CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

ORCA – Online Research @ Cardiff

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository:https://orca.cardiff.ac.uk/id/eprint/147585/

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Veeramani, S. and Muthuswamy, S. 2022. Reinforcement learning based path planning of multiple agents of SwarmItFIX robot for fixturing operation in sheetmetal milling process. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 10.1177/09544054221080031

Publishers page: https://doi.org/10.1177/09544054221080031

Please note:
Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See http://orca.cf.ac.uk/policies.html for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.

Reinforcement learning based path planning of multiple agents of SwarmItFIX robot for fixturing operation in sheetmetal milling process

Abstract: SwarmItFIX (self-reconfigurable intelligent swarm fixtures) is a multi-agent setup mainly used as a robotic fixture for large Sheet metal machining operations. A Constraint Satisfaction Problem (CSP) based planning model is utilised currently for computing the locomotion sequence of multiple agents of the SwarmItFIX. But the SwarmItFIX faces several challenges with the current planner as it fails on several occasions. Moreover, the current planner computes only the goal positions of the base agent, not the path. To overcome these issues, a novel hierarchical planner is proposed, which employs Monte Carlo and SARSA TD based model-free Reinforcement Learning (RL) algorithms for the computation of locomotion sequences of head and base agents, respectively. These methods hold two distinct features when compared with the existing methods, (i) the transition model is not required for obtaining the locomotion sequence of the computational agent, and (ii) the state-space of the computational agent become scalable. The obtained results show that the proposed planner is capable of delivering optimal makespan for effective fixturing during the sheet metal milling process.

Keywords: Flexible fixturing systems, intelligent robotic fixtures, multi-agent robot systems, SwarmItFIX, model-free reinforcement learning.

List of notations

A	Action space of the base agent that consist of *(B0-B12)*
C_{cd}	Coordinate of the centroid of base position
N_b	Number of docking positions *(C1, C2, ..., C78)*
N_e	Number of edges that connects the docking pins
N_p	Number of docking pins *(P1, P2, ..., P52)*.
$P(s,a)$	Prospective new position attained by choosing action a in state s
$Q(s,a)$	Q value of the current state-action pair
$Q(s',a')$	Q value of the target state-action pair
$R(s,a)$	Reward function
S	State space of the base agent that consist of $(s_1$-$s_{78})$
T_1, T_2, T_3, T_4	Areas of triangles formed by the vertices e_1, e_2, e_3, e_4, and C_{cd}.
b_j, b_{j+1}	Goal positions of base agent
b_{curr}	Current position of the base agent
d_c^g	Euclidean distance between current state b_{curr} and goal state b_j
e_1,e_2,e_3,e_4	Vertices of the polygon used for identification of base goal position.
h_j, h_{j+1}	Support locations of head agent
(h_{1_x}, h_{1_y}), (h_{2_x}, h_{2_y})	Coordinates of the first and second support locations of head agent.
l_{arm}	Length of the serial arm of SwarmItFIX robot
n	Number of iterations required for the computation of optimal policy
r	Reward value obtained
s_d	Safest distance considered between trajectory and base
α	Step size parameter

β	Angle of the line segment connecting two corner positions of a polygon
γ	Discount factor
ε	Epsilon-greedy value
π	Policy
π^*	Optimal policy
τ	Expected return value of the state.

1. Introduction

Currently, manufacturing industries are equipped with advanced technologies of higher flexibility potential due to the emergence of mass customization.[1] Especially sheet metal machining processes are becoming effortless due to technological advancements such as advanced CNC machining centres, tube laser cutting. However, positioning and orienting large sheets of metals using a conventional fixture setup is cumbersome. Improper fixturing may cause sheet metal deformation, which results in serious dimensional inaccuracies during machining.[2] Camelio et al. (2004) have presented an optimization algorithm that proved to be reducing the dimensional inaccuracies by locating proper fixture positions for sheet metal assembly operations.[3] Automotive industries update their product design once in every model year, which mandates the industries every time to update their fixture setup as well.[4] This fixture upgrade process is not only expensive but also increases the lead-time of the manufacturing process. Thus, to avoid such problems, industrial robot manipulators equipped with proper end effectors can be designed as reconfigurable fixturing systems.[5] Such reconfigurable robotic fixturing systems are technically termed as Robot Fixtureless Assembly (RFA). This RFA is capable of grasping, immobilizing, positioning, and orienting varieties of parts just by changing software commands.[6] Also, RFA reduces the workpiece reconfiguration time, which ultimately leads to a reduction in lead-time of the manufacturing process. The initial installation cost of RFA is slightly high when compared with conventional fixture setup. But finally, this RFA reduces around 20% of the total fixturing cost [2], Improves autonomy [7], improves reconfigurability [8], and ultimately increases the throughput.[9,10] The workspace of the robot arm can be increased further by mounting it on the mobile platform. This facilitates the robotic machining of large-scale workpieces with higher positional accuracy. [11,12]

SwarmItFIX (Fig. 1) is one such patented mobile RFA setup that supports sheet metals at the bottom when machining happens on top of the metal sheet.[13,14] It follows the five-step sequence to traverse between any two support locations. These steps ensure coordination among head, PKM and mobile base agents of the SwarmItFIX, and each step corresponds to the sequence of locomotion of one agent. The novel Swing and Dock (SaD) locomotion strategy of the mobile base has already been explained in the previous works.[15-17] Head is a vacuum type of end effector used to support the sheet metal on its bottom surface. For the head to reach the specific

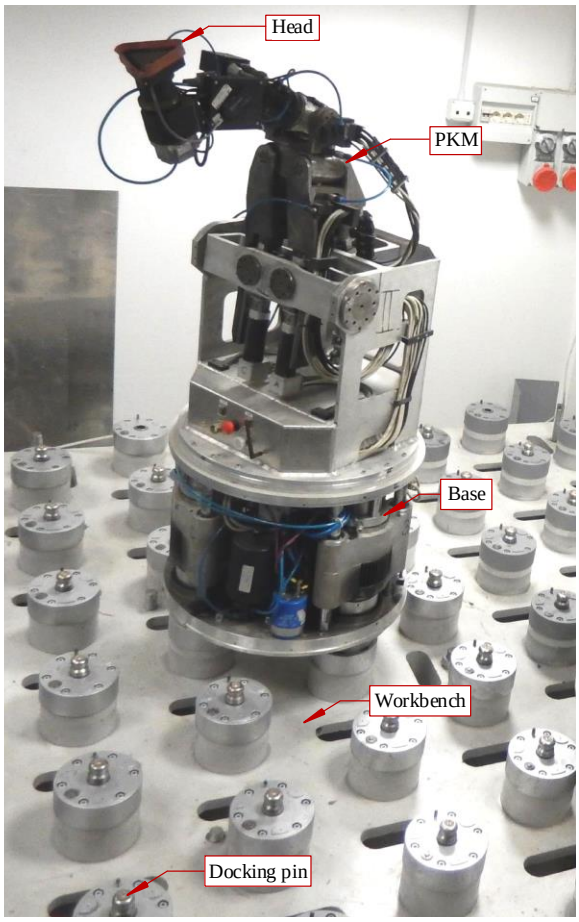


Figure 1. SwarmItFIX setup at the University of Genoa, Italy

coordinate within the workspace, the cooperation of all the three agents become essential. SwarmItFIX is a redundant manipulator with a 4R serial arm (R – Rotational) mounted on a Parallel Kinematic Machine (PKM) having three limbs. Two limbs are with the kinematic architecture of RRPR, and the other is with SPR (P - Prismatic, S - Spherical) configuration. The design is in such a way that the PKM is to accomplish the positioning task, whereas the serial arm is to orient the end-effector head. The PKM with the serial arm provides six Degrees of Freedom (DOF) pose together. The SwarmItFIX head has a dedicated motor and hence, an independent DOF to accommodate the orientation of the head along with the sheet metal. This enables de-coupling in inverse kinematics, where only the six DOF pose of the end-effector vacuum head is mainly required and hence, orientation can be achieved independently. The detailed kinematic analysis of the PKM has already been done. For the PKM, only the 6-dof pose/Euler angles of the end-effector (head) with respect to the base are required to achieve a feasible path based on the methodology proposed. Once target orientation of the head and position of the base are obtained, the motion trajectory of the PKM can be generated between the two positions based on the inverse kinematics model.[18] The entire manipulator is mounted on a mobile base with a harmonic drive to provide an additional DOF of rotation. Hence, the kinematic structure of

the SwarmItFIX robot is specially designed to be redundant to provide at least one solution to the motion planning problem. Various heuristic approaches are proved to be promising in the robot path planning and domain.[19] But the problem is that these methods do the computation again when the same goal positions are repeated, i.e. no policies are stored.

Reinforcement Learning is a viable alternative for the conventional techniques in the path planning domain. RL overcomes shortcomings such as replanning and detour. This technique uses the rewards observed by the computational agent while taking appropriate action in each state to learn the optimal (or nearly optimal) policy. Then, using this optimal policy, the sequence of operations performed by the physical agent can be computed.[20] Initially, the computational agents would not have any knowledge about the optimal policy, but they learn the same by interacting with the computational environment.[21,22] The skills acquired by the computational agents can be evaluated by using the intrinsic motivations in the RL framework.[23] This optimal policy learned by the computational agent can be transferred to the robots for its optimal locomotion.[24,25] The authors have utilized two different algorithms, viz, extended single-agent Q-learning algorithm and Team Q-learning algorithm, for a fully cooperative multi-robot box pushing task.[26] Though single-agent Q-learning is developed for static environments, it still copes with the dynamic environment (environment with more than one agent) and offers positive and better results when compared with the Team Q-learning algorithm. The same authors have also presented Sequential Q-learning with Kalman Filtering (SQKF) algorithm, which is a modified version of the distributed Q-learning algorithm for the same task.[27] SARSA obtains better state estimates for mobile robot delivery tasks when compared with other RL approaches as confirmed by Ramachandran and Gupta (2009).[28]

Numerous ways have been explored by the researchers for solving inverse kinematics of the robot mechanisms. The inverse kinematic analysis of any parallel manipulators which resembles the Stewart platform can be effectively performed using homogenous transformations. Also, the solvers which utilize multidimensional geometric information of the mechanism, efficiently solve inverse kinematics problems. The geometric constraints of the mechanisms can be satisfied by devising a fixed sequence of actions which ultimately reduces the degree of freedom of a mechanism. [29,30]

The objectives of the present work are;

i) to develop a hierarchical RL based path planner for the path planning of head and base agents of SwarmItFIX,

ii) to develop a methodology for the identification of goal positions of the SwarmItFIX-base in line with all support locations,

iii) to formulate the path planning problem of SwarmItFIX base agent as MDP for a fixed-size environment that is scalable for the larger size environments,

iv) to propose a unique reward function that enables the computational agent to explore the environment,

v) to identify and utilize a suitable model-free reinforcement learning technique for solving the formulated MDP, and

vi) to test the proposed planner with various tool trajectories.

The rest of this paper is organized as follows. The research gap and problem definition are detailed in section 2. The five-step locomotion strategy is explained in section 3. Details of MDP modelling and the proposed RL based offline architecture is presented in section 4. The results are discussed in section 5. Conclusion statements and details of future works are described in section 6. followed by references.

2. Research gap and problem definition

Based on the detailed literature survey, it has been confirmed that,

- To consecutively provide support throughout the machining process, the agents of the robot must coordinate in a sequential manner. The existing hierarchical CSP approach which performs this coordination has the following issues that need to be addressed.[31,32]
  - i. The hierarchical CSP approach only computes the goal positions of the SwarmItFIX base and not the sequence of locomotion. This is with the condition that the two consecutive goal positions should have a common docking pin; otherwise, the planner would fail.
  - ii. When the base CSP fails to identify such goal positions, it tracks back and forces to replan to find a new head position. This replanning procedure causes unnecessary detours.
- To overcome these issues, a planner with the liberty of choosing the goal positions more flexibly and capable of computing the locomotion sequence of SwarmItFIX base need to be developed.
- The action sequence of the SwarmItFIX base agent has to be computed with reduced makespan and minimum learning time and without any backtracking or replanning. The term makespan here denotes the number of steps required to be performed by the SwarmItFIX for accomplishing the given task (for reaching the goal

position). The minimal makespan would facilitate the machining with a higher feed rate, which ultimately increases the throughput of the system.

3. The five-step locomotion of SwarmItFIX

In order to traverse between two support locations, all agents of the SwarmItFIX robot have to coordinate by performing the locomotion of each agent in a sequential manner. The mobile base acts as a primary locomotion source which traverses the robot to various locations within the workbench. When the robot reaches the defined location, the mobile base locks its position to support the workpiece. Now, the PKM positions the equilateral triangle shaped head in the defined support location along with the required orientation. The PKM is linked with a 3 DOF 3R spherical wrist which gives the additional DOF needed for the orientation of the head. Let (h_j, b_j) and (h_{j+1}, b_{j+1}) be the position of the head (h) and base agent (b) at support locations j and $j+1$, respectively (Fig. 2 and Fig. 4). The transition (positioning alone) between h_j and h_{j+1} is due to the movement of PKM and base. The associated orientation is taken care of by the independent rotational motion of the head alone. Starting from the j^{th} supporting position (h_j, b_j), the SwarmItFIX robot performs the following five-step transitions and reaches the next supporting position (h_{j+1}, b_{j+1}).

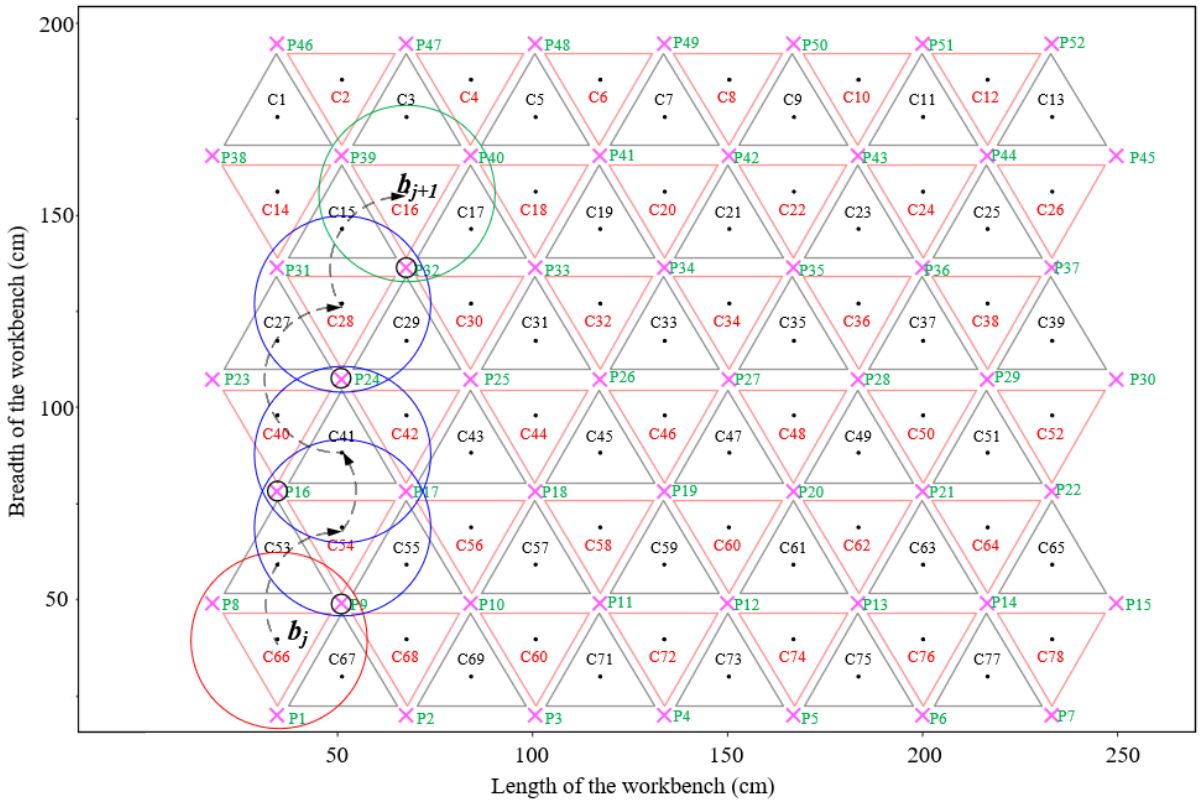


Figure 2. SaD locomotion of the base agent (The positions sequence: C66→C54→C41→C28→C16; sequence of docked pins during locomotion: P9→P17→P24→P32)

- Transition 1: Retraction of the head away from current support location of the work-piece (h_j).
- Transition 2: Movement of the head agent to parking position to avoid collision with other agents.
- Transition 3: Locomotion of base ($b_j \rightarrow b_{j+1}$) along with the simultaneous counter-rotation of the PKM (Fig. 2). As the PKM is already parked safely, the collision-free locomotion of the whole manipulator at this stage is completely ensured.
- Transition 4: Movement of head to a nearby position which is a normal approach away from next support location h_{j+1}.
- Transition 5: Movement of the head to the next support location of the workpiece h_{j+1}.

4. Reinforcement Learning based Path planner module

This section presents the proposed planner (Fig. 3), which works based on the coordination of various agents of the SwarmItFIX robot.

4.1. The proposed path planner module

The proposed path planner module developed based on Reinforcement Learning for the coordination of multiple agents of the SwarmItFIX robot is shown in Fig. 3. This path planner basically comprises two major modules; the head planning module along with the Monte Carlo RL module, which was proposed earlier and the newly proposed base planning module along with the SARSA RL module.

4.1.1. Workpiece module

The workpiece module gets executed once for each tool trajectory. It analyses the tool trajectory obtained from CAM software, and splits the complete trajectory into various linear segments and then calculate the angular information of all segments and these are called vertex angles. This module also calculates the number of vertices present along the trajectory and their type and number of segments.

4.1.2. Multi-agent RL module

Based on the data received from the previous module, this multi-agent module computes the position and orientation of vertex heads initially and then proceeds to the head planning submodule. This submodule computes the position and orientation of intermediate head positions as given in Fig.3.

Base planning module

The first step in this module is the identification of goal positions (b_j) for all head support locations (h_j) in the workbench (Fig. 4). Hence, the coordinates of h_j are utilized for the computation of goal positions. In this regard,

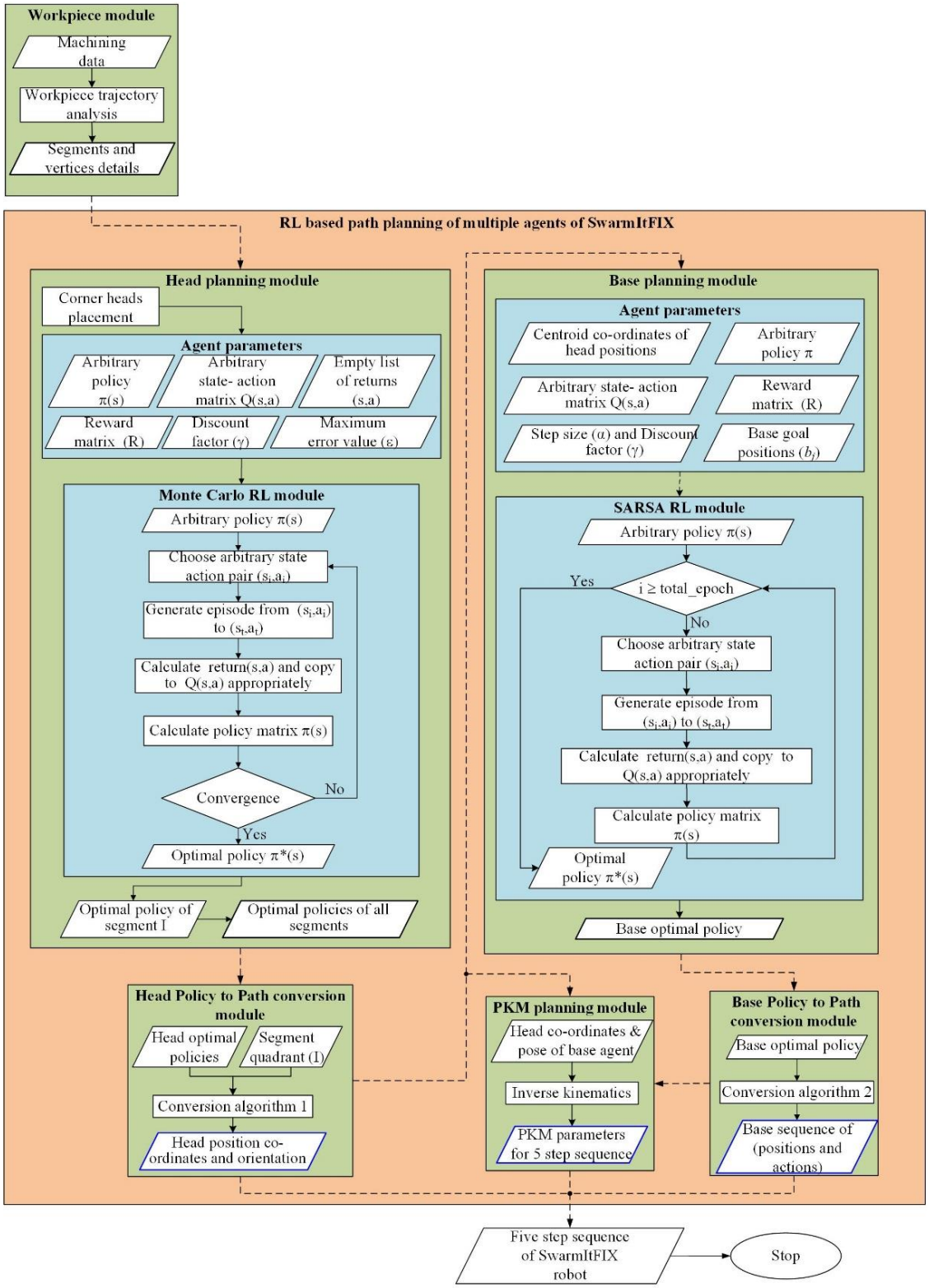


Figure 3. RL based offline path planner

initially, a polygon is created with projections of centroids of head support locations 1 (h_1) and 2 (h_2) on the workbench as the two inner corners e_1 and e_2, respectively. The two outer corners of the polygon e_3 and e_4 are

calculated using Eq. (1) and Eq. (2). Finally, the base positions in the workbench, which satisfies the constraints of Eq. (3) and Eq. (4), are considered as the goal positions (b_j) of the base agent.

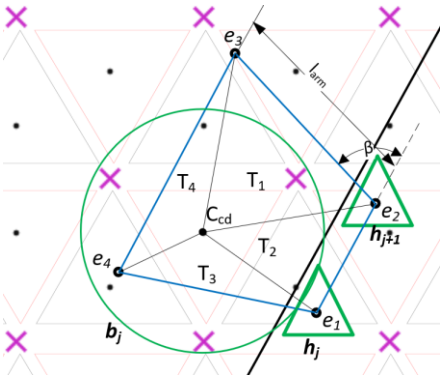


Figure 4. Identification of base goal positions

$$e_3 = \begin{Bmatrix} h_{2\_x} + [l_{arm} \times cos(\beta)], \\ h_{2\_y} + [l_{arm} \times sin(\beta)] \end{Bmatrix} \quad (1)$$

$$e_4 = \begin{Bmatrix} h_{1\_x} + [l_{arm} \times cos(\beta)], \\ h_{1\_y} + [l_{arm} \times sin(\beta)] \end{Bmatrix} \quad (2)$$

$$area(e_1, e_2, e_3, e_4) \geq sum(T_1, T_2, T_3, T_4) \quad (3)$$

$$dist(e_1, C_{cd}) \geq s_d \;\&\; dist(e_2, C_{cd}) \geq s_d \quad (4)$$

where, C_{cd} is the coordinate of the centroid of base position. T_1, T_2, T_3, and T_4 are the areas of triangles formed by the vertices e_1, e_2, e_3, e_4, and C_{cd}. s_d is the safest distance considered between trajectory and base, and it will act as an important parameter to avoid collision between two base agents. l_{arm} is the length of the serial arm. β is the angle of the line segment connecting two corner positions of a polygon.

Now for making the base agent traverse between the current position (b_{curr}) and the goal position (b_j), a policy that is capable of suggesting an optimal sequence of actions with the objective of minimising the makespan is required. Basically, the base agent traverses on the modular workbench, where its dimensions can be scaled based on the size of the workpiece. Hence, this path planning problem is modelled as Markov Decision Process (MDP), and the SARSA model-free RL algorithm is utilized to solve it.

SARSA Temporal Differencing (TD) algorithm is defined by the quintuple $<s_i,a_i,r_i,s_{i+1},a_{i+1}>$ that perform transitions between i^{th} state-action pair and $i+1^{th}$ state-action pair and learns the state-action values $Q(s,a)$. Whereas, in the other model-free RL algorithms, such as MC control, the transitions are performed between two different states, s_i and s_{i+1}. Various agent parameters used in the SARSA base planning module are:

State space ($s \in S$)- As shown in Fig. 2, the workbench consists of 52 docking pins (N_p) and each of which combines with neighbouring docking pins and form triangular configurations of 129 edges (N_e) in total. Using Euler's relation Eq. (5), the number of possibilities of obtaining the base position (N_b) is found to be 78. Thus, the state space S contains 78 states in a pattern of a triangular grid which comprises 6 rows and 13 columns.

$$N_p - N_e + N_b = 1 \tag{5}$$

Action space ($a \in A$)- Each state shares the docking pins with 13 neighbouring states. Hence, from any particular state, an agent could perform 13 different actions (B0-B12) to reach the neighbouring states. Out of these 13 actions, B11 and B12 are not considered in the action space due to the following reasons.

- When these two actions are performed on an odd-numbered state, the computational agent reaches $b_{curr}+11$ and $b_{curr}+15$. Whereas, if it is performed on an even-numbered state, it reaches two different states, $b_{curr}-11$ and $b_{curr}-15$.
- These states can also be reached by performing the other actions twice consecutively.

Hence, for the locomotion of the base agent, 11 different actions (B0-B10) can be assigned. Fig. 5 describes the details of action space. The details of corresponding locomotion for all the actions are presented in Table 1.

Reward matrix R(s,a)- The reward matrix is constant throughout the process, and it is calculated as below.

$$R(s,a) = \begin{cases} (1 - d_c^g), if\ P(s,a) = b_j \\ -d_c^g, otherwise \end{cases} \tag{6}$$

$$d_c^g = \|b_{curr} - b_j\| \tag{7}$$

where, d_c^g is the euclidean distance between current state b_{curr} and goal state b_j. *P(s,a)* is the prospective new position attained by choosing action *a* in state *s*.

State-action matrix Q(s,a)- The size of the state action matrix is 11 x 78 (actions x states).

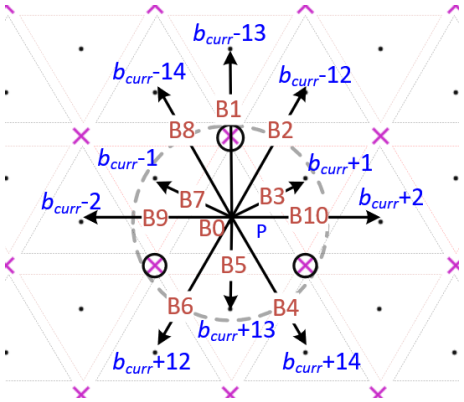


Figure 5. Action space

Table 1. Actions and locomotion mapping

Sl. No.	Action	Base locomotion						Next position
		current state - even ▽			current state - odd △			
		Leg	Direction	Angle, deg.	Leg	Direction	Angle, deg.	
1	B0	No locomotion and stay in the current position						$*b_{curr}$
2	B1	1	CCW	60	2	CCW	180	b_{curr} +13
3	B2	3	CW	120	2	CCW	120	b_{curr} +14
4	B3	2	CW	60	2	CCW	60	b_{curr} +1
5	B4	2	CW	120	3	CCW	120	b_{curr} -12
6	B5	2	CCW	180	1	CW	60	b_{curr} -13
7	B6	2	CCW	120	1	CW	120	b_{curr} -14
8	B7	1	CW	60	1	CCW	60	b_{curr} -1
9	B8	1	CCW	120	2	CW	120	b_{curr} +12
10	B9	1	CW	60	1	CCW	120	b_{curr} -2
11	B10	3	CCW	120	3	CW	120	b_{curr} +2

CW = Clockwise, CCW = Counter clockwise, $*b_{curr}$ =Current position.

Step size (α)- It is a factor ($0 < \alpha \leq 1$) that decides the amount of error (the difference between the target utility and old estimated utility) that has to be integrated into the new estimate of Q-value. The step size is also called the learning parameter. When α is 0, then the agent does not learn at all, whereas if α is 1, the agent highly considers the error value and learns.

Discount factor (γ)- It plays a major role in choosing between the current rewards and future rewards. It also varies similar to α. When γ gets close to 0, the future rewards become negligible. When γ gets close to 1, more importance will be assigned to future rewards.

The operation of the SARSA algorithm (Table 2) begins with choosing the arbitrary state-action pair (s_i,a_i), and the state s_i will be the initial state of the episode. The corresponding Q-value will be calculated using Eq. (10) and will be updated in the state-action matrix $Q(s,a)$ against the corresponding initial state-action pair (s_i,a_i). Finally, the optimal policy (π^*) obtained will provide an efficient base action sequence for all the support locations of the tool trajectory.

The expected return (τ) of the state can be calculated using Eq. (8).

$$\tau = E_\pi \left[\sum_{j=0}^{\infty} \gamma^j r_{i+j+1} \right] \tag{8}$$

which can be further simplified as under

$$\tau = E_\pi \left[r_{i+1} + \gamma r_{i+2} + \gamma^2 r_{i+3} + ... + \gamma^j r_{i+j+1} \right]$$

$$\tau = E_\pi \left[r_{i+1} + \gamma \left(r_{i+2} + \gamma r_{i+3} + ... + \gamma^{j-1} r_{i+j+1} \right) \right]$$

$$\tau = E_\pi \left[r_{i+1} + \gamma Q\left(s_{i+1}, a_{i+1}\right) \right] \tag{9}$$

The update rule for the SARSA-TD control algorithm is given in Eq. (10).

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\left[r_{i+1} + \gamma Q(s', a') - Q(s_i, a_i)\right] \quad (10)$$

r_{i+1} defines the reward obtained at the state s_{i+1}, j is the weightage parameter of the discount factor for the future rewards. Based on the calculated state-action matrix $Q(s,a)$, the policy matrix will also get updated.

4.1.3. Base policy to path conversion (BPP) module

Given the respective start positions, this BPP module gathers about π^* from the base planning module and converts them into a sequence of states and actions. Initially, for the first sequence, the home position will act as the start position. Now, this module refers to the first optimal policy and computes the sequence of actions. The base agent needs to perform this sequence in order to reach the goal position (b_1). Similarly, in the second sequence, positions b_1 and b_2 are the start and goal positions. This process gets iterated until the sequence for all the goal positions (b_j) are obtained.

Table 2. SARSA TD control algorithm

- ***Input:***
  - Step size $\alpha \in (0,1]$, small $\varepsilon > 0$
  - $\pi(s) \in A(s)$ (arbitrarily), for all $s \in S$
  - Q (s, a) $\in \mathbb{R}$ (arbitrarily), for all $s \in S$, $a \in A(s)$
  - Returns (s, a) $\leftarrow$ empty list, for all $s \in S$, $a \in A(s)$
- ***Output:*** Optimal policy π^*, updates Q(s, a)
- ***Loop for each episode:***
  - Initialize $s_i \in S$
  - Choose $a_i \in A(s_i)$ using an epsilon greedy policy.
  - ***Loop for each step of the episode***
    - Take action a_i, observe r, s'
    - Choose $a' \in A(s')$ using an epsilon greedy policy
    - $Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\left[r_{i+1} + \gamma Q(s', a') - Q(s_i, a_i)\right]$
    - $s_i \leftarrow s'$; $a_i \leftarrow a'$
  - Until *s* is terminal
- ***Loop for all states***
  - $\pi(s_t) = \arg\max_{a_i \in A(s)} Q(s_i, a_i)$
- ***Return π^****

4.1.4. PKM planning module

As mentioned earlier, the PKM module is equipped with a fixed sequence of actions. For all the goal states of the head-base pair, the PKM traverses in a fixed sequence and facilitates the head agent to reach the goal state. Given the target position of the head, by using the existing inverse kinematic model, the state variables of the PKM can be calculated.

5. Results and discussion

Simulation of multi-agent path planning of the robot is carried out using a novel RL based planner proposed in this work. The configuration of the system used for simulation includes Intel Xeon CPUE3-1225v6 Processor 3.30 GHz, Windows 10, 64-bit OS with 16.0 GB RAM. Mathematical models are implemented using Python 3.7 idle interface. The results obtained are grouped under three categories and presented as follows.

5.1. Estimation of RL parameters

The path planning problem of the base agent of the SwarmItFIX robot is modelled as Markov Decision Process (MDP), and model-free Reinforcement Learning (RL) algorithm viz., SARSA TD control is deployed to solve the MDP. Initially, the optimum values of various parameters of the SARSA algorithm are estimated as detailed below.

- Five different test cases, C1 as goal position and C20, C6, C35, C52, and C78 are the various start positions are considered. Hence, the distance between the start position and the goal position in all the cases are different from each other.
- Initially, a simulation experiment is executed for finding the optimal discount factor (γ) within the range 0.8 $\leq \gamma <$ 1 based on literature. The corresponding result obtained is shown in Fig. 6a. When $\gamma <$ 0.96, the computational agent could not reach the goal position even after 1000 steps for cases with higher distances between start and goal positions (C52 and C78). The process also gets terminated for such cases. For the other three cases (C20, C6, and C35 as start positions), the computational agent reaches the goal position.

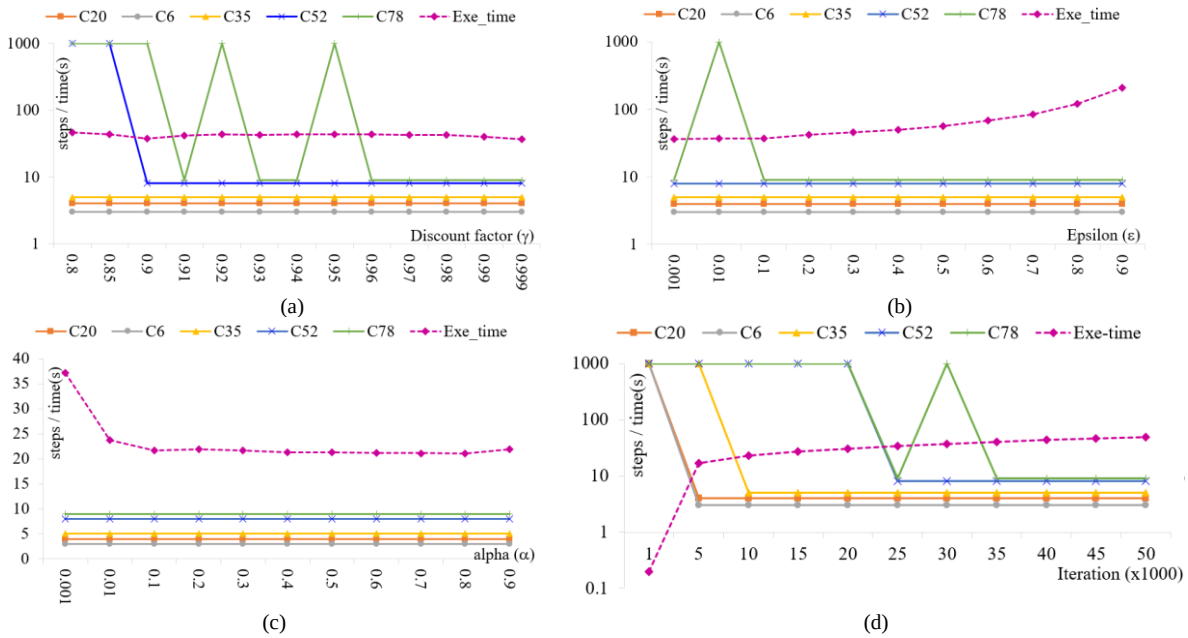


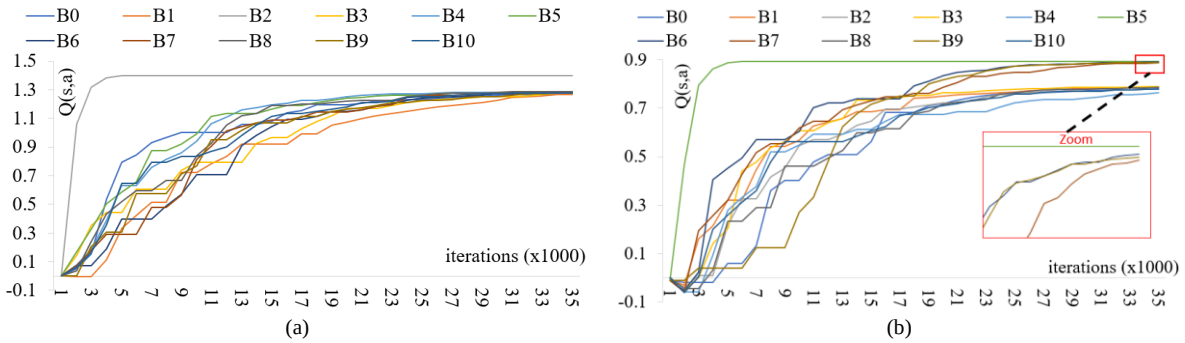
Figure 6. RL parameters estimation: (a) discount factor (b) greedy value (c) step size and (d) number of iterations.

- By increasing the γ value further, the computational agent finds the goal position for all the test cases. But when γ is close to 1, the goal position is attained in all the cases with the least execution time (37.15s).
- Similarly, experiments are carried out for finding optimal step size (α), greedy value (ε), and the number of iterations (n) and the associated results are presented in figures 6b, 6c, and 6d, respectively. The optimal values found are γ=0.999, α = 0.1, ε=0.1, and n=35000.

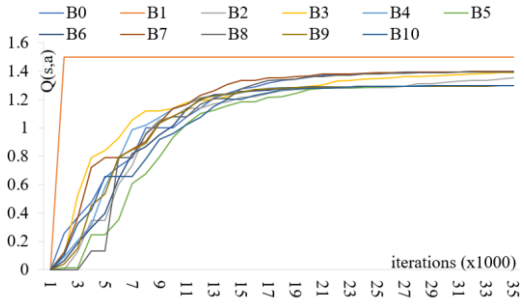
5.2. Convergence of optimal policies

The results obtained while executing the SARSA algorithm for base agent planning of the SwarmItFIX robot are discussed now. The goal positions of the base agent are calculated by applying Eq. (1) and Eq. (2), and the total number of goal positions (b_i) is calculated as sixteen for the tool trajectory.

- The base agent, from its start position, performs a sequence of actions and reaches each goal position. As the trajectory contains 16 goal positions, the base agent needs to perform 16 sequences to cover the complete trajectory. Thus, 16 different optimal policies are required for the base agent. For the first sequence, the home position C66 become the start position, and C42 become the corresponding goal position (b_1). Similarly, C42 and C29 are the start and goal (b_2) positions, respectively, for the second sequence.
- The three plots in Fig. 7 show the Q-values of all 11 actions of 5 different states. Fig. 7a and Fig. 7b corresponds to the first sequence (π_1^*) whereas, Fig. 7c correspond to the second sequence (π_2^*). From Fig. 7a, the Q-values of all 11 actions that belong to state C66 can be observed. Whereas Fig. 7b and Fig. 7c are related to C13, and C42 respectively.
- Further, it can be observed in the first sequence at state C66 (Fig. 7a) that, when the iteration progresses, the Q-value of action B2 increases at a faster rate when compared with the Q-values of the rest of the actions. This clearly shows that in the first sequence at C66, B2 is the optimal action that is recommended for the base agent. Similarly, optimal action for C13 (first sequence) and optimal action for C42 (second sequence) can be observed from Fig. 7b and Fig. 7c, respectively.



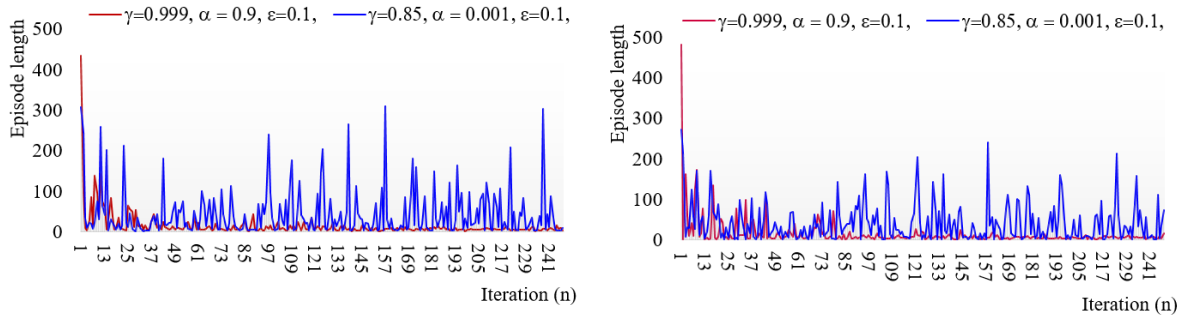
(a) (b)



(c)

Figure 7. Q-values vs actions: (a) optimal policy 1 state C66 (b) optimal policy 1 state C13 and (c) optimal policy 2 state C42

- From Fig. 7c, it can be observed that action B1 is immediately converged as an optimal action when compared with the actions B2 and B5, as shown in figures 7a and 7b, respectively. This is due to the fact that, in the second sequence, the state C29 is the start position which is a neighbouring state of goal position C42. Hence, the distance d_c^g is comparatively lesser. Interestingly, higher d_c^g develops the possibility of more than one optimal action for a particular state. This is evident from Fig. 7b that the Q-values of actions B5, B6, and B7 are very close to each other.
- The convergence of the optimal policy for the first two sequences is shown in Fig. 8a and Fig. 8b by plotting the step length of the episodes of all the iterations. For both sequences, the path planner has been executed with two different sets of RL parameters. Set 1 comprises of identified optimal values $\gamma = 0.999$, $\alpha = 0.1$, $\varepsilon = 0.1$, whereas, the corresponding values for set 2 are $\gamma=0.85$, $\alpha = 0.001$, $\varepsilon=0.01$. In the first iteration of both sets, the computational agent starts from a random state, follows a non-optimal policy, and takes more than 500 steps to reach the goal state. But, when the iteration progress, the policy converges gradually and becomes optimal. Finally, the agent reaches the goal position in fewer steps. It can also be observed from both the sequences that set 1 with the optimal parameters converge much faster than that of set 2.



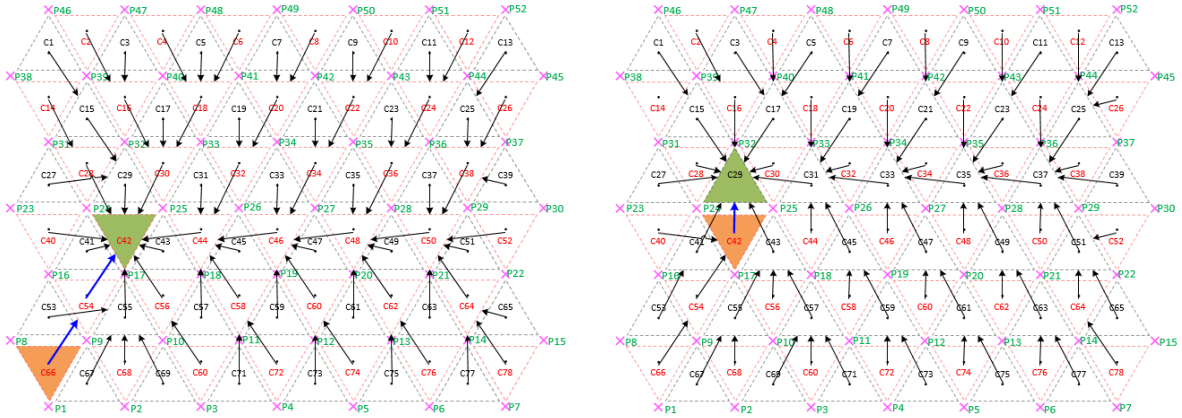
(a) (b)

Figure 8. Learning curve: (a) first sequence and (b) second sequence

5.3. Execution on various tool trajectories

The optimal policies for both first and second sequences are shown in Fig. 9a and Fig. 9b, respectively.

- The arrows shown at the states indicate the action of the corresponding state recommended by the optimal policy (π*). States C42 and C29 become the goal positions of the first sequence and second sequence, respectively.



(a) (b)

Figure 9. Optimal policy: (a) π_1^* and (b) π_2^*

- When starting from any state with the help of these optimal policies, the sequences for reaching the corresponding goal positions can be obtained in an efficient manner. Finally, when all sequences are combined for completing the trajectory, the agent performs 23 locomotion steps starting from the home position to the last goal position while passing through all intermediate goal positions.
- The details of actions performed and rewards earned by the agent at each locomotion are depicted in Fig. 10. The planning model has also been tested with various trajectories, and the results are shown in Fig. 11. It confirms the placement of both head and base agents based on the outcome of the planner. In the placement of the base agent, the continuous circles (green) represent the goal positions of the base agent, whereas the discrete circles (blue) become the intermediate positions. The lines (red) that connect the base goal positions and head positions denote the active states of PKM and head agents.
- Table 3 compares the SARSA algorithm with the baseline Policy Iteration (dynamic programming) algorithm in terms of computation time, makespan, and the number of iterations iteration for all the 16 sequences of the polygonal trajectory shown in Fig. 11c. It can be clearly observed that the SARSA algorithm produces optimal results in terms of makespan.

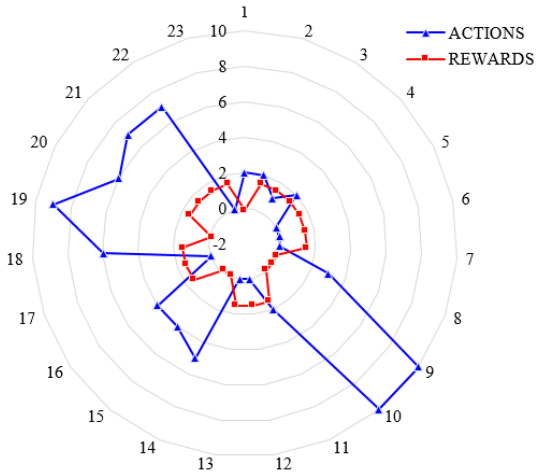
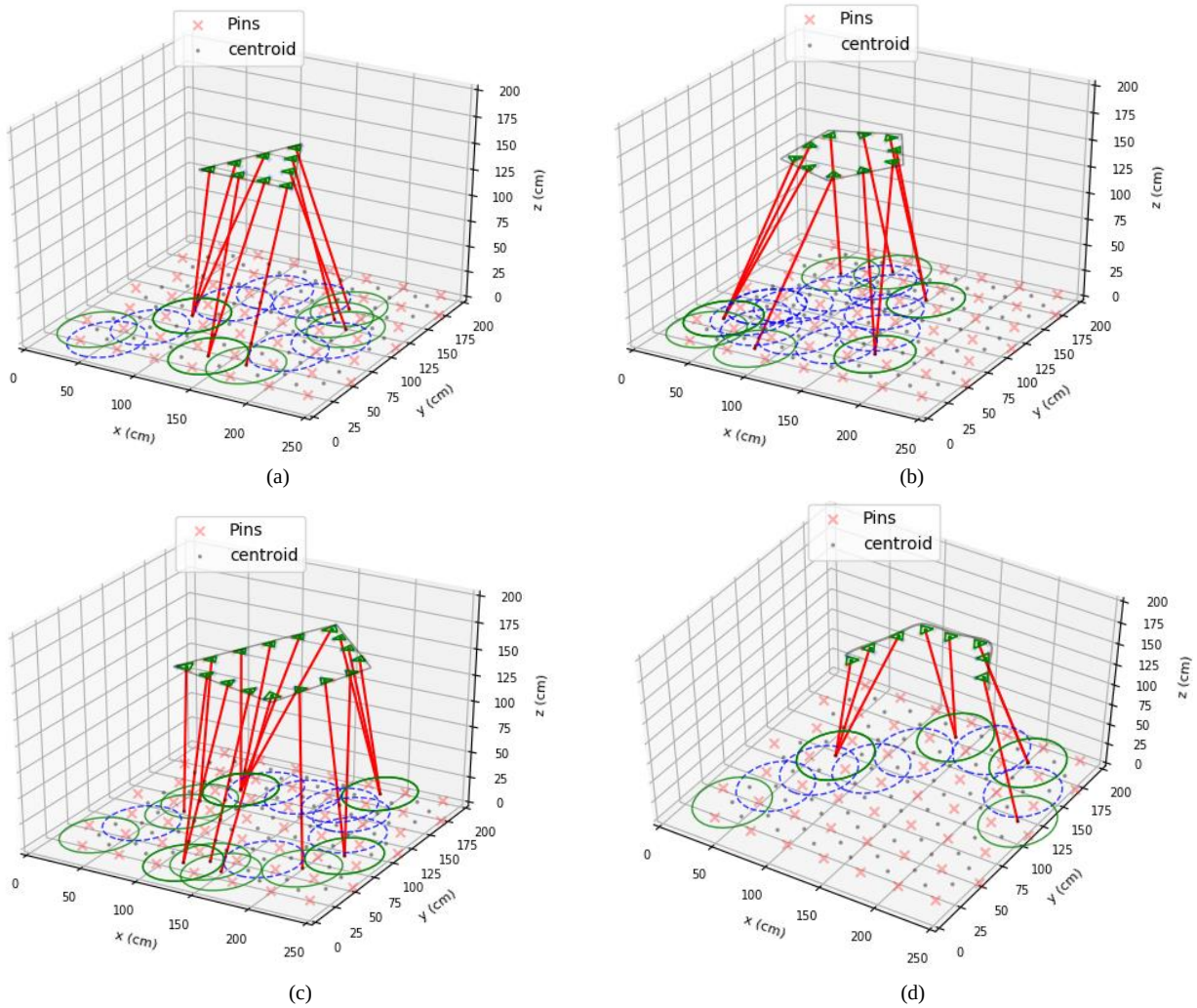


Figure 10. Actions and rewards obtained by base



(a) (b)

(c) (d)

Figure 11. Head and base placement on other tool trajectories: (a) triangular trajectory (b) pentagonal trajectory (c) polygonal trajectory and (d) open curved trajectory

Table 3. Comparison of results based on polygonal trajectory

Sequence	Parameters					
	Execution Time (s)		Makespan		Iterations	
	SARSA TD	PI	SARSA TD	PI	SARSA TD	PI
1	17.49±0.36	0.3±0.012	2	2	35000	9
2	17.53±0.29	0.3±0.011	1	1	35000	9
3	17.69±0.15	0.3±0.319	1	1	35000	9
4	17.69±0.15	0.2±0.012	0	0	35000	8
5	17.69±0.15	0.2±0.014	0	0	35000	8
6	17.69±0.15	0.3±0.009	0	0	35000	9
7	18.31±0.21	0.2±0.010	4	4	35000	8
8	18.31±0.21	0.2±0.008	0	0	35000	8
9	18.31±0.21	0.2±0.011	0	0	35000	8
10	20.27±0.63	0.3±0.010	3	3	35000	10
11	20.27±0.63	0.3±0.028	0	0	35000	10
12	17.96±0.32	0.3±0.017	2	2	35000	9
13	19.15±0.23	0.2±0.012	2	2	35000	8
14	18.05±0.34	0.2±0.014	1	1	35000	8
15	17.82±0.34	0.2±0.006	1	1	35000	8
16	17.82±0.34	0.2±0.014	0	0	35000	8

6. Conclusion and future work

In this work, a model-free RL based path planner is developed and presented for the coordination of multiple agents of a SwarmItFIX robotic fixture. Based on the results obtained, the following conclusions are drawn.

- The proposed path planner is found to be an efficient in terms of achieving the coordination among multiple agents of a single SwarmItFIX robot. The disadvantage of having a replanning procedure present in the previous CSP approach has been avoided by the proposed SARSA TD based planner. Hence, replanning and detour are eliminated.
- Initially, the Monte Carlo based RL algorithm was deployed for the computation of base agent optimal policy for 5 test cases, as shown in Fig.6. Though there is no difference in the optimal policies produced by both the algorithms, the SARSA algorithm computes the optimal policy nine times faster than the MC algorithm (360s vs 40s). Hence, the major part of this research work is carried out based on the SARSA algorithm.
- The optimal value of the parameters used to conduct the experiment are found to be γ *(discount factor)* $= 0.999$, α *(step-size)* $= 0.1$, ε *(greedy value)* $= 0.1$, and number of iterations = 35000. With these optimal values, the SARSA TD algorithm has been executed further for obtaining the desired action sequence for various trajectories, as shown in Fig. 11. Finally, the results obtained are compared with the results of the baseline model-based Policy Iteration (PI) algorithm, as presented in Table 3.

- The proposed SARSA based model works well with all the test cases of simple machining trajectories, as shown in Fig. 11. A compatibility check of the proposed planner for complex curved trajectories is proposed for future research. The possibilities of utilization of the SwarmItFIX robot as a robotic fixture for other machining operations will also be investigated.
- Multi-robot planning is more complex when compared with the planning of multiple agents of one SwarmItFIX robot. This is because the environment becomes dynamic, and the locomotion of one robot becomes the moving obstacle for the other. Thus, in future, the same model-free RL based planning scheme will be extended further to solve this dynamic multi-robot path planning problem.

Acknowledgement

This work is an outcome of the sponsored project (Project No: MEC/DIMEUG-ITALY/2015-16/MSK/008/02) with a financial grant from the University of Genoa, Italy. The support being received from Prof. Matteo Zoppi, Prof. Rezia Molfino, and Dr. Keerthi Sagar of PMAR Research Group @ The University of Genoa is highly acknowledged.

Declaration

The Author(s) declare(s) that there is no conflict of interest.

References

1. Michalos G, Makris S, Papakostas N, et al. Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach. CIRP J Manuf Sci Technol 2010; 2: 81–91.
2. Cai W, Hu SJ, Yuan JX. Deformable Sheet Metal Fixturing: Principles, Algorithms, and Simulations. *J Manuf Sci Eng* 1996; 118: 318–324.
3. Camelio JA, Hu SJ, Ceglarek D. Impact of fixture design on sheet metal assembly variation. *J Manuf Syst* 2004; 23: 182–193.
4. Arzanpour S, Fung J, Mills JK, et al. Flexible fixture design with applications to assembly of sheet metal automotive body parts. *Assem Autom* 2006; 26: 143–153.
5. Bi ZM, Zhang WJ. Flexible fixture design and automation: Review, issues and future directions. *Int J Prod Res* 2001; 39: 2867–2894.
6. He S, Deng Y, Yan C, et al. A tolerance constrained robot path circular interpolation method for industrial SCARA robots. *Proc Inst Mech Eng Part B J Eng Manuf* 2021; 235: 1061–1073.

7. Sahu OP, Biswal BB, Mukherjee S, et al. Multiple Sensor Integrated Robotic End-effectors for Assembly. *Procedia Technol* 2014; 14: 100–107.
8. Makris S, Michalos G, Eytan A, et al. Cooperating Robots for Reconfigurable Assembly Operations: Review and Challenges. Procedia CIRP 2012; 3: 346–351.
9. Foumani M, Smith-Miles K, Gunawan I. Scheduling of two-machine robotic rework cells: In-process, post-process and in-line inspection scenarios. *Rob Auton Syst* 2017; 91: 210–225.
10. Huang J, Pham DT, Wang Y, et al. A case study in human–robot collaboration in the disassembly of press-fitted components. *Proc Inst Mech Eng Part B J Eng Manuf* 2020; 234: 654–664.
11. Tao B, Zhao X, Yan S, et al. Kinematic modeling and control of mobile robot for large-scale workpiece machining. Proc Inst Mech Eng Part B J Eng Manuf 2020; 1–10.
12. Ding Y, Zhang Z, Liu X, et al. Development of a novel mobile robotic system for large-scale manufacturing. Proc Inst Mech Eng Part B J Eng Manuf 2021; 235: 2300–2309.
13. Sagar K, de Leonardo L, Molfino R, et al. The SwarmItFix Pilot. *Procedia Manuf* 2017; 11: 413–422.
14. Molfino RM, Zoppi M, Zlatanov D. *Bench and method for the support and manufacturing of parts with complex geometry*. 8495811B2, US Patent, 2013.
15. Veeramani S, Muthuswamy S, Sagar K, et al. Artificial intelligence planners for multi-head path planning of SwarmItFIX agents. *J Intell Manuf* 2020; 31: 815–832.
16. de Leonardo L, Zoppi M, Xiong L, et al. SwarmItFIX: a multi-robot-based reconfigurable fixture. *Ind Robot An Int J* 2013; 40: 320–328.
17. Veeramani S, Muthuswamy S, Sagar K, et al. Multi-Head Path Planning of SwarmItFIX Agents: A Markov Decision Process Approach. In: *Mechanisms and Machine Science*. Springer, Cham, pp. 2237–2247.
18. Zoppi M, Zlatanov D, Molfino R. Kinematics Analysis of the Exechon Tripod. In: *Volume 2: 34th Annual Mechanisms and Robotics Conference, Parts A and B*. ASMEDC, pp. 1381–1388.
19. Mac TT, Copot C, Tran DT, et al. Heuristic approaches in robot path planning: A survey. Rob Auton Syst 2016; 86: 13–28.
20. Sutton RS, Barto AG. *Reinforcement Learning, An Introduction*. The MIT Press, 2018.
21. Wang B, Liu Z, Li Q, et al. Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning. *IEEE Robot Autom Lett* 2020; 5: 6932–6939.

22. Goharimanesh M, Mehrkish A, Janabi-Sharifi F. A Fuzzy Reinforcement Learning Approach for Continuum Robot Control. *J Intell Robot Syst* 2020; 100: 809–826.

23. Davoodabadi Farahani M, Mozayani N. Acquiring reusable skills in intrinsically motivated reinforcement learning. *J Intell Manuf* 2021; 32: 2147–2168.

24. Fan T, Long P, Liu W, et al. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int J Rob Res* 2020; 39: 856–892.

25. Busoniu L, Babuska R, De Schutter B. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans Syst Man, Cybern Part C (Applications Rev* 2008; 38: 156–172.

26. Wang Y, De Silva C. Multi-robot Box-pushing: Single-Agent Q-Learning vs. Team Q-Learning. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3694–3699.

27. Wang Y, De Silva C. Sequential Q-Learning With Kalman Filtering for Multirobot Cooperative Transportation. *IEEE/ASME Trans Mechatronics* 2010; 15: 261–268.

28. Ramachandran D, Gupta R. Smoothed Sarsa: Reinforcement learning for robot delivery tasks. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2125–2132.

29. Husty ML, Pfurner M, Schröcker H-P. A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mech Mach Theory* 2007; 42: 66–81.

30. Barton M, Shragai N, Elber G. Kinematic Simulation of Planar and Spatial Mechanisms Using a Polynomial Constraints Solver. *Comput Aided Des Appl* 2009; 6: 115–123.

31. Zielinska T, Kasprzak W, Szynkiewicz W, et al. Path planning for robotized mobile supports. *Mech Mach Theory* 2014; 78: 51–64.

32. Kasprzak W, Zlatanov D, Szynkiewicz W, et al. Task planning for cooperating self-reconfigurable mobile fixtures. *Int J Adv Manuf Technol* 2013; 69: 2555–2568.