ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

Multi-Agent Reinforcement Learning for Multi-Robot Intelligent Fixture Planning

Puram Anjalidevi[1]; Pulipati Bharath Chandra Seshu[2]; Torlapati Dileep Chakravarthi[3]; Gaddam Mounika[4]

[4]Assistant Professor, [1,2,3]B. Tech Student

[1,2,3,4] Dept. of Computer Science and Engineering, R.V.R. & J.C College of Engineering, Chowdavaram, Guntur, Andhra Pradesh, India

Publication Date: 2025/05/12

Abstract: Fixture layout planning is critical for securely holding components during production processes. An optimal fixture arrangement minimizes surface deformation and prevents crack propagation, thereby maintaining the structural integrity of components. Traditionally handled by engineers, fixture planning has grown too complex for manual methods alone. Conventional optimization often gets stuck in local optima, limiting effectiveness. While machine learning offers improvements, it demands costly, labeled data. This paper proposes a multi-agent reinforcement learning framework with team decision theory. The approach enables agents to learn collaboratively, improving fixture planning without heavy data reliance by simulating fixture placement on a flexible surface to minimize deformation under uniform pressure. Multiple agents select fixture pairs, with deformation estimated using plate bending theory. The environment supports reinforcement learning and highlights the benefits of strategic, informed placements.

How to Cite: Puram Anjalidevi; Pulipati Bharath Chandra Seshu; Torlapati Dileep Chakravarthi; Gaddam Mounika (2025), Multi-Agent Reinforcement Learning for Multi-Robot Intelligent Fixture Planning. *International Journal of Innovative Science and Research Technology*, 10(4), 3285-3294. https://doi.org/10.38124/ijisrt/25apr1710

I. INTRODUCTION

As technology advances, emerging machine learning models are driving the development of innovative solutions. Reinforcement Learning (RL) agents, in particular, offer significant potential in automated fixture planning by optimizing for factors such as cost, robustness, and usability. According to recent studies on automatic fixture design, RL agents learn through trial and error, receiving rewards for effective designs and penalties for suboptimal ones gradually enhancing their performance over time. The use of intelligent fixtures for reconfigurable manufacturing systems has garnered increasing attention in recent years. Fixtures, often referred to interchangeably as fixturing or fixel elements, are integral components within jig structures. These fixtures play a critical role in securely positioning and restraining a workpiece during various manufacturing processes, such as drilling, joining, or riveting. By ensuring precise placement and stability, fixtures enable efficient and accurate completion of complex manufacturing tasks. These fixtures are employed to ensure that the component remains within specified deformation limits, minimizing the risk of excessive deflection and the potential propagation of internal cracks.

The design of physical fixtures, particularly robotic end effectors, plays a pivotal role in manufacturing, with the design process being highly dependent on the specific type of component being constrained. Fixture planning aims to strategically position fixturing elements to minimize residual stress and deformation during manufacturing operations, thereby preserving the component's structural integrity and performance. Traditionally, this task was handled by experienced design engineers, but recent advancements have explored optimization based approaches. Despite these improvements, fixture layout planning remains a complex challenge due to the inherent limitations of conventional optimization methods.

Reinforcement learning (RL) has emerged as a promising alternative to traditional optimization techniques, particularly for problems involving multiple agents. While previous methods predominantly relied on a single robotic fixture, real-world manufacturing environments often involve multiple fixtures that must coordinate their actions in a shared environment. This paper introduces a multi-agent reinforcement learning (MARL) framework for optimizing fixture layout planning, focusing on the collaborative behavior of robotic fixtures. The paper provides an overview of current fixture planning methods, presents the Multi-Robot Fixture Planner (MRFP) methodology, and evaluates its effectiveness through a case study on aerospace wing panel and spar drilling, comparing its performance to state-of-the-art fixture planning methods.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

II. RELATED WORK

The design of fixtures, particularly robotic end effectors, is a critical aspect of manufacturing, with the specific design approach depending on the type of component being processed. The objective of fixture planning is to strategically position fixturing elements to minimize residual stress and deformation during manufacturing, thereby ensuring the component's structural integrity and performance.

➢ *Existing System*

Historically, this task was managed by experienced engineers; however, recent advancements have introduced optimization-based techniques. Despite these innovations, fixture layout planning remains a complex challenge due to the limitations of traditional optimization methods. Fixture design in manufacturing involves two main parts: designing the physical fixtures and developing the strategy to hold the component in place. While there have been improvements in the physical design of fixtures, especially with the use of robotic fixtures, creating effective fixture plans is still a challenge. These plans are typically made by experienced operators, and their main purpose is to ensure that a component is securely held in place before tasks like drilling or riveting are performed. The goal is to minimize any unwanted deformation or movement of the component during these processes.

Definition 1: Let A ⊆ A be a set of fixture locations chosen from the global set of positions A. The locations of A during a manufacturing operation τ should reduce the experienced deformation in 3-dimensions $f_w(\tau)$

$$A^* \in \arg\min_{A \subseteq \mathcal{A}} |f_w(\tau)|$$

The majority of research on fixture design planning can be categorized into three primary approaches: part similarity-based design, optimization-based design, and machine learning-based design. Case-Based Reasoning (CBR) was one of the earliest methods used, relying on the assumption that components within the same Stock Keeping Units (SKUs) are sufficiently similar to allow for the reuse of fixture plans. Another approach, Rule-Based Reasoning (RBR), generates rules to identify similarities between components. Both methods are particularly effective when the components are variations of the same SKU. However, these approaches depend on the assumption that components share a level of similarity and that the initial fixture plans stored in the database are optimal. Limitations arise in effectively retrieving suitable plans, especially when components deviate from the expected similarity, which can lead to the misidentification of appropriate fixture designs.

In parallel with Case-Based Reasoning (CBR) and Rule-Based Reasoning (RBR) methods, optimization-based approaches have also gained significant traction in fixture layout planning. These approaches extend the basic framework of fixture design by incorporating various constraints and parameters, aiming to optimize the fixture arrangement while adhering to specific design requirements and operational conditions. The objective is to identify the most efficient fixture configuration that meets these additional criteria and improves the overall design performance.

Minimise

$\tau \in T$ (max $|f_w(\tau)|$)

s.t. $a_i \leq x_i \leq b_i$

$c_i \leq y_i \leq d_i$

Where:

T = Set of manufacturing operations performed

x_i, y_i = Coordinate positions of fixture i

a, b = Upper and lower position limits in the x-direction

c, d = Upper and lower position limits in the y-direction

$f_w(\tau)$ = Deformation in component due to task τ

Fixture design planning has used optimization methods for a long time, starting around the 1990s. One common method is the genetic algorithm, which starts with rough solutions and gradually improves them. Other techniques like Particle Swarm Optimization and strategies like reducing the number of fixture pins have also been tried. Some researchers added specific rules, like limiting how many fixtures can be placed on one surface, to improve results. However, these methods sometimes get stuck and can't find the best solution.

More recently, machine learning has been used to improve fixture planning. Early methods used techniques like neural networks to recognize patterns, similar to older methods that relied on comparing new tasks to past examples. But supervised learning has limits because it needs a lot of labeled data and still doesn't guarantee the best result. That's why self-supervised learning methods like reinforcement learning (RL) are now being explored. In RL, the system learns through trial and error without needing labeled data. Some researchers use RL to remove one fixture at a time, or train a virtual model (digital twin) to find better fixture plans. However, these methods can be slow and may not work well across different tasks unless trained separately for each one.

➢ *Multi-Agent Reinforcement Learning*

Multi-agent reinforcement learning (MARL) builds on the traditional RL methods by scaling the Markov decision processes (MDPs) to multiple agents. Single agent RL is characterised by the tuple $\langle S, A, R, P, \gamma \rangle$, where at each state st the agent takes an action at which transitions the environment to the next state s_{t+1} and provides reward Rt to the agent. The agent seeks to maximise the expected value of the discounted reward from the current state:

$$V(s) = \mathbb{E}_{s_{t+1} \sim P, a_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s \right]$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

Where actions at each step at are chosen from a policy π. The agent's overall goal is to find a policy that maximises the cost found in equation. For multiple agents, the sequential decision making of multi-agent reinforcement learning (MARL) is now a factor of all the agents operating in the environment.

This can be formulated as a Markov or Stochastic game defined as a tuple $\langle N, S, \{A\}_{1:n}, P, \{R\}_{1:n}, \gamma \rangle$. In a Markov game, each agent $n \in N$ takes an action from their action space which forms the joint action for all agents $a_t = a^1{}_t \times \ldots \times a^n{}_t, \forall n \in N$. The probabilistic state transition function P now maps the joint action and the current state into the new state $P : s_t \times a_t \rightarrow s_{t+1}$. Similarly to the single agent problem, each agent wants to maximise their cumulative reward through the value function:

$$V^n_{\pi^n, \boldsymbol{\pi}^{-n}}(s) = \mathbb{E}_{a_{t+1} \sim P, \boldsymbol{a}_t \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R^n_t | s_0 = s \right]$$

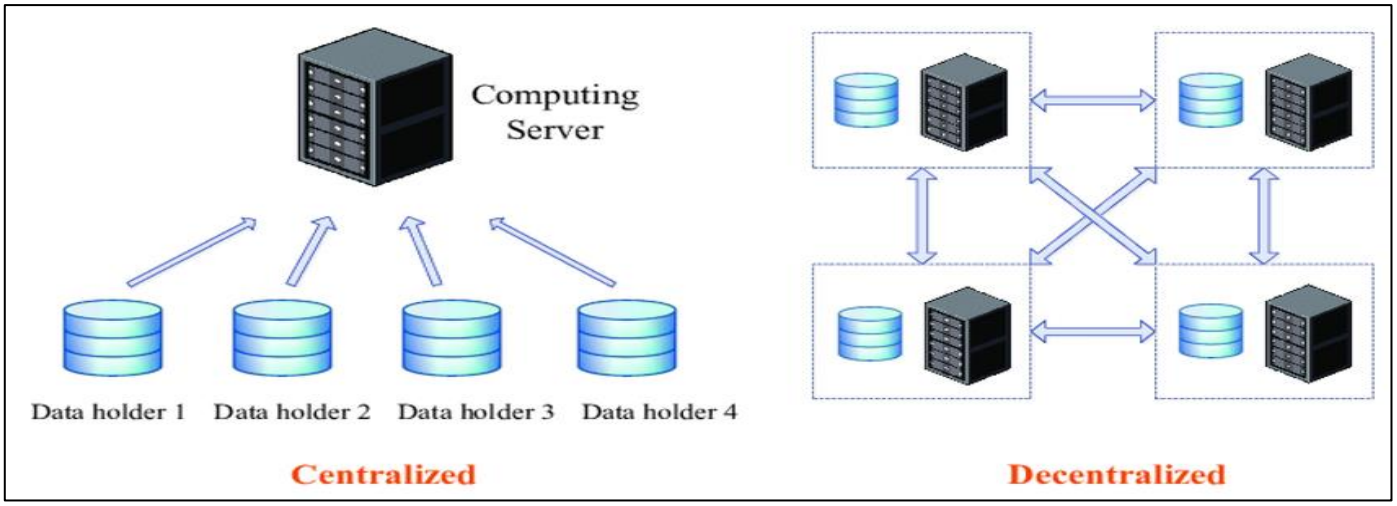

Fig 1 Comparison between the Architecture of a Centralized Multi-Agent Reinforcement Learning (LEFT) and the Decentralized Multi-Agent Reinforcement Learning (RIGHT) Architectures.

Equation shows that the value function is dependent on the joint policy of all agents $\boldsymbol{\pi} = \{\pi^1, \ldots, \pi^n\}$. This gives way to the equilibrium condition known as Nash equilibrium. Definition 2: For a set of agents N, an agent's policy $\pi^n{}_*$ can be considered a best response to the set of policies for all other agents $\pi^{-n}{}_* = \{\pi^1{}_*, \ldots, \pi^{i-1}{}_*, \pi^{i+1}, \ldots, \pi^n{}_*\}$ excluding agent n provided that the inequality:

$$V^n_{\pi^n_*, \boldsymbol{\pi}^{-n}_*}(s) \geq V^n_{\pi^n, \boldsymbol{\pi}^{-n}_*}(s), \forall s \in \mathcal{S}$$

In multi-agent reinforcement learning (MARL), the objective is to learn agent policies that converge to a Nash Equilibrium, a state in which no agent can unilaterally improve its outcome by deviating from its current policy. The literature typically distinguishes between two primary training paradigms: centralized and decentralized training. Centralized training employs a single, joint policy that governs the actions of all agents, making it well-suited for homogeneous agent populations. A common implementation of this approach is Centralized Training with Decentralized Execution (CTDE), where agents are trained collectively using shared information but operate independently during deployment. In contrast, decentralized training assigns each agent its own policy, enabling them to learn and act independently based on local observations. This approach is particularly important in environments with heterogeneous agents or in competitive and partially cooperative scenarios, where individual objectives and behaviors may diverge. Decentralized training is often more applicable in real-world systems, where agents are inherently distributed and may possess varying roles, capabilities, or goals.

The decentralized and interactive nature of agents leads to a non-stationary environment, as the learning dynamics of one agent continuously alter the environment observed by others. To address this challenge, solution methods must explicitly account for this non-stationarity. Hu and Wellman demonstrated that Q-learning can be extended to compute Nash Equilibria by incorporating information about other agents' actions into the Q-value updates. This formulation can be further generalized through the use of function approximators, such as neural networks, enabling scalability to more complex and high-dimensional settings. Actor-critic architectures have also shown considerable promise in MARL, with effective implementations utilizing either a single actor with multiple critics or fully decentralized configurations involving multiple actors and critics. Additionally, MARL frameworks have been successfully applied to hybrid human-agent environments, where social modeling informs the agents' decision-making processes.

III. MULTI-AGENT FIXTURE PLANNING

This section presents the Multi-Robot Fixture Planner (MRFP) system, developed using Multi-Agent Reinforcement Learning (MARL). It begins with an introduction to a simplified reinforcement learning framework tailored for fixture planning, which serves as the foundation for the subsequent development. The discussion then progresses to the cooperative mechanisms among multiple robotic agents,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

detailing how collaboration enhances the decision-making process. Finally, a comprehensive overview of the fixture design planning process in a multi-robot environment is provided, highlighting the integration and coordination strategies employed within the system.

➢ *Feature-driven Bandits*

As mentioned , many systems don't actually need the full complexity of a Reinforcement Learning (RL) model, known as a Markov Decision Process (MDP). Instead, they can use a simpler approach called contextual bandits. This is a version of the multi-armed bandit problem often discussed in RL. The main idea behind contextual bandits is that each situation (or state) the agent sees doesn't depend on what happened before. In other words, there's no need to worry about how one state leads to another, so the system doesn't need a transition model (called P in MDPs). The tuple $\langle S,A,R,P,\gamma \rangle$ becomes $\langle S,A,R \rangle$. For fixture planning, Feature-driven Bandits represent the pro cesses that are performed when fixtures are applied to a system, in particular drilling tasks. The set of drilling positions can be defined as the states S, where the agent is then allowed to choose an action from its policy that represents a fixturing position. This action and state is then used in the reward function R that is based on the maximum deformation that the component is experiencing during this task, which the agent uses to update its policy. In the contextual bandit setting, agents are looking to minimise the episodic regret over the total set of states:

$$\mathfrak{R}(\mathcal{S}) = \mathbb{E}\left[\sum_{s\in\mathcal{S}} R_{s,a^*} - R_{s,a}\right]$$

In this context, a* denotes the optimal action that yields the highest expected reward in a given state, represented as $R_{s,a*,}$ and is compared against the actual reward obtained, $R_{s,a.}$ As previously discussed, standard reinforcement learning (RL) and contextual bandit approaches are designed to select a *single action per decision step*. While this is appropriate for fixture planning involving a single agent placing one fixture at a time, many industrial applications require *multiple fixtures to be positioned simultaneously* across a component. This significantly increases the complexity of the problem, rendering single-agent solutions computationally intractable.

➢ *Decision Making for Multi-Agent Fixtures*

Deploying multiple robotic fixtures to secure a single large component necessitates cooperative behavior among the fixtures to minimize deformation during operation. To accommodate this, the standard contextual bandit framework can be extended to support multiple agents, each with distinct action spaces, forming the tuple $\langle S, G, R, \{A\}_{1:N} \rangle$. Here, G represents a team of N agents, each with potentially heterogeneous capabilities due to physical constraints that limit where fixtures can be placed. As a result, each agent possesses a unique action space. During each decision round, the team is presented with a specified drilling location, and each agent $V_n \in G$, for all $n \in N$, must select an appropriate fixturing position from its respective action space $a_n \in A_n \forall\ V_n \in G$.

$$\boldsymbol{a} = \prod_{i=1}^{N} a_i \text{ and } \boldsymbol{a}_{-n} = \boldsymbol{a} \setminus a_n \quad \forall v_n \in \mathcal{G},\ n \in N$$

Definition 3: We denote the action set of agent $V_n \in G$, $\forall\ n \in N$ as A_n as a finite discrete set of fixturing positions that each agent can take for each drilling position. The set of positions must satisfy the relationship:

$$\mathcal{A}_n := \{a \mid a \in \Gamma \text{ and } 0 < |\mathcal{A}_n| < |\Gamma|\}$$

Since each agent is only allowed to place fixtures in specific areas, and not in other agents' areas, we make sure that every agent has its own unique set of possible actions. This means that no two agents can choose the same action their action sets don't overlap at al. Now, suppose there are several drilling holes on a component. For each hole, every agent picks one action from its own action set. This action represents the position where the agent wants to place its fixture, and the choice is made based on the agent's policy (which is basically its decision-making strategy that depends on the specific hole). The joint decision rule defined denotes the position of all fixtures on a component, which is translated into a reward using the reward function R: s× a→$R_{s,a}$.

Choosing a useful and meaningful reward function in RL is challenging. While binary rewards r ∈ [0,1] are common, defining deformation limits varies by component. the reward must be convex to enable team-based learning.However, this is hard when the optimal reward R_{s,a^*} is unknown or undefined. The optimal reward can be defined as the mean of the reward function $\bar{R}$ over a finite set of states T, which can leads to the euation :

$$\begin{aligned}\mathfrak{R}(\mathcal{S}) &= \mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{v_n\in\mathcal{G}} \bar{\mathcal{R}} - R_{s,\boldsymbol{a}}\right] \\ &= TN\bar{\mathcal{R}} - \mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{v_n\in\mathcal{G}} R_{s,\boldsymbol{a}}\right]\end{aligned}$$

This allows us to define the cost function as a maximisation problem which are more com mon in RL as the first term in previous equation is constant, leaving us with the objective for training as:

$$\max_{\phi\in\Phi} \mathbb{E}_{\boldsymbol{a}\sim\Phi}\left[\sum_{s\in\mathcal{S}}\sum_{v_n\in\mathcal{G}} R_{s,\boldsymbol{a}}\right]$$

The objective is to maximize the sum of rewards for all agents, i.e., the return of a policy.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

- The inequality from Theorem 1 holds if Equation 13 remains the training objective.
- The reward function must be differentiable for optimization.
- For concave problems, agents should be rewarded more as deformation approaches zero

e.g., using a Gaussian reward.

$$\mathcal{R}(s, \boldsymbol{a}) = \exp\left[-\frac{d_x^2}{2\sigma_x^2} - \frac{d_z^2}{2\sigma_z^2}\right]$$
$$\text{s.t. } d_x, d_z \in \mathbb{R}^2_{>0}$$
$$\mathcal{R} \in [0, 1]$$

Where:

d_x, d_z = Maximum deformations in the x, z directions

σ_x, σ_z = Variables determining the width of the curve

The mean reward can be calculated over the function's measurement domain.

$$\bar{\mathcal{R}} = \frac{1}{\text{Area}(U)} \int_U \mathcal{R}(s, \boldsymbol{a}) dA$$

➢ *Multi-Robot Fixture Planning*

To train the multi-robot fixture planning system, each robot follows a rule or strategy (called a policy, noted as φ_n) that helps it decide what to do at different drilling positions. These strategies are trained using method called *Nash-Q learning*. This method is chosen because it's good at finding the best set of actions for all robots when they each have a limited number of actions to choose from (discrete actions). It's also better at reaching these good solutions compared to many other multi-robot learning methods. Nash-Q is a great fit for this system because it can perform at a human level, unlike many other learning methods.

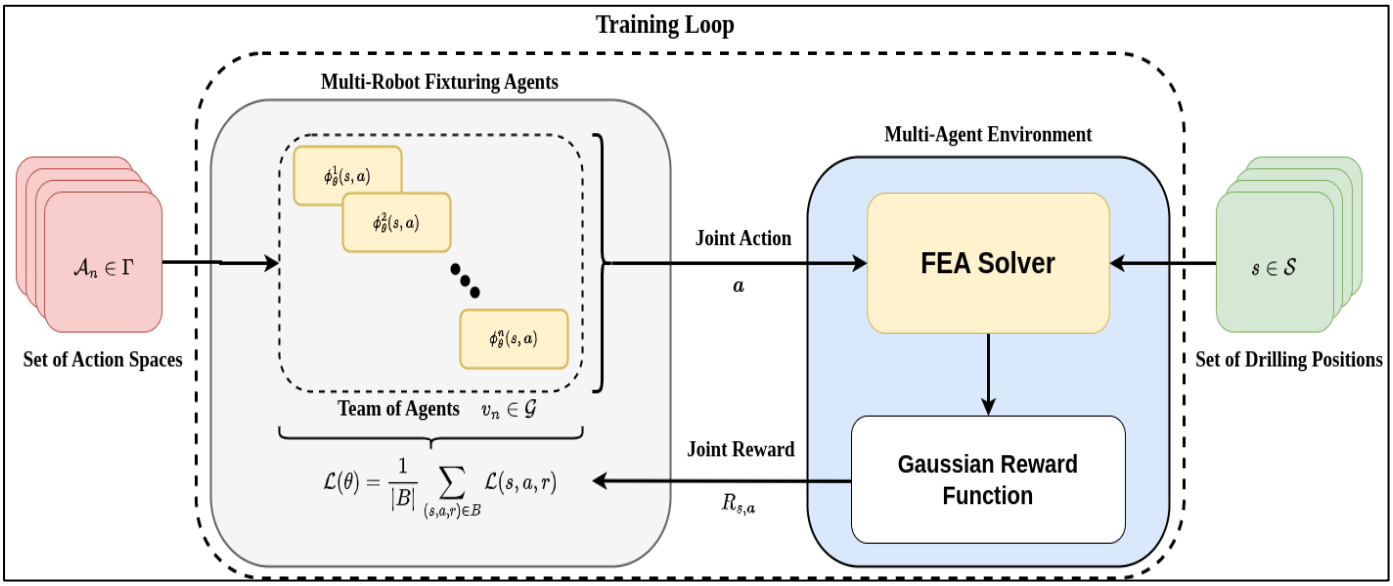

Fig 2 An Outline of the Training Process for the Multi-Robot Fixture Planner. Each Agent Policy φ Maps an Action to a Drilling Position Based on the Joint Action Learning Process. The FEA Solver uses the Drilling Positions and the Joint Action to Generate a Deformation Profile, Which the Reward Function uses to Generate a Reward for all the Agents to Optimise their Policies.

It's also an *off-policy* approach, which means it can learn from any available data, not just data collected during training. This makes it more efficient and requires fewer training samples compared to *on-policy* methods like policy gradient (PG).

Another big advantage is that Nash-Q reduces uncertainty in decision-making. PG methods use randomness to choose actions, which can lead to inconsistent behavior not ideal for critical tasks like those in aerospace. Nash-Q, on the other hand, helps the robots make more reliable decisions, even when they don't have full information about what others are doing. This makes it especially useful for figuring out the best way to place fixtures with multiple robots.

We can rewrite the equation with the maximisation approach to determine Nash equilibrium Q-values for each agent at each state in the game:

$$Q(s, \{a_n^*, \boldsymbol{a}_{-n}^*\}) \geq Q(s, \{a_n, \boldsymbol{a}_{-n}^*\})$$

The training cycle is shown in figure 2 where the process starts with the robots selecting actions from a Set of Action Spaces ($A_n \in \Gamma$) These actions are determined by each agent's Policy Network ($\phi\theta(s,a)$), which learns and improves over time. The team of agents ($V_n \in G$) works together to choose a *Joint Action* that is then tested in the FEA Solver. The FEA Solver simulates the physical effects of the joint action on the workpiece using Finite Element Analysis

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

(FEA). Based on the simulation results, a Gaussian Reward Function calculates a Joint Reward ($R_{s,a}$) which evaluates how well the action minimized deformation. The agents then update their policy networks using the calculated loss function L(θ), helping them learn the best strategies. This loop continues until the agents maximize the rewards, ensuring stability and precision at the Set of Drilling Positions ($s \in S$).

IV. RESULTS

This section of the paper will outline the experimentation and results for evaluating the multi-agent reinforcement learn ing approach for robotic fixtures. To benchmark our approach, this section will cover two extensive case studies that are common within fixture design planning and a conceptual approach.

In figure 3, there is a definite position of equilibrium, indicating the presence of a player-by-player equilibrium for the multi-agent f ixture design problem at the point where each agent chooses their best action. When the agents are trained in simulation to find the optimal positions for a single drilling hole in the method of the multi-armed bandit

Table I Training Details for Test Cases

Parameters	Value
Number of Agents	5
Number of Episodes	100
€- decay Starting Value	0.9
€- decay Ending Value	0.05
€- decay rate	3500
Q-Value α Parameter	0.8
Learning rate	1×10^{-4}
Reward Function Direction	x and z
Material type	Aluminum, Steel, Titanium
Number of Fixtures	5

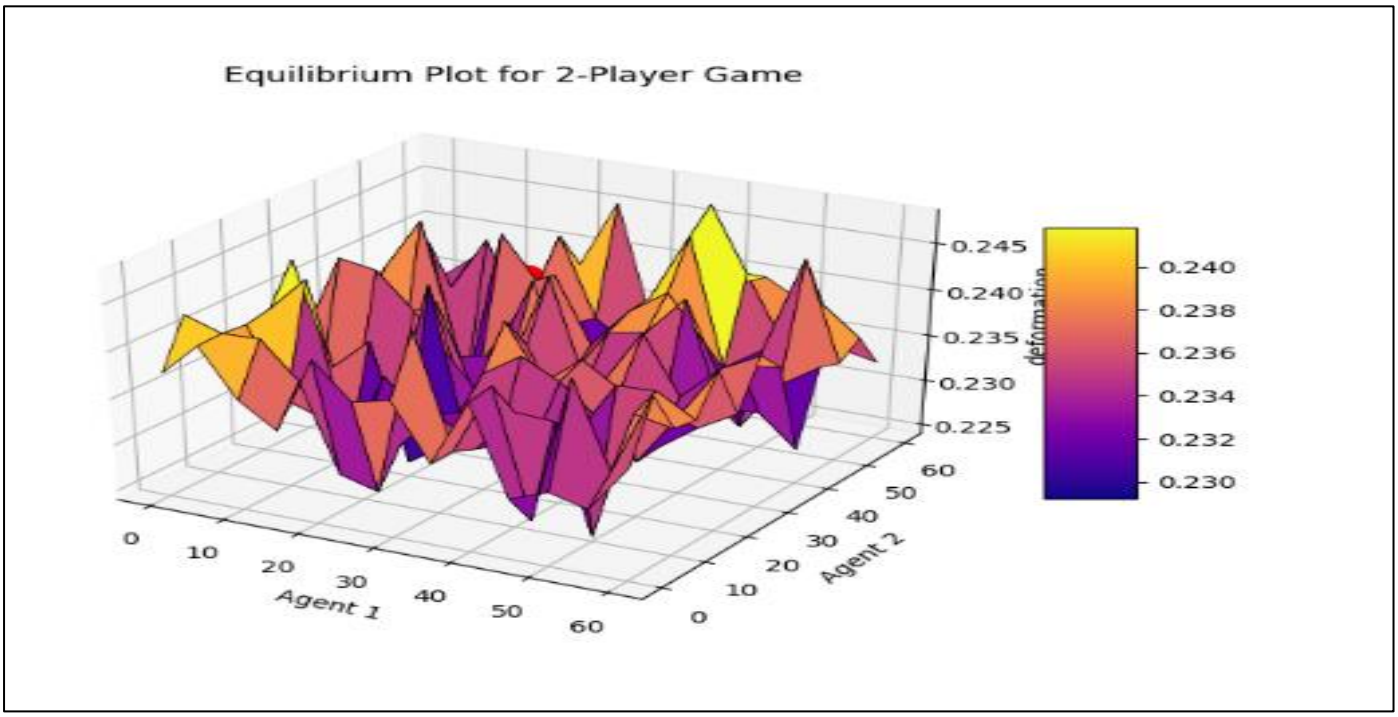

Fig 3 An Equilibrium Plot for the 2-Player Game that Demonstrates the Existence of a Player-by-Player Equilibrium.

The results of the training step can be found on the first three of graphs in figure 4 where some initial observations can be made. In figure 4(c), we can see that only fixture sets 5 and greater are able to improve on the quasi-optimal policy based on the deformation tolerance. The figure 4(a) helps assess how well the agents are performing on individual steps across all episodes. Initially, rewards might fluctuate or be low, but a positive trend suggests that the agents are learning better fixture placements that minimize deformation. We can see A steadily increasing return over episodes indicates effective learning and improved coordination between agents in figure 4(b) Since reward is tied to deformation reduction, higher returns mean better fixture strategies. Regret is highest early on when agents are still learning. A downward trend in figure 4(c) graph is a good sign it means agents are making fewer mistakes and getting closer to optimal decisions. Among all the output visualizations, the figure 4(d) stands out as a major diagnostic tool for evaluating the

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

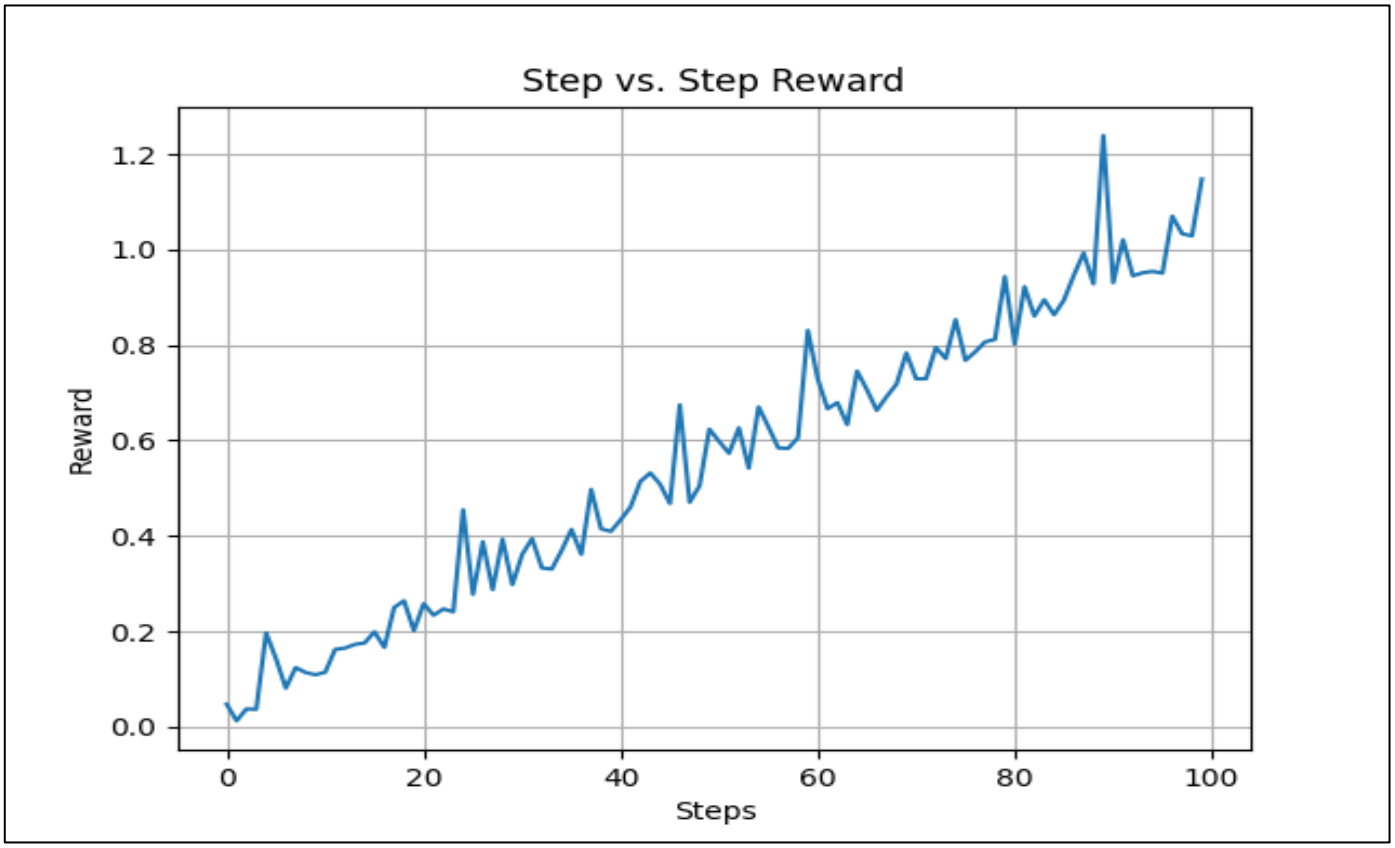

Fig 4(a) Step vs. Step Reward

consistency and robustness of fixture placement strategies in the MARL system. Unlike line plots that focus solely on average trends over time, this plot reveals the full distribution of deformation values across different fixture configurations. The shape and width of each violin provide deep insight

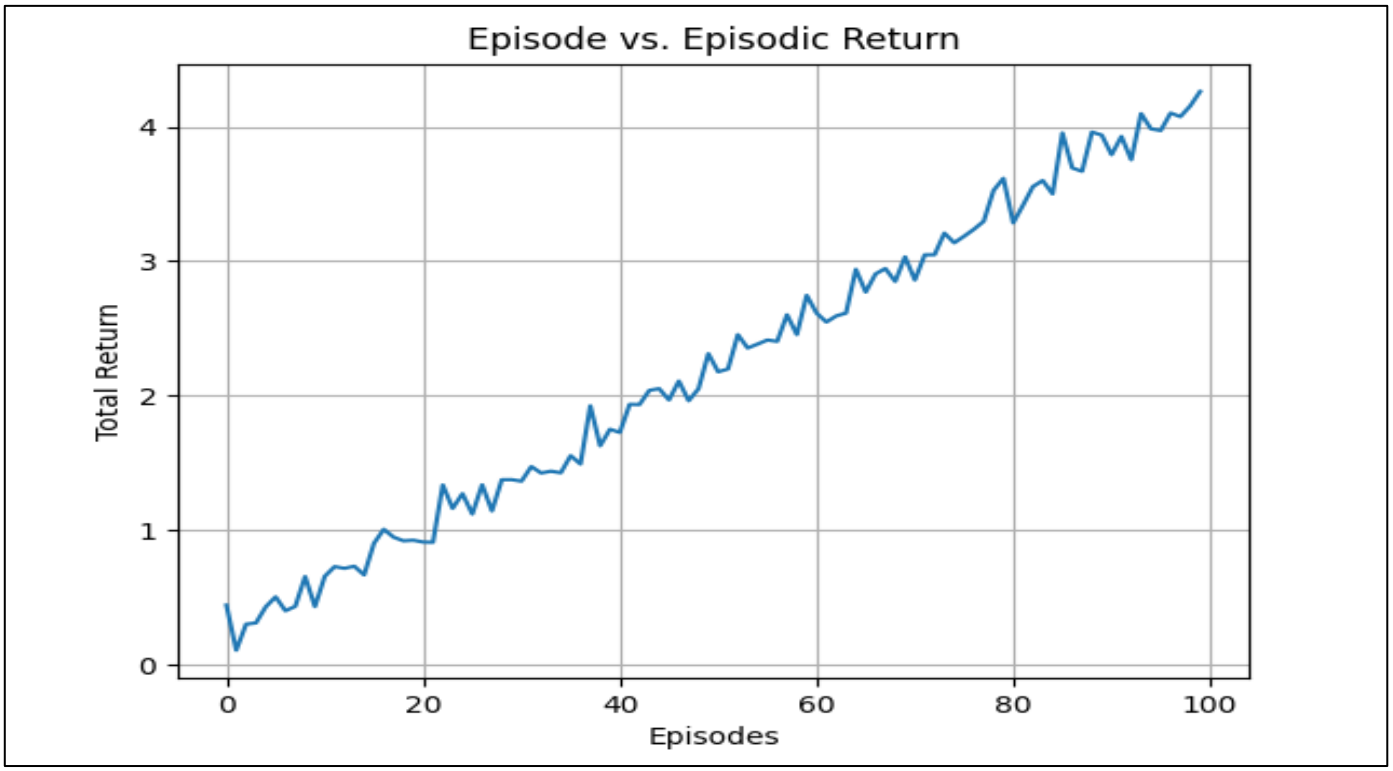

Fig 4(b) Episode vs. Episodic Return

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

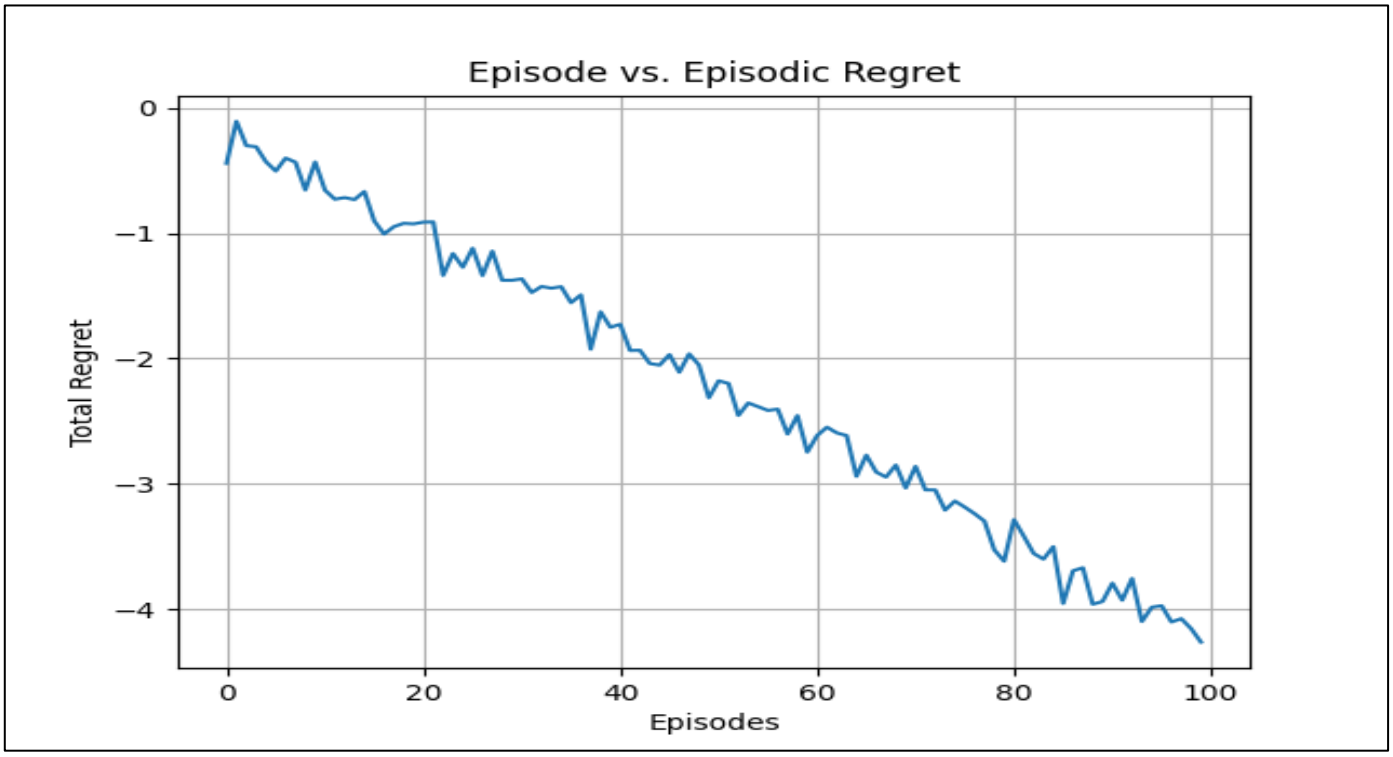

Fig 4(c) Episode vs. Episodic Regret

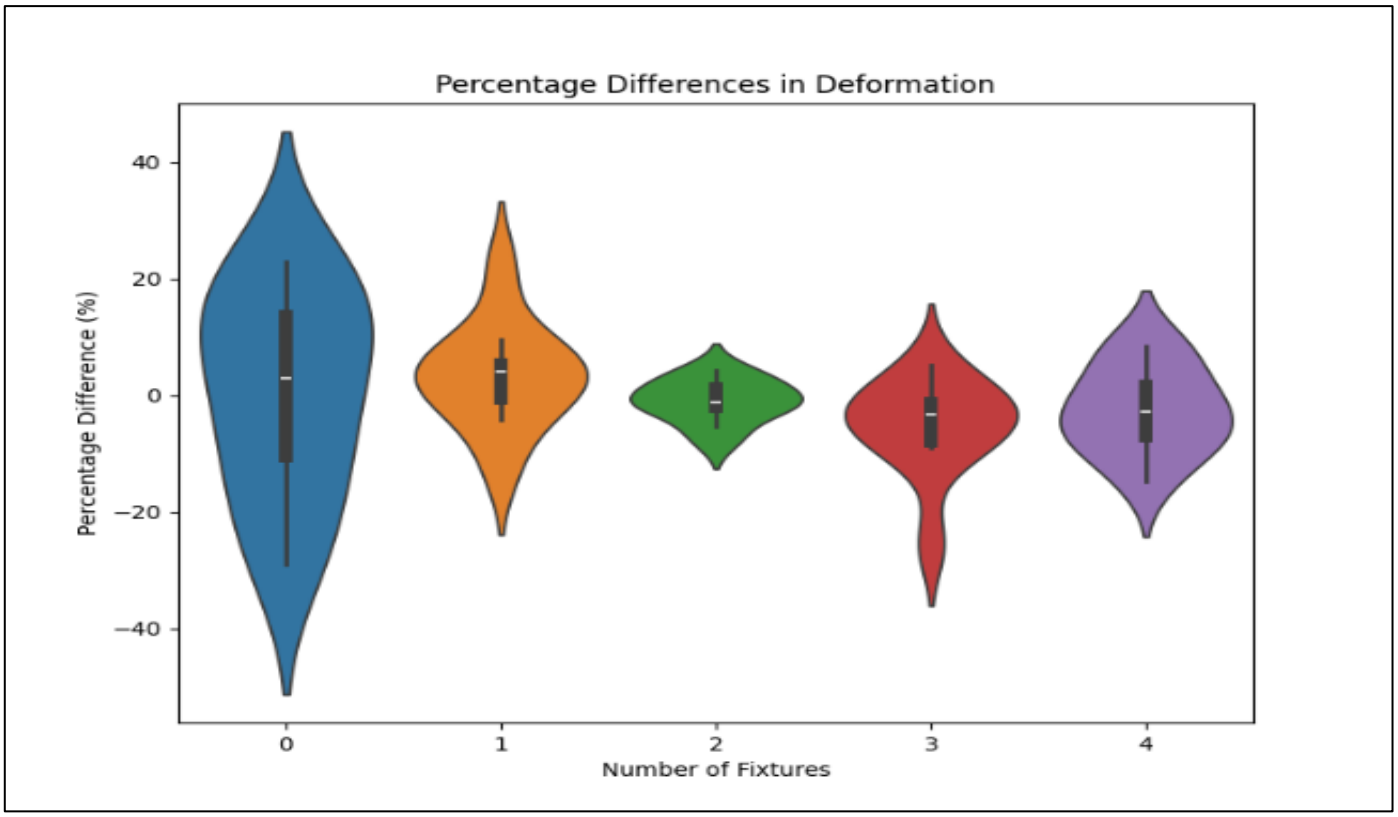

Fig 4(d) Percentage Differences in Deformation

into how variable or stable the deformation outcomes are highlighting whether the agents are converging toward consistent, optimal strategies. A narrow, short violin close to zero deformation difference indicates a highly effective and reliable fixture setup, whereas wide, irregular violins suggest inconsistency and learning instability.

V. TEST CASES

The first test case focuses on fixture planning for a bus component. In this scenario, we simulate a scenario where a large bus part, like a metal panel, needs to be supported by fixtures during assembly to prevent it from bending or

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

deforming. We use a multi-agent reinforcement learning setup, where each agent is responsible for selecting one fixture location from a set of predefined candidate points on the bus model. The environment either in Unity or a Python-based simulator takes these fixture placements and simulates the resulting deformation of the bus part. Based on how much the part bends under simulated conditions, the agents receive a shared reward: the lower the deformation, the higher the reward. During training, each agent observes information about the current fixture layout and bus geometry, makes a decision about where to place its fixture, and then updates its behavior using the reward feedback. Over many episodes, the agents learn to coordinate and place fixtures in optimal locations that minimize deformation. This approach mimics a real-world scenario in manufacturing, where smart, adaptive fixture design is crucial for quality control and efficient production.

The second test case addresses the problem of fixture planning for a rotating fan assembly comprising a central motor and multiple wings (blades). This scenario simulates operational conditions in which the fan is subjected to rotational forces, leading to potential deformation of the wings if not adequately supported. The objective is to determine optimal fixture placements that provide structural stability and minimize deformation during rotation.

A multi-agent reinforcement learning (MARL) framework is employed, wherein each agent is tasked with selecting one fixture position from a predefined set of candidate points on the fan model. The simulation environment developed using Unity or a Python-based physics engine evaluates the impact of these fixture placements by simulating rotational dynamics and calculating the resulting structural deformations.

Agents receive a shared reward based on the degree of deformation observed: the lower the deformation, the higher the reward. Throughout the training process, each agent observes the fan geometry and current fixture configuration, selects an action (i.e., a fixture location), and updates its policy using reward feedback. Over successive episodes, the agents learn to coordinate their decisions to identify fixture placements that maximize structural integrity and balance under dynamic loading.

This test case reflects real-world challenges in the manufacturing and assembly of rotating components, where accurate fixture design is critical to ensuring both mechanical stability and operational safety.

VI. CONCLUSION

In this paper, we introduced a multi-agent reinforcement learning method for finding optimal fixture plans for compo nents during drilling tasks. We outlined how robotic fixtures in a multi-agent system can constrain components and reorganise themselves to positions that are optimal for the desired task. We outlined the combination of multi-agent reinforcement learning and team decision theory into a framework that enables robotic fixtures to reconfigure themselves into optimal positions.

FUTURE WORK

Future work in this field would benefit from considering configurations involving n agents and m fixtures, where $n>m$, $n<m$, $n=m$, and $n\neq m$, to generalize the framework's adaptability across diverse real-world setups. Investigate how communication constraints or network topologies affect coordination among agents as the number of agents and fixtures scales up.

REFERENCES

[1] W. Park, J. Park, H. Kim, N. Kim, and D. Y. Kim, “Assembly part positioning on transformable pin array fixture by active pin maximization and joining point alignment,” IEEE Trans. Autom. Sci. Eng., vol. 19, no. 2, pp. 1047–1057, Apr. 2022.

[2] Boyle, Y. Rong, and D. C. Brown, “A review and analysis of current computer-aided fixture design approaches,” Robot. Comput. Integr. Manuf., vol. 27, no. 1, pp. 1–12, Feb. 2011.

[3] S. Gronauer and K. Diepold, “Multi-agent deep reinforcement learning: A survey,” Artif. Intell. Rev., vol. 55, no. 2, pp. 895–943, 2022.

[4] J. H. van Schuppen and T. Villa, Eds., Coordination Control of Dis tributed Systems (Lecture Notes in Control and Information Sciences), vol. 456. Cham, Switzerland: Springer, 2015.

[5] F. Liqing and A. S. Kumar, “XML-based representation in a CBR system for fixture design,” Comput.-Aided Des. Appl., vol. 2, nos. 1–4, pp. 339–348, Jan. 2005.

[6] C. Luo, X. Wang, C. Su, and Z. Ni, “A fixture design retrieving method based on constrained maximum common subgraph,” IEEE Trans. Autom. Sci. Eng., vol. 15, no. 2, pp. 692–704, Apr. 2018.

[7] L. Xiong, R. Molfino, and M. Zoppi, “Fixture layout optimization for f lexible aerospace parts based on self-reconfigurable swarm intelligent f ixture system,” Int. J. Adv. Manuf. Technol., vol. 66, nos. 9–12, pp. 1305–1313, Jun. 2013.

[8] S. Wang, Z. Jia, X. Lu, H. Zhang, C. Zhang, and S. Y. Liang, “Simultaneous optimization of fixture and cutting parameters of thin walled workpieces based on particle swarm optimization algorithm,” Simulation, vol. 94, no. 1, pp. 67–76, Jan. 2018.

[9] Q. Feng, W. Maier, T. Stehle, and H.-C. Möhring, “Optimization of a clamping concept based on machine learning,” Prod. Eng., vol. 16, no. 1, pp. 9–22, Aug. 2021.

[10] C. Cronrath, A. R. Aderiani, and B. Lennartson, “Enhancing digital twins through reinforcement learning,” in Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE). Washington, DC, USA: IEEE Computer Society, Aug. 2019, pp. 293–298.

[11] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning Series). Cambridge, MA, USA: MIT Press, 2018.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr1710

[12] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The sur prising effectiveness of PPO in cooperative, multi-agent games," in Proc. 36th Conf. Neural Inf. Process. Syst. New Orleans, LA, USA: Neural Information Processing Systems Foundation, Mar. 2022, p. 29.

[13] S. Lu, Y. Hu, and L. Zhang, "Stochastic bandits with graph feedback in non-stationary environments," in Proc. 35th AAAI Conf. Artif. Intell. (AAAI), vol. 10, 2021, pp. 8758–8766.

[14] R. Bogenfeld, C. Gorsky, and T. Wille, "An experimental damage tolerance investigation of CFRP composites on a substructural level," Compos. C, Open Access, vol. 8, Jul. 2022, Art. no. 100267.

[15] V. Mnih, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[16] S. Zamir, "Bayesian games: Games with incomplete information," in Encyclopedia of Complexity and Systems Science, R. A. Meyers, Ed., New York, NY, USA: Springer, 2009, pp. 426–441.

[17] S.Veeramani, S. Muthuswamy, K. Sagar, and M. Zoppi, "Artificial intel ligence planners for multi-head path planning of SwarmItFIX agents," J. Intell. Manuf., vol. 31, no. 4, pp. 815–832, Apr. 2020.

[18] J. Kudela and R. Matousek, "Recent advances and applications of surrogate models for finite element method computations: A review," Soft Comput., vol. 26, no. 24, pp. 13709–13733, Dec. 2022.