to the Digger and it returns to HomeSite if the task is finished (Goal_Achieved) through the location Travel2Home.

C. Transporter Model

The model for Transporter is shown in Fig. 5. This model shows an autonomous Transporter responsible for moving material from the excavation site to the dumping site. The Transporter also includes a clock variable (Battery) to represent power consumption, which varies at different rates depending on the actions being performed. Similar to the Digger and Loader, the Transporter also starts from the HomeSite and proceeds to the Loading location through the Travel2Site when received activation signal from Executor. When it reaches the Loading location, it enables a flag that is T_Avl to inform Loader model that the transport vehicle is available. The loading process continues until the load volume reaches capacity ($\texttt{Load\_vol} = \texttt{Max\_LV}$). After that the Transporter travels to Dumping through the Travel2Dump location where it unloads the material ($\texttt{Load\_vol} = 0$) within Dump_T time units. After that, it travels back to the Loading through the Travel2Loading for another cycle if it has enough battery level for the next cycle. Otherwise it will go to charging. The charging mechanism of texttTransporter is also similar to the Digger and it returns to HomeSite if the task is finished (Goal_Achieved) through the Travel2Home location.

D. Charging System Model

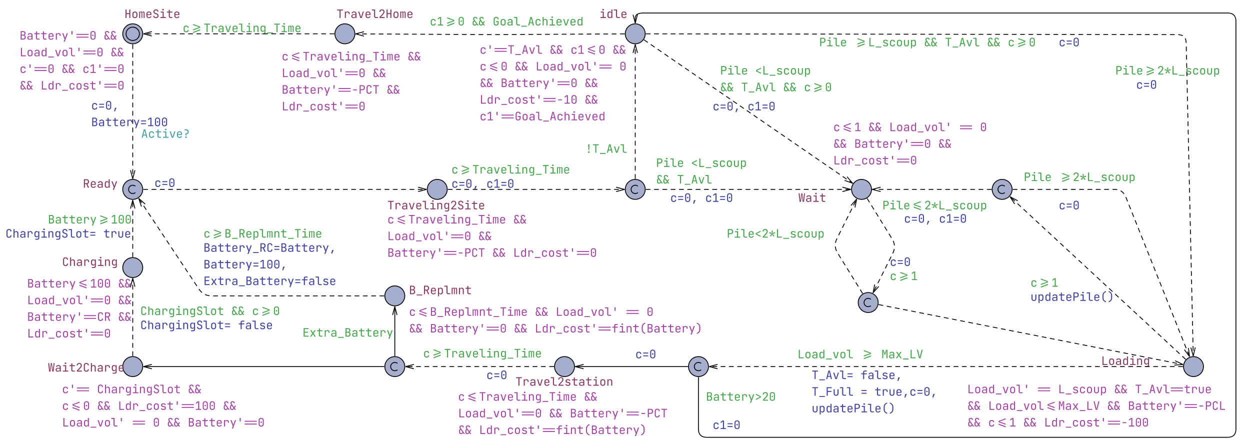
The model for the Charging System is designed to manage the recharging of vehicle batteries and extra batteries to ensure continuous operation of all vehicles. When a vehicle opts for battery replacement, it takes a fully charged extra battery and leaves its used battery at the charging station. The model then initiates the charging process for the used battery. Once fully charged, the model enables the Extra_Battery flag, signaling that a replacement battery is now available for the next vehicle in need.

E. Executor Model

The model for Executor is shown in Fig. 6. The executor acts as the decision-making orchestrator in the SoS. It starts in Start and sends an activation signal to all vehicles by synchronizing with Active! channel. The executor then goes to the Evaluate location, where it continuously monitors the progress of the CSs. It sends a Goal_Achieved signal to CSs to terminate the activity, when the goal is achieved. The model ensures that excavation, loading, transportation, and charging processes are executed efficiently and synchronously.

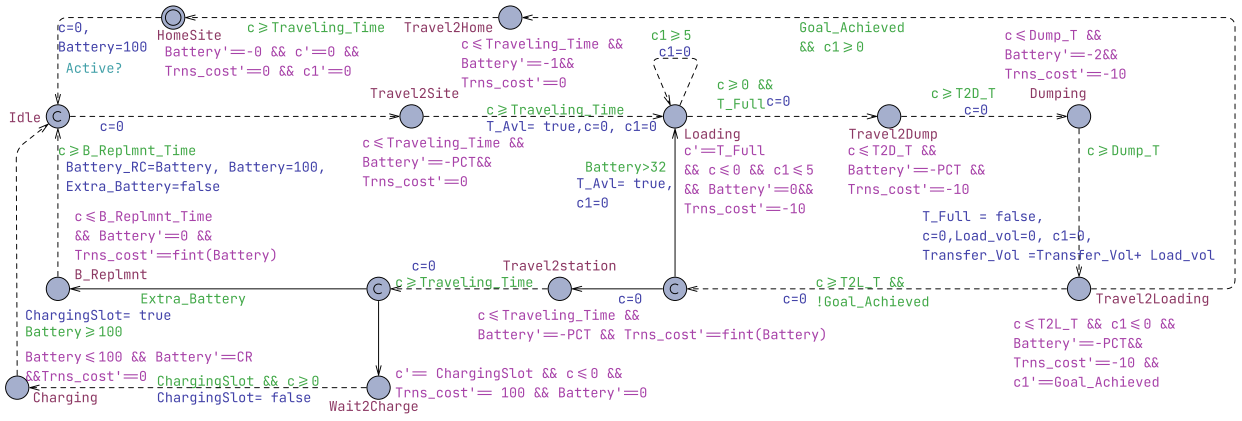
VI. Simulation Results and Discussion

Following the strategy synthesis query stature in Section II, we construct the following strategy as shown in the box. Here, T_Cost is the objective function, and the goal of synthesis is to minimize T_Cost. The query passes the set of discrete and



HomeSite
Travel2Home
idle
c≥Traveling_Time
c1≥0 && Goal_Achieved
Battery'==0 && Load_vol'==0 && c'==0 && c1'==0 && Ldr_cost'==0
c≤Traveling_Time && Load_vol'==0 && Battery'==-PCT && Ldr_cost'==0
c'=T_Avl && c1≤0 && c≤0 && Load_vol'== 0 && Battery'==0 && Ldr_cost'==-10 && c1'==Goal_Achieved
Pile ≥L_scoup && T_Avl && c≥0
c=0
Pile <L_scoup && T_Avl && c≥0
c=0, c1=0
Pile≥2*L_scoup
c=0
c=0, Battery=100
Active?
!T_Avl
c≤1 && Load_vol' == 0 && Battery'==0 && Ldr_cost'==0
Ready
c=0
c≥Traveling_Time
c=0, c1=0
Pile <L_scoup && T_Avl
c=0, c1=0
Wait
Pile ≥2*L_scoup
c=0
Traveling2Site
c≤Traveling_Time && Load_vol'==0 && Battery'==-PCT && Ldr_cost'==0
Pile≤2*L_scoup
c=0, c1=0
Pile<2*L_scoup
Battery≥100
ChargingSlot= true
c≥B_Replmnt_Time
Battery_RC=Battery, Battery=100, Extra_Battery=false
c=0
c≥1
c≥1
updatePile()
Charging
Battery≤100 && Load_vol'==0 && Battery'==CR && Ldr_cost'==0
B_Replmnt
c≤B_Replmnt_Time && Load_vol' == 0 && Battery'==0 && Ldr_cost'==fint(Battery)
ChargingSlot && c≥0
ChargingSlot= false
Extra_Battery
c≥Traveling_Time
c=0
Load_vol ≥ Max_LV
T_Avl= false, T_Full = true,c=0, updatePile()
Loading
Wait2Charge
c=0
Travel2station
c≤Traveling_Time && Load_vol'==0 && Battery'==-PCT && Ldr_cost'==fint(Battery)
Battery>20
c1=0
Load_vol' == L_scoup && T_Avl==true && Load_vol≤Max_LV && Battery'==-PCL && c≤1 && Ldr_cost'==-100
c'== ChargingSlot && c≤0 && Ldr_cost'==100 && Load_vol' == 0 && Battery'==0

Fig. 4. UPPAAL Model for Loader Machine



c=0, Battery=100
Active?
HomeSite
c≥Traveling_Time
Battery'==-0 && c'==0 && Trns_cost'==0 && c1'==0
Travel2Home
c≤Traveling_Time && Battery'==-1&& Trns_cost'==0
c1≥5
c1=0
Goal_Achieved && c1≥0
c≤Dump_T && Battery'==-2&& Trns_cost'==-10
Dumping
Travel2Site
c≥0 && T_Full
c=0
c≥T2D_T
c=0
Idle
c=0
c≥Traveling_Time
T_Avl= true,c=0, c1=0
Loading
c'==T_Full && c≤0 && c1≤5 && Battery'==0&& Trns_cost'==-10
Travel2Dump
c≤T2D_T && Battery'==-PCT && Trns_cost'==-10
c≥Dump_T
c≥B_Replmnt_Time
Battery_RC=Battery, Battery=100, Extra_Battery=false
c≤Traveling_Time && Battery'==-PCT&& Trns_cost'==0
Battery>32
T_Avl= true,
c1=0
T_Full = false, c=0,Load_vol=0, c1=0, Transfer_Vol =Transfer_Vol+ Load_vol
c≤B_Replmnt_Time && Battery'==0 && Trns_cost'=fint(Battery)
B_Replmnt
c=0
Travel2station
c≥Traveling_Time
c=0
c≥T2L_T &&
c=0
!Goal_Achieved
Travel2Loading
Extra_Battery
ChargingSlot= true
Battery≥100
c≤Traveling_Time && Battery'==-PCT && Trns_cost'=fint(Battery)
c≤T2L_T && c1≤0 && Battery'==-PCT&& Trns_cost'==-10 && c1'==Goal_Achieved
Battery≤100 && Battery'==CR &&Trns_cost'==0
Charging
ChargingSlot && c≥0
ChargingSlot= false
Wait2Charge
c'== ChargingSlot && c≤0 && Trns_cost'== 100 && Battery'==0

Fig. 5. UPPAAL Model for Transporter Machine

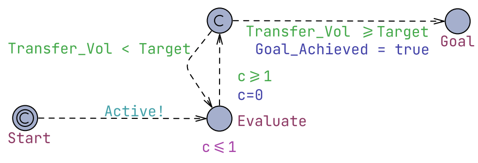


Fig. 6. UPPAAL Model for Task Execution Controller

continuous variables to the Q-learning algorithm embedded in Uppaal Stratego to develop a strategy.

After making the strategy, we run the following simulation query to simulate the model under the synthesized strategy (equation 2). The simulation query prints the `Target` and `Transfer_Vol` representing the target mission and the transferred volume in a given time. Fig.7 shows our simulation results. Initially, the material transfer process, represented by the light blue curve, took 1,455 time units to reach the target. After implementing the optimized strategy, shown by the dark blue curve, the process was completed in just 915 time units. This makes a 37 percent reduction in time. Such improvement indicates that the learning-based approach enhanced resource optimization, and coordination and minimized idle and waiting time. The stepwise nature of the curves indicates batch transfers, and the steeper incline in the optimized approach highlights a faster transfer rate with fewer delays. The improvement could be attributed to better resource optimization, reduced waiting times, and a more effective workflow for the `Digger`, `Loader`, and `Transporter`.

$$\begin{aligned}&\texttt{strategyOptStrategy} = \min E(\texttt{T\_cost})[T \leq 5000]\{\texttt{Digg.location},\\ &\texttt{Loader.location}, \texttt{Transporter.location}, \texttt{Executor.location},\\ &\texttt{ChargingSlot}, \texttt{Transfer\_Vol}, \texttt{Target}, \texttt{Load\_vol}, \texttt{Goal\_Achieved},\\ &\texttt{Extra\_Battery}, \texttt{T\_Avl}, \texttt{T\_Full}, \texttt{T\_Capacity\_F}\} \rightarrow \{\texttt{Dgr\_cost}\\ &\texttt{T\_cost}, \texttt{Ldr\_cost}, \texttt{Trns\_cost}, \texttt{Digg.battery}, \texttt{Loader.Battery},\\ &\texttt{T}, \texttt{Pile}, \texttt{Loader.Battery}, \texttt{Transporter.Battery}\} :<>\\ &\qquad \texttt{Executor.Goal} \; \&\& \; \texttt{T} \leq 5000\end{aligned}$$

$$\texttt{simulate}\ [\leq \texttt{T}; 1]\{\texttt{Target}, \texttt{Transfer\_Vol}\}\ \texttt{under}\ \texttt{OptStrategy} \tag{2}$$

The waiting times for machines, as shown in Fig. 8 (right side), further support the improvements observed in the material transfer process. In Fig. 8 (left side), which represents the system before applying the learning-based strategy synthesis, the transporter, digger, and loader experienced frequent and evenly distributed waiting periods. The presence of multiple vertical lines indicates recurring idle or waiting times, suggesting inefficiencies in scheduling and coordination among the CS. In contrast, the right side side of Fig. 8, illustrates a significant reduction in these waiting instances after the optimized strategy was implemented, especially for the `Digger` and `Loader`. Although the `Transporter` still encounters some idle periods, the overall waiting time across the system has decreased. This improvement has resulted in better synchronization among excavation, loading, transportation, and charging processes. This reduction in waiting time directly correlates with the decrease in total mission completion time, which hints at improved system-wide resource utilization and coordination.

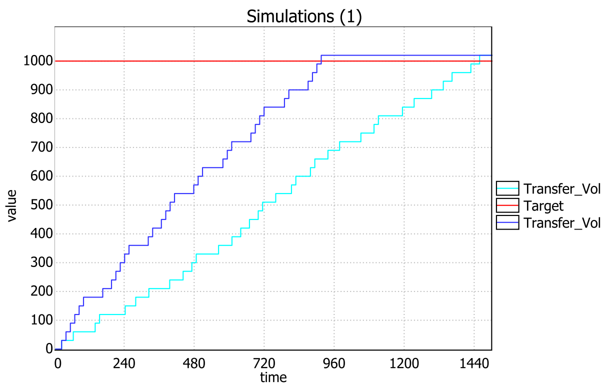


Fig. 7. Simulation Result: with and without strategy

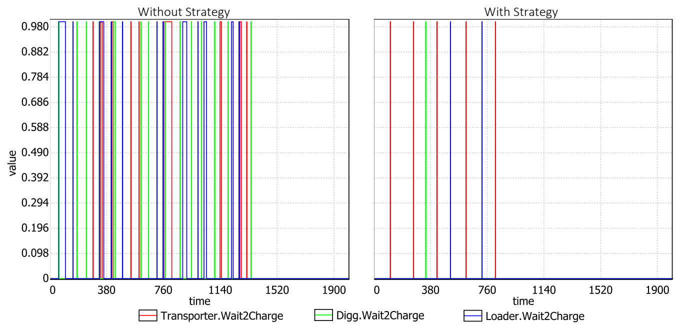


Fig. 8. Waiting time of Loader, Transporter, and Digger

VII. Conclusion

In this paper, we introduced a formal modeling and strategy synthesis approach aimed at optimizing resources and enhancing operational efficiency in autonomous SoS within the construction domain. Using Uppaal Stratego, we modeled construction systems, including diggers, loaders, transporters, and charging systems. We applied Q-learning—a built-in function in Uppaal Stratego-for strategy synthesis, which resulted in a significant reduction of approximately 37 percent in mission completion time. This improvement was primarily achieved through optimized resource allocation and reduced vehicle waiting times.

Future research may explore extending the framework to multiple constellations with more complex dependencies and further validate the methodology in different domains.

References

[1] ISO/IEC JTC 1/SC 7, *ISO/IEC/IEEE 21841:2019. Systems and Software Engineering — Taxonomy of System of Systems*, Std., 2019. [Online]. Available: https://www.iso.org/standard/71957.html

[2] J. P. Castellanos-Ardila, N. Ali, S. Punnekkat, and J. Axelsson, "Making systems of systems orchestrations safer," in *European Safety and Reliability (ESREL) and Society for Risk Analysis Europe (SRA-E)*, 2025.

[3] N. Ali, S. Punnekkat, and A. Rauf, "Modeling and safety analysis for collaborative safety-critical systems using hierarchical colored petri nets," *Journal of Systems and Software*, vol. 210, p. 111958, 2024.

[4] X. Yin, B. Gao, and X. Yu, "Formal synthesis of controllers for safety-critical autonomous systems: Developments and challenges," *Annual Reviews in Control*, vol. 57, p. 100940, 2024.

[5] D. Fraser, R. Giaquinta, R. Hoffmann, M. Ireland, A. Miller, and G. Norman, "Collaborative models for autonomous systems controller synthesis," *Formal Aspects of Computing*, vol. 32, pp. 157–186, 2020.

[6] R. C. Armstrong, R. J. Punnoose, M. H. Wong, and J. R. Mayo, "Survey of existing tools for formal verification," Sandia National Lab.(SNL-CA), Livermore, CA (United States), Tech. Rep., 2014.

[7] Uppaal Team, "Uppaal: Model Checking and Validation Tool," 2025, accessed: 2025-03-12. [Online]. Available: https://uppaal.org/

[8] A. David, P. G. Jensen, K. G. Larsen, M. Mikučionis, and J. H. Taankvist, "Uppaal stratego," in *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*. Springer, 2015, pp. 206–211.

[9] M. U. Iftikhar and D. Weyns, "A case study on formal verification of self-adaptive behaviors in a decentralized system," *arXiv preprint arXiv:1208.4635*, 2012.

[10] R. Gu, K. Tan, A. H. Høeg-Petersen, L. Feng, and K. G. Larsen, "Commonupproad: A framework of formal modelling, verifying, learning, and visualisation of autonomous vehicles," in *International Symposium on Leveraging Applications of Formal Methods*. Springer, 2024.

[11] M. Naeem, "Energy-efficient cyber-physical systems: A model-based approach," Ph.D. dissertation, Aalborg University Open Publishing, Denmark, 2024.

[12] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[13] M. Naeem, R. Gu, C. Seceleanu, K. Guldstrand Larsen, B. Nielsen, and M. Albano, "Energy-efficient motion planning for autonomous vehicles using uppaal stratego," in *International Symposium on Theoretical Aspects of Software Engineering*. Springer, 2024, pp. 356–373.

[14] J. Axelsson, "A refined terminology on system-of-systems substructure and constituent system states," in *2019 14th Annual Conference System of Systems Engineering (SoSE)*. IEEE, 2019, pp. 31–36.

[15] T. J. Inocêncio, G. R. Gonzales, E. Cavalcante, and F. E. Horita, "Emergent behavior in system-of-systems: A systematic mapping study," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 2019, pp. 140–149.

[16] S. C. Beland and A. Miller, "Assuring a complex safety-critical systems of systems," *SAE Transactions*, pp. 974–988, 2007.

[17] M. W. Maier, "Architecting Principles for Systems-of-Systems," *The Journal of the International Council on Systems Engineering*, vol. 1, no. 4, 1998.

[18] J. S. Dahmann and K. J. Baldwin, "Understanding the current state of us defense systems of systems and the implications for systems engineering," in *2nd Annual IEEE Systems Conference*, 2008.

[19] M. Naeem, M. Albano, K. G. Larsen, B. Nielsen, A. Høedholt, and C. Ø. Laursen, "Modelling and analysis of a sigfox-based iot network using uppaalsmc," *IEEE Sensors Journal*, vol. 23, no. 10, 2023.

[20] T. Nordstrom, L. R. Sutfeld, and T. Besker, "Exploring different actor roles in orchestrations of system of systems," in *2024 19th Annual System of Systems Engineering Conference (SoSE)*. IEEE, 2024, pp. 190–196.