Ontology-based Adaptive Reward Functions

Saeedeh Ghanadbashi[1], Akram Zarchini[2] and Fatemeh Golpayegani[1]

[1]School of Computer Science, University College Dublin, Ireland
[2]Department of Computer Engineering, Sharif University of Technology, Iran

Abstract
Reinforcement Learning (RL) is a learning approach where agents receive feedback in the form of a reward function from the environment, allowing them to learn through trial and error. In dynamic environments with unexpected events, there is often a need to design new or adaptive reward functions to dynamically adapt the behavior based on the changing dynamics of the environment. Current methods for specifying reward functions are limited to manual reward function definition or extracting/inferencing from human demonstrations. On the other hand, ontologies with the ability to provide a structured representation and organize concepts and properties hierarchically, facilitate a deeper understanding of the environment, empowering agents to identify and comprehend new events.

This paper presents a new Ontology-based Adaptive Reward Function (OARF) method, which dynamically creates new reward functions based on domain ontologies. The OARF method is evaluated in a job shop scheduling environment, demonstrating its superiority over a state-of-the-art baseline algorithm. The evaluation shows improved resource utilization rate, total processed orders, decreased average waiting time, and total failed orders, highlighting the effectiveness of the OARF method.

Keywords
Adaptive reward, Dynamic environments, Job shop scheduling, Multi-objective, Ontology, Reinforcement Learning (RL), Reward function, Unexpected events

1. Introduction

In the field of Reinforcement Learning (RL), agents learn optimal behavior by interacting with their environment and receiving rewards. The reward function, defined by the RL problem designer, guides the agent toward its goal. In dynamic environments, unexpected events introduce uncertainties and require the learning agent to be adaptable in order to effectively navigate and respond to changing conditions. Statically defining reward functions in such environments is not useful because the dynamics of the environment can change over time, leading to a mismatch between the predefined rewards and the desired behavior. Dynamically adjusting or adaptive reward functions are more useful in dynamic environments because they can adapt to changing circumstances and provide relevant feedback to guide the agent's decision-making [1].

Several approaches have been suggested in the literature, one of which is Inverse Reward Design (IRD), a method that derives reward functions from expert demonstrations [2]. However, a significant challenge arises due to the limited availability of human expertise. The reward shaping technique modifies the reward function to guide the learning agent towards desired behaviors. It provides additional, intermediate rewards during the learning process to facilitate faster and more effective learning. Reward shaping encompasses various approaches such as state-based potentials [3], intrinsic rewards [4], and learning-based reward shaping [5] to provide incentives for agents. However, there is a potential difficulty in shaping rewards for events that have not been explicitly encountered during the training phase. Dynamic reward shaping incorporates verbal feedback [6] or demonstrations [7] presenting human preferences and expectations. However, challenges may arise due to the potential for incorrect feedback or untimely demonstrations. Reward Machines (RMs) represented as Finite-State Automata (FSAs) [8] is a formalism used in RL that represents the desired behavior of an agent by specifying rewards associated with different states and events in an environment. However, in dynamic environments, unexpected events can disrupt the predefined reward structure, leading to unpredictable behavior.

Ontologies serve as a representation of human knowledge and aid in inferring properties within the environment. We propose a new Ontology-based Adaptive Reward Function (OARF) method, in which the agent initially constructs an ontology-based schema that represents its observations using concepts, properties, and relationships. Subsequently, by inferring beliefs about properties and constraints that define allowable values for those properties, the agent generates a new re-

MRC 2023 – The Fourteenth International Workshop Modelling and Representing Context. Held at ECAI 2023, 30.09.-5.10.2023, Kraków, Poland.

saeedeh.ghanadbashi@ucdconnect.ie (S. Ghanadbashi); a.zarchini@ce.sharif.edu (A. Zarchini); fatemeh.golpayegani@ucd.ie (F. Golpayegani)
0000-0003-0983-301X (S. Ghanadbashi); 0000-0003-4738-7604 (A. Zarchini); 0000-0002-3712-6550 (F. Golpayegani)

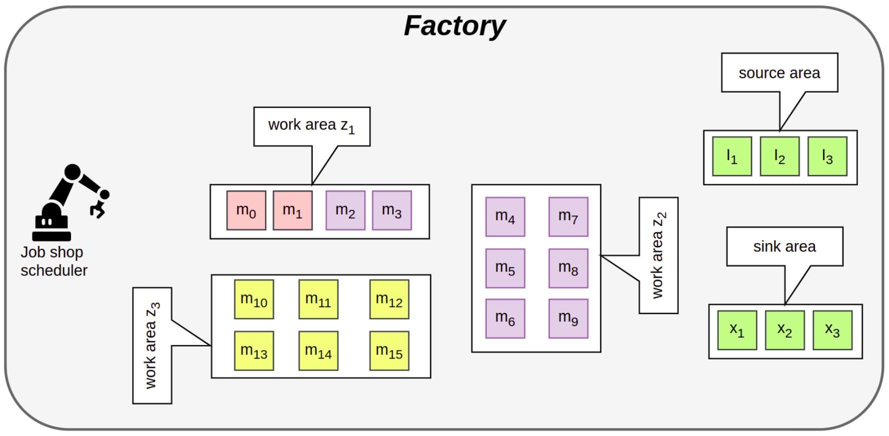


Figure 1: Simulated job shop scheduling environment. I_1, I_2, and I_3 are sources that generate orders. $\{m_0, m_1, \ldots, m_{15}\}$ are the machines that can process one order at a time. Machines are categorized into three groups, n_1 (purple color), n_2 (pink color), and n_3 (yellow color). Machines in the same group can perform a similar operation. z_1, z_2, and z_3 indicate work areas that are located in different locations in the factory. x_1, x_2, and x_3 are sinks that consume the processed orders.

ward function. This new reward function aims to maximize/minimize the value of a property associated with a positive/negative belief satisfying the constraints of the property.

The rest of this paper is organized as follows. Section 2 describes a motivating example in the job shop scheduling, and Section 3 explains the problem statement. Section 4 describes our method. In Section 5, the scenarios are defined, and the results are analyzed. Finally, our conclusion and future works are drawn in Section 6.

2. Job Shop Scheduling

Job Shop Scheduling (JSS) is a problem that involves processing multiple orders on multiple machines in a specific sequence. Unforeseen events in JSS can include machine breakdowns, order arrivals or cancellations and priority changes. In this section, JSS is presented as a motivating example to demonstrate the application of the new OARF method.

In the JSS environment, the scheduler agent selects an order from the queue of orders in the source. Then, it moves to a work area and selects a machine in the desired group to process the order. Finally, it selects an order from the queue of orders in the machine and places the processed order in a sink (see Figure 1).

3. Problem Statement

In an RL algorithm, the state s^t represents the observation of the agent at time step t, the action a^t is the decision made by the agent, and the reward r^t quantifies the immediate feedback or consequence associated with taking that action in a particular state. When setting up an RL algorithm, human experts typically define reward functions to guide the agent toward the desired goal. However, in dynamic environments, reward functions need to be created and modified dynamically and adaptively to accommodate unforeseen events. For instance, in the JSS environment, this involves defining a new reward function, such as *minimizing the waiting time for urgent orders*, when a new situation such as *generating high-priority orders* is detected. In this research, we propose using ontology to define/adapt new reward functions for newly identified events.

Ontology can be defined as a formal representation of knowledge that captures concepts, relationships, and properties within a domain to facilitate meaningful understanding and reasoning (e.g., interpretation of an unforeseen or unexpected event). For instance, an ontology, as depicted in Figure 2, is employed to capture the underlying concepts in the JSS environment. This ontology comprises seven fundamental concepts: Belief, Source, Order, WorkArea, Group, Machine, and Sink, each corresponding to a distinct entity in the JSS environment. Subclasses are defined within the ontology to represent more specialized concepts/properties under each superclass. Furthermore, human experiences inform the association of specific beliefs with certain properties. For instance, WaitingTime is linked to negative Belief, indicating that prolonged waiting time negatively affects the agent's performance.

As the environment undergoes changes, the concepts and properties observed by the agent also change. In response, the agent can adapt its reward function to max-

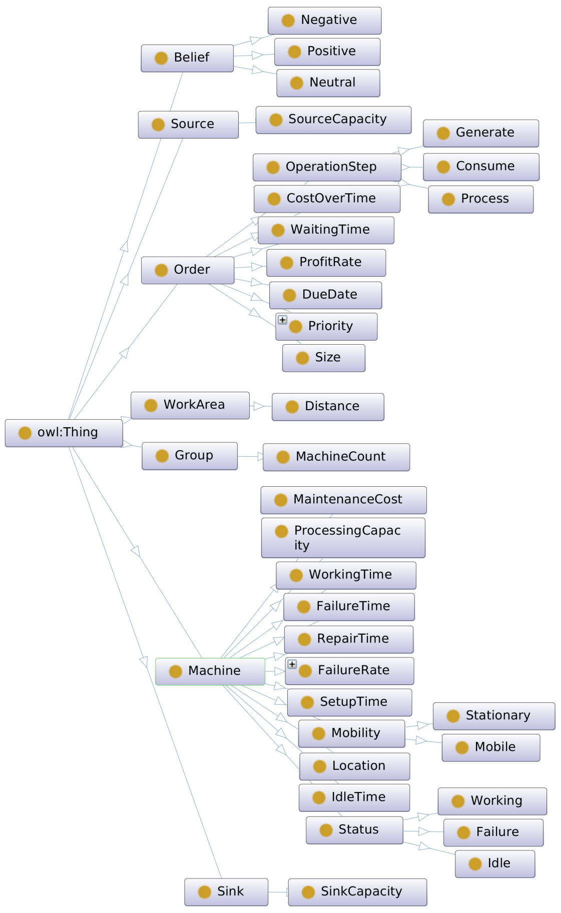


Figure 2: Ontology for the job shop scheduling environment.

imize positive properties and minimize negative properties, ensuring its behavior aligns with the desired objectives.

4. Ontology-based Adaptive Reward Function Method

We propose a new Ontology-based Adaptive Reward Function (OARF) Method, wherein the agent initially employs an ontology-based schema to represent its current state by concepts, properties, and relationships between them. Subsequently, a mechanism is established to prioritize the most crucial properties for immediate consideration. Then, the constraints associated with the property are inferred by employing logical reasoning over the existing relationships. Ultimately, the agent formulates new reward functions based on the belief value and constraints associated with highly prioritized properties encountered in the environment.

4.1. Stage 1: Ontology-based Modeling

In our previous paper [9], we discussed how the utilization of ontology allows agents to effectively represent and comprehend their observations through an ontology-based schema. The Semantic Sensor Network (SSN) ontology, a model based on a standard for sensor networks [10], is employed to describe the data collected by sensor resources (see Figure 3).

We define $O^t = \{C^t, F^t, L^t, W^t\}$ as the ontology-based schema describing the data monitored/observed by the agent at time step t. C represents the set of concepts, F represents properties and L represents the set of relationships over these concepts that expresses which concepts are associated with which concepts/values by which properties ($L \subseteq C \times F \times C$). The agent encounters different sets of concepts and properties when carrying out different tasks. For instance, while engaged in order selection, the job shop scheduler agent's observation is limited to the concept "Order" and its corresponding properties. Ontology engineers may assign weights W to the relationships that determine their importance. For example, within the context of JSS, the waiting time plays a crucial role in evaluating the scheduling process's performance, allowing an ontology engineer to assign a high importance weight to the relationship "hasWaitingTime" ($W(\text{hasWaitingTime}) = \text{High}$). Five degrees of weight are available and can be converted to numerical values using predefined mappings: "Lowest", "Low", "Middle", "High", and "Highest".

4.2. Stage 2: Property Prioritization

By leveraging the assigned relationships' weights, the agent can prioritize properties associated with relationships of higher importance. Let $W : L \to \mathbb{R}$ be the function that assigns a weight to each relationship, indicating its importance. For each property $f \in F$, we can define its priority based on the weights of the relationships associated with it:

$$Priority(f) = \Sigma_{l \in L_f} W(l) \qquad (1)$$

, where L_f is the set of relationships connected to property f.

4.3. Stage 3: Constraints Deducing

In this stage, through logical reasoning, the agent can derive allowable values for a specific property f based on the relationships it has with other concepts or properties in the ontology. The result is a set of equations that define the constraints $H(f)$ for property f. For example, in JSS, the agent can deduce the constraint

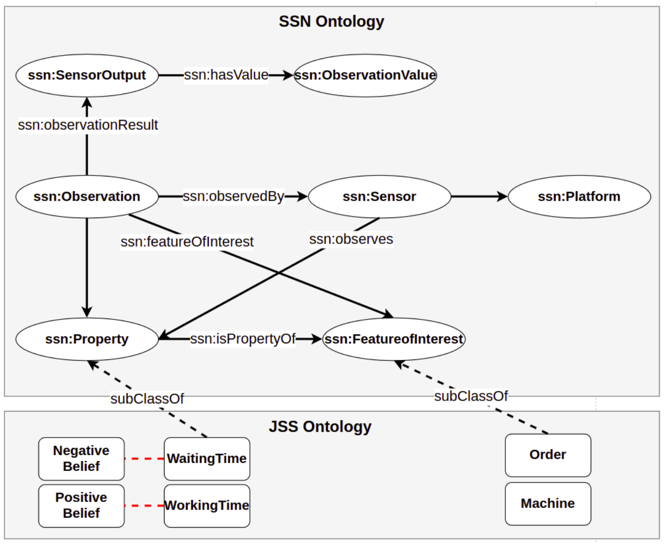


Figure 3: The application of the SSN ontology, combined with cross-domain knowledge, is employed to annotate and present sensor data [9].

H("'WorkingTime') : "WorkingTime"(m) ≥ "Target-WorkingTime", ensuring that the working time of machine m is equal to or greater than a specified target working time.

4.4. Stage 4: Reward Function Extraction

A set of reward functions for different states can be defined based on the properties with negative or positive beliefs in an ontology-based schema. To do so, the agent aims to maximize/minimize the value of properties associated with positive/negative beliefs subjecting to the relevant constraints (see Equation 2).

$$\begin{aligned}
&\text{if Belief}(f) = \text{Positive: } a \leftarrow \arg\max_{\forall a \in A} f, H(f) \leq 0 \qquad (2)\\
&\text{else if Belief}(f) = \text{Negative: } a \leftarrow \arg\min_{\forall a \in A} f, H(f) \leq 0\\
&\text{else if Belief}(f) = \text{Neutral: } f \text{ not considered for rewards}
\end{aligned}$$

In this formulation, the objective is to maximize/minimize the value of property f. The constraints are written in the form of inequality constraints, where each constraint should be less than or equal to zero. For example, for the order selection task in JSS, since waiting time is associated with negative belief, the agent defines minimizing the waiting time as its new reward function. By incorporating the reward function of minimizing waiting time into the agent's learning process, it optimizes the sequence of operations and allocates appropriate machine resources to minimize idle time and maximize throughput.

5. Evaluation

We conducted an evaluation of the OARF method in a simulated JSS environment (see Figure 1). We defined low, medium, and high priority orders as 30%, 50%, and 20% of the total orders, respectively. Furthermore, we set the failure rate for machines at 50% to be low, 30% to be medium, and 20% to be high. We performed 1000 simulated episodes, each with 100 simulation steps, and evaluated the performance of the baseline method against the OARF-enabled baseline method.

To evaluate the OARF method, we defined four **scenarios** that cover different order loads (number of orders per time step) and due date requirements. The scenarios are presented in Table 1.

The **baseline method** [11, 12] utilized the LIFO algorithm for order selection in the source area, the TRPO learning algorithm for machine selection, and the FIFO algorithm for order selection in the machine.

In the **proposed approach**, the agent employs the TRPO learner to select orders in the source area, aiming to maximize the due date of orders (represented by a new reward function defined based on the "DueDate" property in the domain ontology). This strategy ensures that orders with closer due dates are prioritized for processing. Then, the agent defines the reward function for machine selection as "maximizing the working time of machines". Next, to select orders in the machine, the agent employs the TRPO learning algorithm for four out

Table 1
Job shop scheduling environment settings.

Scenario		Description
Order Load	Light	Three orders are generated at each time step.
	Heavy	Six orders are generated at each time step.
Due Date Level	High	5% of orders have a low due date, 80% have a medium due date, and 15% have a high due date.
	Low	25% of orders have a low due date, 70% have a medium due date, and 5% have a high due date.

of sixteen machines with a new reward function of "minimizing the waiting time of orders". For the remaining twelve machines, the agent utilizes the baseline FIFO strategy.

To assess the proposed method's performance, we employed the following **performance metrics**: average working time of machines, the average waiting time of orders, the total number of orders that have failed to meet the required due date requirement, and the total number of successfully processed orders.

5.1. Results and Discussion

We conducted a thorough analysis of the performance metrics across ten runs for each scenario and the results are depicted in Figure 4. These results demonstrate that our OARF-enabled baseline method yielded improvements in the average utilization rate and total processed orders, while simultaneously reducing the average waiting time and total failed orders as compared to the baseline method. For the details of percentage change in metrics refer to Table 2. These results indicate that OARF method had a more significant impact on scenarios with a higher number of orders. Moreover, the improvement in the Low scenarios is lower than in High scenarios. This is because the number of low due date orders increases, and our proposed method shows less improvement due to the limited number of resources.

6. Summary and Future Directions

Within the field of Reinforcement Learning (RL), artificial agents learn optimal behavior by interacting with environments, guided by reward functions that dictate desired behavior. Reward function design becomes challenging in dynamic environments due to the ever-changing nature of the system. In a dynamic environment, the underlying dynamics, goals, and constraints can evolve over time, making it difficult to define a static reward function that remains effective. In order to facilitate the dynamic modification of reward functions, we introduce a novel method called Ontology-based Adaptive Reward Function (OARF). This approach involves defining a new reward function by considering the belief value and constraints associated with properties. This paper opens up several avenues for further exploration. While our focus has been on testing the proposed solution in job shop scheduling scenarios, future work will involve evaluating the effectiveness of OARF method in other scenarios and environments, including non-deterministic environments. The accuracy and completeness of the ontology used in the OARF method play a critical role in its performance, particularly in dynamic environments that necessitate frequent ontology evolution and updates. Additionally, comprehending and exploring an ontology-based schema can present challenges and summarization techniques can be utilized to provide concise overviews that focus on essential concepts, thereby facilitating better understanding [13, 14].

Furthermore, it is important to acknowledge that concepts and properties may vary depending on the context. Thus, the RL agent may need to employ different ontologies when operating under different conditions. Assessing the impact of incorrect beliefs about properties on the overall process becomes crucial in such cases. Additionally, a mechanism for dynamically adjusting weighting and objective priorities should be defined (multi-objective problem).

References

[1] S. M. Devlin, D. Kudenko, Dynamic potential-based reward shaping, in: Proceedings of the 11th international conference on autonomous agents and multiagent systems, IFAAMAS, 2012, pp. 433–440.

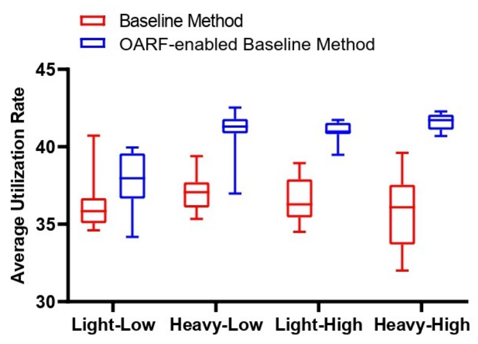
[2] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, A. Dragan, Inverse reward design, Advances in Neural Information Processing Systems 30 (2017).

[3] A. Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: International Conference on Machine Learning (ICML), volume 99, 1999, pp. 278–287.

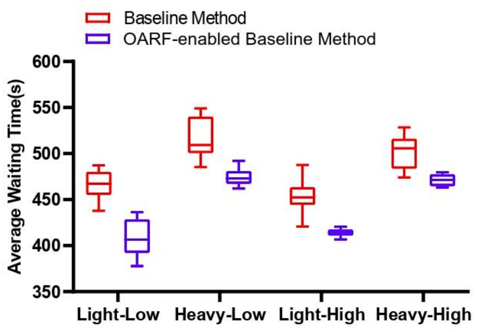
[4] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: International Conference on Machine Learning (ICML), PMLR, 2017, pp. 2778–2787.

[5] D. Yang, Y. Tang, Adaptive inner-reward shaping in sparse reward games, in: International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.

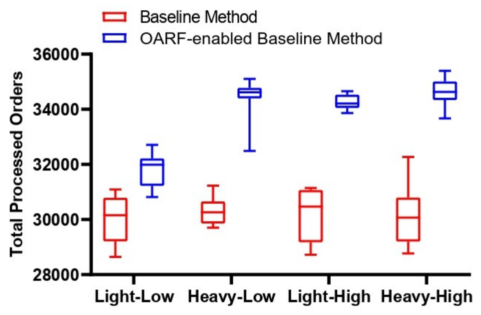
[6] A. C. Tenorio-Gonzalez, E. F. Morales, L. Villasenor-Pineda, Dynamic reward shaping: Training a robot



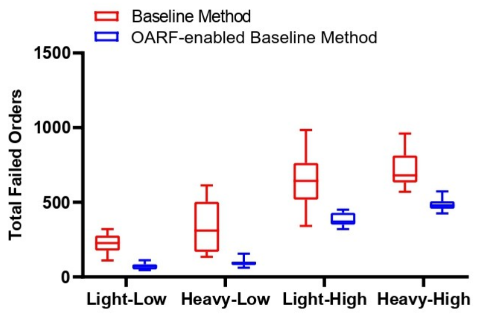
(a) Average utilization rate.



(b) Average waiting time.



(c) Total processed orders.



(d) Total failed orders.

Figure 4: The baseline method and the OARF-enabled baseline method.

Table 2
The Percentage Change in Performance Metrics – The Baseline Method and the OARF-enabled Baseline Method

	Performance Metrics				
Scenario	**Utilization Rate** (increase)	**Waiting Time** (decrease)	**Processed Orders** (increase)	**Failed Orders** (decrease)	**AVG**
Light-High	5%	13%	6%	68%	23%
Light-Low	11%	8%	14%	73%	27%
Heavy-High	12%	9%	14%	41%	19%
Heavy-Low	16%	6%	15%	32%	17%
AVG	11%	9%	12%	54%	-

by voice, in: Ibero-American Conference on Artificial Intelligence (IBERAMIA), Springer, 2010, pp. 483–492.

[7] A. Hussein, E. Elyan, M. M. Gaber, C. Jayne, Deep reward shaping from demonstrations, in: International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 510–517.

[8] R. Toro Icarte, T. Klassen, R. Valenzano, S. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: International Conference on Machine Learning (ICML), PMLR, 2018, pp. 2107–2116.

[9] S. Ghanadbashi, A. Zarchini, F. Golpayegani, an ontology-based augmented observation for decision-making in partially observable environments, in: International Conference on Agents and Artificial Intelligence (ICAART), SCITEPRESS, 2023, pp. 343–354.

[10] A. Haller, K. Janowicz, S. J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, C. Stadler, The modular SSN ontology: A joint W3C and OGC standard speci-

fying the semantics of sensors, observations, sampling, and actuation, Semantic Web 10 (2019) 9–32.

[11] A. Kuhnle, Simulation and reinforcement learning framework for production planning and control of complex job shop manufacturing systems, 2020. URL: https://github.com/AndreasKuhnle/SimRLFab, accessed: 2022-06-01.

[12] A. Kuhnle, N. Röhrig, G. Lanza, Autonomous order dispatching in the semiconductor industry using reinforcement learning, Procedia CIRP 79 (2019) 391–396.

[13] C. E. Pires, P. Sousa, Z. Kedad, A. C. Salgado, Summarizing ontology-based schemas in PDMS, in: International Conference on Data Engineering Workshops (ICDEW), IEEE, 2010, pp. 239–244.

[14] S. Pouriyeh, M. Allahyari, K. Kochut, H. R. Arabnia, A comprehensive survey of ontology summarization: Measures and methods, arXiv preprint arXiv:1801.01937 (2018).