Intelligent Agent Based Operator Support and Beam Orbit Control Scheme for Synchrotron Radiation Sources

R. P. Yadav[1], P. V. Varde[2], P. S. V. Nataraj[3] and P. Fatnani[1]

[1]*ACS, Raja Ramanna Centre for Advanced Technology, Indore 452 013, India*
[2]*RRSD, Bhabha Atomic Research Centre, Trombay, Mumbai 400 085, India*
[3]*IDP in Systems and Control Engineering Group, Indian Institute of Technology Bombay, Powai, Mumbai 400 076, India*
rpyadav@rrcat.gov.in, varde@barc.gov.in, nataraj@sc.iitb.ac.in, fatnani@rrcat.gov.in

Abstract

Synchrotron radiation (SR) sources provide very high photon flux light in a very narrow opening angle with wavelength ranging from visible to hard X-rays for use in experiments related to material science, physics, chemistry and biology. In the beam lines (BL) the SR position is highly dependent on the electron beam position and angle at the source point. The tuning of accelerator for getting the desired electron beam position and angle at the source point is a time consuming and regular job done during commissioning of new BL or when accelerator is operated at new operating point. This paper presents a novel intelligent agent based operator support and beam orbit control scheme for accelerator control. The proposed multi-agent based scheme is well suited for the multilayer control system architectures of synchrotron radiation sources. The scheme successfully distributes the orbit control job to multiple low complexity reactive agents that work simultaneously and control the local orbit for individual BL and insertion devices (ID) in an optimized manner. The proposed scheme of beam orbit control in particular is very useful for machines like INDUS-2, where new BL are in the process of commissioning as this scheme reduces the operator efforts and accelerator tuning time for providing beam to new BL. It also extends the beam availability to other BL (already installed and in use) as the agent tunes the accelerator in systematic way and under constraints on local orbit bump leakage thereby enabling the use of other BL for routine experiments which otherwise was not possible. The effectiveness of the scheme is shown through simulation results obtained by applying the stated scheme on INDUS-2 storage ring model.

Keywords: *intelligent agent, synchrotron radiation sources, accelerator control system, local orbit control*

1. Introduction

Synchrotron radiation sources provide light of very high intensity. Photon flux of order of 10^{13} photons/s/0.1% bandwidth light is a common figure for SR source facilities [1]. This light is of wide bandwidth ranging from visible to hard X-rays. Also the light is emitted in a very narrow angle in the forward direction. Due to these appealing features the SR user community is growing day by day expanding the new areas of SR utilization. In electron storage rings the SR is emitted by the relativistic electrons (electron energy from hundreds of MeV to few GeV is common) when they pass through dipoles or through ID such as wigglers [2] and undulators [2] that are part of electron storage rings. For using the SR in experiments there are BL connected to the storage ring that transport the SR from electron

orbit to the experimental stations. The experiment stations are the part of the BL, designed specifically to perform the particular type of experiment on the sample. In the BL the SR position is highly dependent on the electron beam position and angle at the source point. At the time of commissioning of a new BL, it is first coarsely aligned independently as per the design and then the alignment is fine-tuned by tuning it along with the source point (electron beam position in storage ring). The physical alignment restrictions on the BL side necessitate the residual misalignment to be compensated by correcting the electron beam position and angle at the source point. The electron beam position and angle are corrected by applying the local orbit bump at source point. In normal practice two methods are used for generating the local orbit bump. One is three corrector based closed orbit bump and the other is four corrector based closed orbit bump [4]. Further the four corrector based scheme can be represented by superposition of two distinct three corrector bumps [5]. In the process of manual tuning of accelerator by operators to obtain the desired beam position and angle at source point the operator first calculates the initial corrector values using three/four corrector based scheme and then applies it to the machine. Due to difference in the theoretically calculated values of machine parameters and the actual machine parameters at given operating point, the applied orbit bump does not closes properly and leaks out to the entire electron beam orbit thus producing disturbance throughout the ring. Operator now tries to close the bump to the required level by perturbing the different correctors used for generating the bump in an iterative manner. This task of finding the suitable corrector strength that satisfies the BL alignment and also closes the bump to within limits is a time consuming and routine job done during commissioning of new BL or when accelerator is operated at new operating point.

Intelligent agent is an autonomous software entity situated in an environment. It is capable of observing its environment (partially or fully) through sensors and can proactively or reactively act upon the environment through actuators. It autonomously directs its activities towards achieving its goals. Different agent architectures are proposed for implementing intelligent agents by researchers. The simple reactive architectures like subsumption [5, 6 and 7] architecture are the most popular architectures used in applications demanding the fast reactive behaviors in real time, like robotics. The Logic based [8], model based and goal based [9 and 10] architectures are some other architectures used for applications demanding complex behavior implementations. The multi-agent system is the system that solves the problem through work division among multiple agents of relatively low complexity and limited actuator / sensor coverage over environment. The multi-agent based control approach has been successfully applied for control of large distributed plants such as power plant control [11], secondary voltage control in power systems [12], shipboard automation [13] and many more.

Inspired by the distributed multilayer control architecture of synchrotron radiation sources this paper presents a novel intelligent agent based operator support and beam orbit control framework. The proposed framework distributes the orbit control job among five different types of agents. To take the advantage of layered control system architecture these different types of agents can be implemented at different layers with varying computation power, memory and communication resources. The *beam line control agents* and *insertion device control agents* are designed with the simple subsumption architecture suited for implementation at the lower layer controllers demanding very less computation power. These agents augment the basic behaviors for optimally controlling the local beam parameters in a reactive manner. Being simple and localized in environment these agents exhibit a fast reactive control cycle for *attain position/angle* and *maintain position/angle* type of goals. Since these lower level agents acquire only the local information available to

them to attain their local goals and as the global goal of controlling the orbit is attained by the combined efforts of these low level agents, the performance degradation of any one agent can affect the overall global goal. For this the performance of the *BL control agents* and *ID control agents* are evaluated on periodic basis by *monitoring agent* situated at the higher level of control architecture layer. The *monitoring agents* periodically acquire the data from multiple lower layer agents, evaluate their performance and provide this feedback to them. A low level agent with *average* or *poor* performance takes assistance from *trainer agents* that are again at higher level in hierarchy and implements the complex behavior for gradient based constrained reinforcement learning plans. These *trainer agents* coordinate the student agents training for skill development and actually refine\correct those beliefs whose knowledge content has degraded due to the dynamics of changing environment. The trainer agent forms the team of low level agents comprising of student agent (the agent who is trained to generate the close bump using correctors) and the cooperating agents which provide the information about the actions of student agents on the global beam orbit. The trainer agent in a coordinated fashion suggests the actions to be performed by the student agent to generate the closed bump. It then evaluates the utility of the suggested action by gathering the information from the other cooperating agents taking part in training process and decides the next action to be suggested to the student agent. This iterative process continues till the learning goal is completed. In case the agent finds that the performance is altogether not in accordance with the expected it can seek fault assistance from *fault assistance agents*. The *fault assistance agent* finds the faulty sensor and actuators by logic equation evaluation. At the highest hierarchy there is the model based goal based *orbit control agent* that interacts with the user and database to generate the lower layer agents and control their activities in a synchronized and systematic manner. Also at the same hierarchy there is the *closed orbit minimization agent* which intermittently checks the corrector strength and the beam positions and decides the optimized orbit for *orbit maintain* goal.

The contribution of this paper to the multi-agent research field and to accelerator controls is that this paper extends the use of multi-agent based control to the field of local beam orbit control in synchrotron radiation sources thereby improving the overall reliability and beam availability to the users. It further improves the available beam quality to users through prioritized optimization in case the multiple BL demand the beam correction simultaneously. Further to this the paper presents a novel gradient based cooperative multi-agent learning scheme. The gradient based reinforcement learning schemes have been proposed in past by many researchers [14, 15, 16 and 17]. The scheme presented in this paper differs from all of these as all the previously proposed schemes are focused on identifying the suitable agents actions through their interaction with environment which transforms the system from current state to the desired state but they have not considered the cases where the agents actions performed for learning may lead the environment / system to a state such that the system can no longer be used further. For example if the learning activity by a *BL control agent* disturbs the beam at nearby or other high priority BL such that the experiment data gets affected then the purpose of the orbit control is defeated or if the perturbation introduced by the learning agent disturbs the overall beam to an extent that the complete beam gets killed then there is no beam in orbit to control. Therefore for such cases this paper presents the new scheme of agent's skill learning under constrained condition for improving the system reliability. The presented accelerator control scheme further reduces the operator efforts towards accelerator tuning by automatically performing the beam line tuning based on the learned tuning patterns from the past and present operator feedback about the operation quality.

The agents are being developed according to the presented multi-agent scheme and the results obtained by applying the early implementation of this scheme on INDUS-2

synchrotron radiation source model are presented. The rest of the paper is organized as follows. Section 2 introduces the subsumption agent architecture, Section 3 discusses the close orbit bump concept, Section 4 presents the overall organization of different agents in the multi-agent environment for beam orbit control and Section 5 discusses the design and implementation for different agents. Section 6 discusses the constraint gradient based reinforcement skill learning. In Section 7 the simulation results are presented and discussed followed by conclusion and future work.

2. Subsumption Agent Architecture

The subsumption architecture was first proposed by Brooks [5] for implementing the reactive agents. The architecture decomposes the complicated intelligent behavior into many simple behavior modules, which are in turn organized into layers. Each layer implements a particular goal of the agent, and higher layers are increasingly abstract. Each layer's goal subsumes that of the underlying layers by means of methods provided for inhabiting or invoking the lower layer goals by the upper layer behaviors if required. There are several modifications to this basic primitive architecture proposed by Brooks. The Figure 1 shows the basic linkages between different modules in one such subsumption architecture, here the modules are organized in vertical layers. The modules operate in parallel, with those higher up in the organization having dominance over those lower down [7]. This means that the higher modules can inhibit the behavior of lower level modules. As with the modular architecture, the designer defines the connections between modules and the dominance relationships that exist between them in the form of inhibit rules. Usually the implementations of inhibiting relationship are implemented by means of priority assignment for individual behaviors. Here the agent's decision-making is realized through a set of task accomplishing behaviors. Each behavior is like an individual action function, continually taking perceptual input and mapping it onto an action to perform.

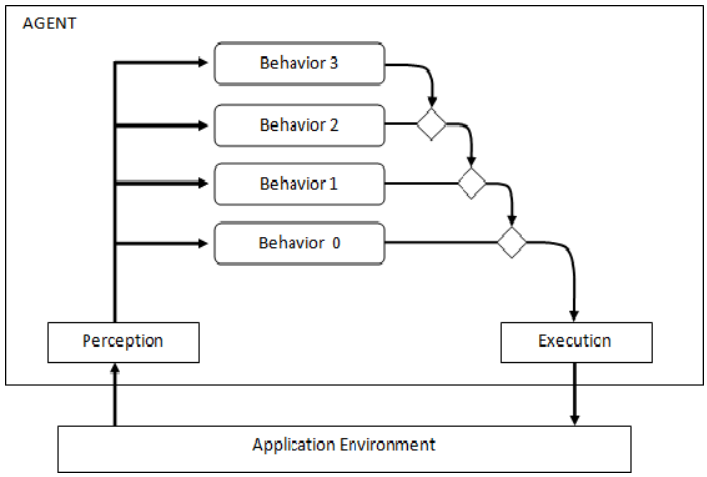


Figure 1. The Subsumption Agent Architecture

No complex symbolic representations and no symbolic reasoning is utilized at all. It mainly implements the rules to map *situation* *action* relationship. The percepts block accepts input from sensors and produces a set of percepts *P*. The action is realized through a set of behavior rules *R*, together with an inhibition relation (-<), over time.

$$((c, a) \mid C \text{ g } P, a \text{ E } A) \quad (1)$$

Where *c* is a set of percepts called condition and *a* is an action. A behavior will fire in state *s* if some function *see(s)* E c (if the condition is satisfied by the percepts). The inhibition relation is a total ordering on the behavior rules. If *r1* inhibits *r2*, then the inhibit rule can be written as *r1 <r2*, i.e., *r1* is lower in the hierarchy than *r2* and hence will get priority over *r2*. Where *r1, r2, ...* are the rules (elements of *R*).The subsumption reactive architecture is well suited for implementing the lower level control agents like *BL* and *ID control agents* in our work as this provides a fast reactive control cycle as well as can run on low computation power controller in the field where the required sensors and actuators are directly connected to the controller. As the accelerator control systems are generally comprised of multi layered architecture [18, 19 and 20] with low computation power controllers at the lower most layer and increasingly high computational power controllers at higher layers. The choice of reactive agent architecture for *BL control agents* and *ID control agents* also favors the scheme from actual deployment and debugging as one can implement and test the behaviors one by one on controller basis in an incremental way till all the behaviors are implemented at all the controllers. The section 5 discusses the necessary goals implemented and their priority for exercising the *BL control agents* and the *ID control agents*.

3. Closed Orbit Bump and Accelerator Environment

In synchrotron radiation sources the highly energetic electrons are made to follow a nearly circular path through magnetic optics comprising of dipoles for bending of electron beam and quadrupoles for focusing/ defocusing of beam [21]. The electron beam position along the beam path in transverse plane (*i.e.,* the plane perpendicular to the beam motion) is controlled using corrector magnets. Depending upon the direction in which the corrector magnet gives kick to the beam they are categorized into vertical corrector magnets and horizontal corrector magnets. Electron beam position Indicators (BPI) are used for measuring the beam position at different locations in the ring. For successful storage of the electrons in the ring, only some specific set of dipole and quadrupole magnet settings are allowed which are governed by the accelerator beam dynamics [21] and specifies the accelerator operating point. In normal practice due to presence of different types of errors (alignment error, field error, manufacturing tolerances *etc.*) in components the electron beam deviates from its ideal design trajectory (normally called as golden orbit). The corrector magnets are used to control the orbit in order to bring it near to the golden orbit (ideally if possible it should match with the golden orbit). This is known as closed orbit distortion (COD) correction. After the beam is corrected for COD if one wants to purposefully change the orbit at desired location then the kicks to the beam are required to be applied using group of correctors. Different schemes for this are adopted in normal practice like two corrector closed bump, three corrector closed bump and four corrector closed bump [22]. Choice of using the particular scheme is normally dependent on the machine operating parameters (ψ and β) and machine design (location of correctors and BPI's). For simplicity we will restrict to three corrector bumps and four corrector bumps only

3.1. Three Corrector Bump

In this scheme three correctors are used to generate the local bump where the first corrector opens the orbit, *i.e.*, deflects the beam away from the orbit the second corrector deflects the beam towards the orbit and the third corrector closes the bump by correcting the angle as shown in Figure 2. In figure 2 let 0_1, 0_2, 0_3 be the kick angles generated by three corrector magnets CV_1, CV_2, CV_3, then for closed bump condition

the θ_1, θ_2, θ_3 must satisfy the Eq. 2 [3] where β_i is the β-function at the i^{th} corrector magnet, $\psi_{ij} = \psi_i - \psi_j$ is the phase advance from the i^{th} to j^{th} corrector magnet.

$$\frac{\theta_1}{\sin\psi_{2,3}} - \frac{\sqrt{\ }}{\sin\psi_{1,3}} - \frac{\sqrt{\ }}{\sin\psi_{1,2}} \qquad (2)$$

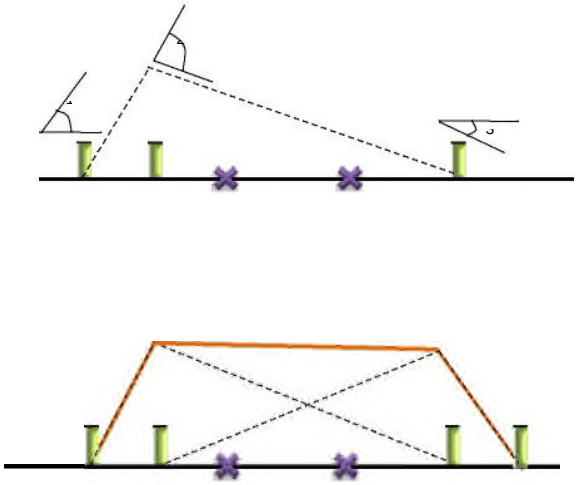


Figure 2. Three Kicker Bump Scheme

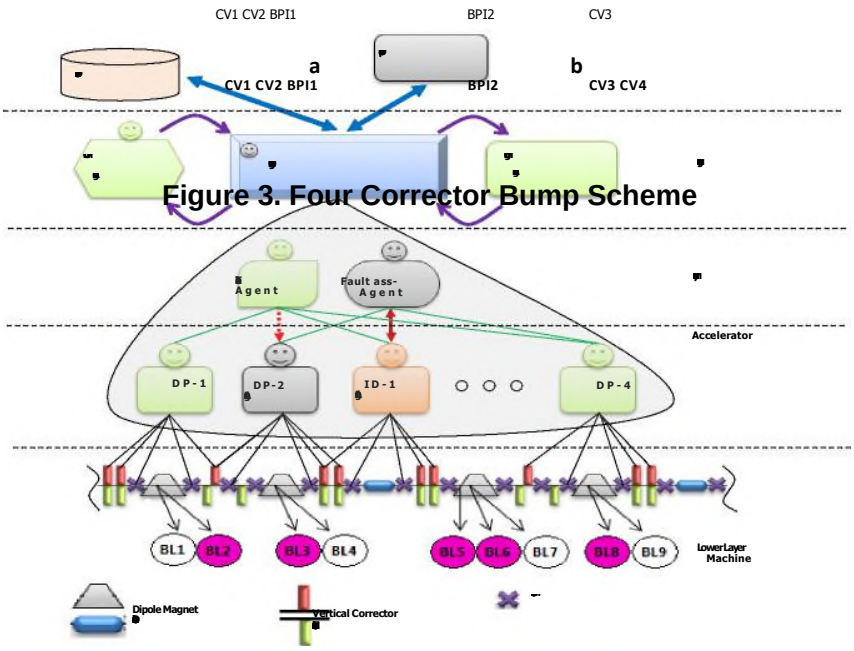


Figure 3. Four Corrector Bump Scheme

Figure 4. Organization of agents for multi-agent based beam orbit control system

3.2. Four Corrector Bump

The four corrector bump can be considered as the combination of two independent three corrector bumps *a* and *b* as shown in Figure 3. Let 0_a, 0_b be the strengths of three corrector bump *a* and bump *b* individually, then the corrector strengths 0_1, 0_2, 0_3, 0_4 will be given by Eq. 3 [4]. For the bump to be closed the matrix coefficients K_{1a}, K_{2a}, K_{3a} and K_{1b}, K_{2b}, K_{3b}, must satisfy the Eq. 2. The beam positions $[x_1, x_2]$ are then related to the bump strengths $[0_a, 0_b]$ through a 2×2 local response matrix R_l given by Eq. 4.The elements of response matrix R_l can be measured experimentally or calculated from the lattice function fl and yr at the BPI and corrector locations using Eq. 5 [4], where ν is the tune and k_i and k_{cj} are the coefficient of sensitivity for BPMs and correctors, respectively.

[ ] [ ] [ ] [ ](3)[ ] [ ] (4)

0 K 3 b √ (| |)

4. Organization of Agents in Multi-agent Beam Orbit Control Environment

The work in this paper implements electron orbit control system for synchrotron radiation sources by means of cooperative working of multiple agents simultaneously to achieve the overall global goal of orbit control. The different types of agents are arranged in the form of layers as shown in the Figure 4. The lower layer agents are the one that are directly interfaced with the accelerator machine components. There are two types of lower layer agents namely *BL control agents* (depicted by DP-1 agents, DP-2 agent and DP-4 agent in Figure 4) and *ID control agents* (depicted by ID-1 agent in Figure 4). The lower layer agents encapsulate the behaviors necessary for controlling the beam position and angle for BL generated from dipoles and that from ID. The behaviors are implemented through subsumption architecture to exhibit fast reactive cycle where in each execution cycle the agent reads the BPI's connected to it and according to the current state and desired state it calculates the corrector strengths using their skill and apply it to the correctors. Additional behavior are incorporated at different priority levels to work in synchronization, seek help from *trainer agents*, assist *trainer agents*, seek help from *fault assistance agents* and to assist *fault assistance agents*. The lower layer agents are simpler in nature and can sense only the environment local to them i.e. they can only sense the beam position at the BPI's which are interfaced with the particular agent and similarly they can only take action on the environment through the actuators interfaced to them. Further the lower layer agents do not possess the capability to communicate and self collaborate with other similar lower layer agents. Thus the lower layer agents can only communicate with the middle layer or higher layer agents and that too with a slower rate than the rate of their execution cycle. In fact the lower layer agent exhibits a high degree of autonomy, performing actions in its environment based on information (sensors, feedback) received from the environment. The *trainer agents* and the *fault assistance agents* are situated at middle layer and are capable of gathering information from lower layer agents thereby observing the relatively larger environment (but at a lower rate than what the lower layer agents can do) as compared to the individual lower layer agents.

These agents implement the higher level of abstraction in their behavior and are formulated to implement the system requirements that needs data collected from multiple lower layer agents such as training of lower layer agents for skill learning towards close bump generation and faulty device (sensor and actuator) identification. Middle layer agents are created on demand by orbit control agent whenever there is a request from lower layer agents for skill development and fault finding assistance. These agents, on becoming active, take control of overall beam orbit control activity and issue command to all lower layer agents to stop their ongoing activity and prepare them to participate in activities for higher level goal achievement. It then issues command to different participating agents according to their role in the team to interact with the environment in a coordinated manner and provide the necessary information required by middle layer agents in their decision making. These agents are implemented with modified subsumption architecture where the behavior selection is done in a manner similar to the subsumption architecture explained in Section 2 but here the behavior functions are of much complex nature and implementation of some behaviors also utilizes the data storage between iterations. At the highest level there are three agents namely *monitoring agent*, *closed orbit distortion minimization agent* and *orbit control agent*. All of these agents are the main agents in this proposed multi-agent based electron orbit control system and always remain active. These are at the highest level of abstraction and control the overall orbit control job. The *monitoring agent* periodically (or when demanded) collects sensory data from lower layer agents and evaluates the individual agents performance. In case of performance degradation it informs the corresponding lower layer agent about its performance (*good*, *average*, *poor* and *bad*). The *closed orbit distortion minimization agent* contains the rough system model (previously measured system's response matrix) and on request it provides the optimized corrector strengths using the singular value decomposition (SVD) [22] method for achieving the desired golden orbit. This information is used for initialization at the time of starting of the overall orbit control system or intermittently if the overall organization of the agents is reinitiated. It also provides the tentative beam orbit for *orbit maintain* goal of different lower layer agents. The *orbit control agent* is responsible for creation of different lower layer and middle layer agents when required. It is implemented with a model based goal based modular architecture. It implements the graphical user interface for interaction with the operator. It reactively interacts with the operator to modify the beliefs in case of doubt, for example if *fault assistance agent* has executed all the related plans but still the faulty device is not found, or a timeout has occurred before a faulty device could be identified. It proactively interacts with the operator to seek suggestions about the present system performance to update the beliefs about the good beam quality at respective BL. It further accepts operator commands in the form of *attain position* and *attain angle* goals for the respective BL and send these goals to the respective *BL control agents* / *ID control agents*. It also provides the updated beliefs to lower layer agents whenever the beliefs are updated or agents are created. For example whenever a belief about the BL operation ([*good: F, 0*], [*average: F, 0*], [*poor: F, 0*]) changes, the same is provided to the lower layer *BL control agent* so that it can decide the optimum position and angle to be attained and maintained when the BL is put to use by opening the safety shutter of that BL. The graphical user interface served by this agent presents the pictorial representation of all the agents present in the multi-agent organization at any instant of time. The different execution states of agents are represented with the different colors in the mimic. The association of accelerator devices with the agent is

shown with the lines connecting devices to the corresponding agents and the interconnection between different agents are represented with the colored linkages between them where the line color shows the direction of message passing between collaborating agents. This live mimic of the overall system is very useful for debugging purpose.

5. Agent Design and Implementation

This section present the abstract implementation for each distinct type of agent that is needed for implementing the proposed multi-agent based orbit control system. Table 1 gives the percepts, beliefs, constraints and behaviors for implementation of *BL control agent*. Table 2 gives the percepts, beliefs, constraints and behavior for implementation of *ID control agent*, where "percepts" are the signals that are interfaced with the agent. In each execution cycle the agent reads data from beam position indicators (*BPI1* and *BPI2*), correctors (*CV1, CV2 and CV3*) and safety shutter (*SS1 and SS2*). It then checks for the firing condition of each behavior and lists the behaviors triggered as per the current percepts. It then selects the highest priority behavior according to the inhibition relation and executes the associated plan. The "beliefs" store the system information needed by the actions in the plans for calculation of parameters such as the optimized angle deviation (δO) to be applied to the electron orbit and the calculated optimized corrector ([$\Delta cv1$, $\Delta cv2$, $\Delta cv3$]) values (using Eq. 3 and 4) to be applied to the correctors to achieve the desired orbit deviation. The "constraints" mainly store the parameter limits needed for assisting in action selection or needed for some parameter calculations like the minimum and maximum limiting values for correctors under different operating modes (*free running mode*, *restricted mode*). The *Skill* stores the information about the effect on local environment that agent's action produces. This information is used by the agent in calculating the suitable corrector current values needed for correcting the local beam orbit. The quality of agent *skill* is calculated using the Eq. 8 where $_{HBump}$ $^{\text{and}}$ $_{PerValue}$ given by Eq. 6 and Eq. 7 represent the generated hump height in the local region and the performance value calculated from percepts of the assisting team member agents. *Skill* level is an important parameter for correct operation of this scheme. It represents the agent's ability towards correcting the electron orbit in region local to it while at the same time isolating the effect of this applied local correction from rest of the orbit. The degradation of the *skill* level causes the leakage of corrective action applied to the local orbit (orbit region which is in the direct influence of an agent) to the entire orbit. This will cause the interference between the simultaneously occurring actions of different agent on the electron beam orbit.

(6)
(7)

(⁄) (⁄)

The behaviors *b0* to *b7* broadly provides the agent ability to correct the beam angle ($\pm\Delta O$) at the beam line in *beam line control agent* and beam angle and position ($\pm\Delta O$, $\pm\Delta P$) in the *insertion device control agent* case, maintain the beam position and angle at BL and ID, help other agents in training and fault finding activities and handling the

cases when agent's performance degrades due to degraded agent skill level under system dynamics.

Table 3 gives the design of trainer agent (for *BL control agent* type student agent). At the time of trainer agent creation, the *orbit control agent* associates the variables in percepts, beliefs, skill and constraints according to the student agent designated to it for imparting the training. At the time of starting, the trainer agent informs all the lower level agents about its intention to conduct the training for its student agent. It then distributes the role to each participating agents that they have to play as team members in training. It then waits till all the participating agents come to a predefined state before proceeding for the start of training activity. The behavior *b0* to *b4* implement the constraint gradient based reinforcement skill learning activity discussed in detail in section 6 by systematically coordinating the activities of distributed lower layer agents through sequential instruction issuing to individual agents such that the actions of all the participating agents become synchronized towards training. Under such team formed condition the complete team can be viewed as a single unit performing the learning activity. After the completion of learning (when either the skill to sufficient level is learned or the activity is stopped due to high priority goals) the trainer agent updates the learned *skill* along and *performance* value to student agent and releases all the participating agents to work independently.

Table 4 gives the design of fault assistant agent. Similar to the trainer agent case this type of agents are also created by *orbit control agent* and variables in percepts, beliefs, skill and constraints are initialized according to the suspected faulty agent designated to it for fault finding. Similar to the trainer agent case at start time fault assistance agent informs all the participating agents about the intended fault finding activity and prepares them for assistance by designating their role in the activity. Based on the historical data, system believes and systems state it then prepares the suspected device list. Using this suspected device list it than prepares the fault finding plan list for all the suspected devices. Behaviors *b0* and *b1* deal with the cases of stopping of fault assistance activity either due to the stopping request from a high priority goal or if the faulty device has been found or if the entire selected fault finding plans have been tried and agent could not associate the fault with any of the suspected devices. Behavior *b2* performs the function of coordinating the activities of participating lower level agents so that the fault finding team of agents can be thought of as a single unit. This behavior selects one fault finding plan from the list. It then directs the plan *actions* to actuating agent to perform the perturbation to environment through actuator and collects the changed environment information supplied from assisting team member agents. It then generates the Boolean variables from the measured parameters. The Boolean variables are then put in the predefined logical equations to conclude that the device is faulty or ok.

Table 5 gives the design for *monitoring agent*. The monitoring agent through its behavior *b0* performs the job of evaluating the performance of the lower level agents. It does so by coordinating the team of agents where one agent performs the demo action on environment and all other distributed team members supply the information about the effect of this demo action on the environment. This is used to evaluate the performance of the agent. The performance of the agent is evaluated in four fuzzy levels; *good, average, poor* and *bad*. The behaviors *b1* and *b2* provide the mechanism of engaging into performance monitoring activity through reactive mechanism (*i.e.*, when *orbit control agent* requests for performance evaluation of some lower level

agent) or through proactive mechanism (i.e. the policy based performance evaluation, both routinely evaluation as well as history based evaluation can be used).

Table 6 gives the *COD agent* design. This agent uses the system model in its beliefs and exhibits the complex capabilities *C0* and *C1* that provide this agent the functionalities to generate the optimized golden orbit and the optimized corrector settings using the global orbit correction schemes like SVD, sliding bump or both depending upon the policy selected. These values are used by the *orbit control agent* to initialize the lower level agents at the time of their creation.

Table 7 gives the design for *orbit control agent*. This agent is a higher level agent and mainly controls the overall local beam orbit control job. It provides the man-machine interface for operators to interact with machine in normal operation when operator directly interacts with the system to start/stop the system, increase/decrease beam angle and position at particular beam lines, declares the particular devices such as BPI's or correctors as faulty devices (*i.e.*, the devices that are not to be used in normal operation), declares the sensitivity of experiment being performed at the beam lines ([angle:: *high*, *medium*, *low*], [position:: *high*, *medium*, *low*]), declares the upcoming beam lines, declares the present orbit to be used as the golden orbit for the system. This agent contains the complete system model in its beliefs. The knowledge base is divided into two parts- machine specific knowledge base and accelerator physic specific knowledge base. The machine specific knowledge base is further divided into static knowledge base and dynamic knowledge base. The static knowledge base contains the machine specific information which does not change according to the system's state of operation like the sequence of components, location of components, coefficients of different components etc. The dynamic data base contains the information that is dependent on the system's state of operation like power supply on/off status, BPI enabled/disabled state etc. The capabilities *C0* to *C5* implement the functionalities for generating the *beam line control agent*, *insertion device control agent*, *trainer agent*, *fault assistance agent* and handling of proactive events, user generated events and machine generated events. Figure 5 shows the detailed architecture for this agent.

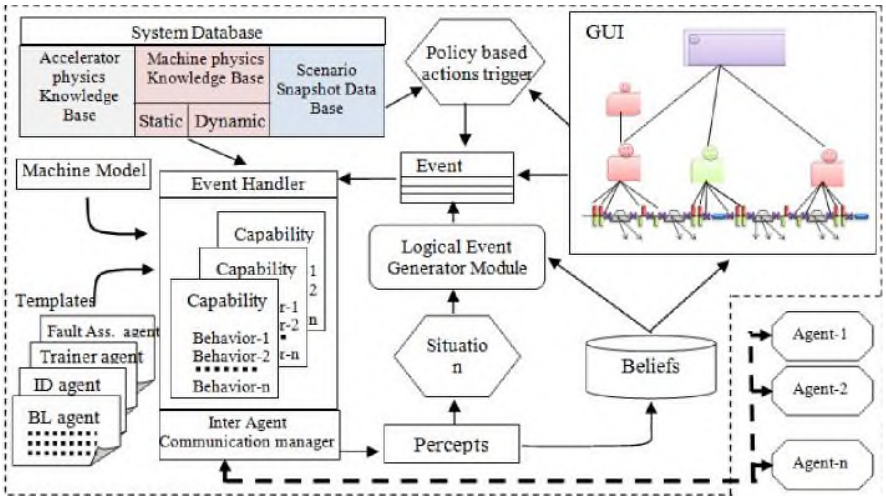


Figure 5. Architecture of Orbit Control Agent

Table 1. Beam Line Control Agent Design

Percepts :: *BPI1, BPI2 , CV1, CV2,CV3, SS1, SS2*

Beliefs :: *performance, self state, related bl, {bl1 ⇔[good: P, θ], [average: P, θ], [poor: P, θ]}, {bl2 ⇔[good: P, θ], [average: P, θ], [poor: P, θ]}, {cv1 ⇔[good: a, Δa], [average: a, Δa] }, {cv2 ⇔[good: a, Δa], [average: a, Δa] }, {cv3 ⇔[good: a, Δa], [average: a, Δa] }*

Skill :: *cv1:cv2:cv3:θ_1: θ_2*

Constraints :: *{cv1 ⇔[free running: max ,min], [restricted mode: max ,min]}, {cv2 ⇔[free running: max ,min], [restricted mode: max ,min]} , {cv3 ⇔[free running: max ,min], [restricted mode: max ,min]}*

Behavior :: *b0: work in synchronous way*
(*Event : execute*)
Execute single cycle
Else
Remain ideal

Behavior :: *b1: remain ideal*
Do nothing skip execution cycles

Behavior :: *b2: assist collaborators*
(*event :supply information*)
Perform the percept gathering
Perform the percept processing
Perform the supply information
(*event: action request*)
Perform the percept gathering
Perform the percept processing
Check feasibility (requested action)
Feasible → perform action.
¬ Feasible → message: action refused.

Behavior :: *b3: serve average performance*
Calculate the error in θ
Generate attain goal.

Behavior :: *b4: serve poor performance*
(*bl is high priority ∧ in use*)→ *wait(b1:remain ideal*).
((¬ *bl is high priority*) ∧ *in use*) → *request training assistance* (*self state ← training*).

Behavior :: *b5: serve bad performance*
Revert the last action on environment
Request fault assistance (*self state ← fault assistance*).

Behavior :: *b6: serve attain goal* (*event: suggested±Δθ to bl*)
Optimized $\delta\theta \leftarrow f_{optimize\text{-}\theta}$ (*Percepts , beliefs , ±Δθ*)
Optimized [Δ*cv1*, Δ*cv2*, Δ*cv3*]$\leftarrow f_{optimize\text{-}cv}$ (*Percepts , beliefs ,skill, constraints,* $\delta\theta$)
Apply [Δ*cv1*, Δ*cv2*, Δ*cv3*] *to system.*

Behavior :: *b7: serve maintain goal*
Calculate the error in θ
Serve attain goal.

Inhibition relation : : *b0,b2 ≺b5≺b4≺b6≺b3≺b7*

Table 2. Insertion Device Control Agent Design

Percepts :: *BPI1, BPI2 , CV1, CV2,CV3,CV4, SS1*
Beliefs :: *performance, self state, {b11 ⇔[good: P, θ], [average: P, θ], [poor: P, θ]}, {cv1 ⇔[good: a, Δa], [average: a, Δa] }, {cv2 ⇔[good: a, Δa], [average: a, Δa] }, {cv3 ⇔[good: a, Δa], [average: a, Δa] }, {cv4 ⇔[good: a, Δa], [average: a, Δa] }*

Skill :: *cv1:cv2:cv3:θ1: θ2: cv2:cv3:cv4:θ1: θ1*
Constraints :: *{cv1 ⇔[free running: max ,min], [restricted mode: max ,min]}, {cv2 ⇔[free running: max ,min], [restricted mode: max ,min]} , {cv3 ⇔[free running: max ,min], [restricted mode: max ,min]} , {cv4 ⇔[free running: max ,min], [restricted mode: max ,min]}*

Behavior :: *b0: work in synchronous way*
(*Event : execute*)
Execute single cycle
Else
Remain ideal

Behavior :: *b1: remain ideal*
Do nothing skip execution cycles

Behavior :: *b2: assist collaborators*
(*event :supply information*)
Perform the percept gathering
Perform the percept processing
Perform the supply information
(*event: action request*)
Perform the percept gathering
Perform the percept processing
Check feasibility (*requested action*)
Feasible → perform action.
¬ Feasible → message: action refused.

Behavior :: *b3: serve average performance*
Calculate the error in P and θ
Generate attain goal.

Behavior :: *b4: serve poor performance*
(*b1 is high priority ∧ in use*)→ *wait*(*b1:remain ideal*).
((¬ *b1 is high priority*) ∧ *in use*) → *request training assistance* (*self state ← training*).

Behavior :: *b5: serve bad performance*
Revert the last action on environment
Request fault assistance (*self state ← fault assistance*).

Behavior :: *b6: serve attain goal* (*event: suggested ±ΔP, ±Δθ to b1*)
Optimized [*Δcv1, Δcv2, Δcv3, Δcv4*]← $f_{optimize\text{-}cv}$ (*Percepts , beliefs ,skill, constraints, ΔP, Δθ*)
Apply [*Δcv1, Δcv2, Δcv3, Δcv4*]*to system.*

Behavior :: *b7: serve maintain goal*
Calculate the error in P and θ
Serve attain goal.

Inhibition relation : : *b0,b2 ≺b5≺b4≺b6≺b3≺b7*

Table 3. Trainer agent for (beam line control type student agent) design

Percepts :: *sensors* ←*assisting team members* , *actuators*[*CV1, CV2,CV3*]←*student agent* .
Beliefs :: *performance* ←*student agent, self state, safe beam movement area, limit bump height*(L_{BH}), *bump step size*(B_{SS}), *Corse step size* (C_{SS}), *fine step size* (f_{SS}), *bpi noise band*(Δ_{noise}),$[BPM]_{Bump\ sense}$, $[BPM]_{quality\ sense}$

Skill :: [*cv1:cv2:cv3:*θ_1: θ_2] ←*student agent*
Constraints :: [*bpi*] ←*golden beam orbit* ,{*beam orbit : relaxed band*(Δ_R), *start band*(Δ_{start}), *stop band*(Δ_{stop}), *constraint band*(Δ_{const})} , *Maximum iterations* (N_{max}), *Useful performance limit*(L_{UsePer}), *Useful bump height limit* ($L_{Use\ BH}$) , *Limit max corrector current change* (ΔI_{CV1}) .

Initialization::
Inform all agents about the training activity.
Formulate coalition for the participating agents and distribute their role.
Wait for different agents to finish their work and come to desired state.
Training starting point ← $f_{starting\ point}$ (*Data Base, Skill, system state*).

Behavior :: *b0: Serve stop training request*
(*Event : stop training*)
Per_{value} ← $f_{cal\ per\ value}$ (*percepts, beliefs*)
If ((Per_{value} < L_{UsePer}) ∧ ($[BPM]_{Bump\ sense\ |max}$> $L_{Use\ BH}$))
Release agents coordinating in training
Release student agent (performance ← *average)*
Else
Release agents coordinating in training
Release student agent (performance ← *poor)*

Behavior :: *b1: Adjust bump height*
Increment CV_1 *by Css*

Behavior :: *b2: Adjust step size*
(*increment*)
Increment C_{SS} *by fss*
(*decrement*)
Decrement C_{SS} *by fss*

Behavior :: *b3: Goal achieved*
Release agents coordinating in training
Release student agent (performance ← *good)*
Record the activity in database for future use

Behavior :: *b4: Execute single training cycle*
[*Suggested moves*] ← $f_{suggest\ moves}$ (*percepts, beliefs, skill, constraint*)
For each suggested move
Coordinate activity
Student agent ←*event: action request* (*suggested move*)
Assisting team members ← *event: supply information*
Per_{value} ← $f_{cal\ per\ value}$ (*percepts, beliefs*)
Stop Coordinate activity
Skill ← $f_{new\ skill\ value}$ ([*Suggested moves*], [Per_{value}], *skill*)
Student agent ← (*update Skill*)

Inhibition relation : : *b0* ≺ *b1*≺*b2*≺*b4*≺*b3*

Table 4. Fault assistance agent design

Percepts :: *sensors* ←*assisting team members*, *actuators* ← *faulty agent* .

Beliefs :: *performance* ←*faulty agent, self state, safe beam movement area, Corse step size* (C_{SS}),*fine step size* (f_{SS}), *bpi noise band* (Δ_{noise}),$[BPM]_{Bump\ sense}$, $[BPM]_{quality\ sense}$,

Skill :: $[cv1:cv2:cv3:\theta_1:\ \theta_2]$ ←*faulty agent*

Constraints :: [*bpi*] ←*golden beam orbit* ,{*beam orbit : relaxed band* (Δ_R), *start band* (Δ_{start}), *stop band* (Δ_{stop}), *constraint band* (Δ_{const})} , *Maximum iterations* (N_{max}), *Limit max corrector current change* (ΔI_{CV1}) .

Initialization::

Inform all agents about the fault assistance activity.
Formulate coalition for the participating agents and distribute their role.
Wait for different agents to finish their work and come to desired state.
[*Suspected device list*] ← $f_{order\ suspected\ device\ list}$ (*Data Base, Skill, system state*).
[*fault finding plan list*] ← f_{plans} (*percepts, beliefs, skill, constraint,*[*Suspected device list*])

Behavior :: *b0: Serve stop fault assistance request*
(*Event : stop fault assistance*)
Release agents coordinating in fault assistance
Release faulty agent (*performance* ← *fault cannot be identified*)

Behavior :: *b1: Goal achieved*
Release agents coordinating in fault assistance
Release faulty agent (*beliefs* ← *faulty*(*device ID*))
Record the activity in database for future use

Behavior :: *b2: Execute single fault assistance cycle*
Coordinate activity
plan← *next plan* (*fault finding plan list*)
faulty agent ←*event: action request* (*plan* (*action*))
Assisting team members ← *event: supply information*
VAR_{Bool} ← $f_{coil\ per\ value}$ (*percepts, beliefs*)
Stop Coordinate activity
If (*Fault equation* (VAR_{Bool}))
self state ←*fault found* (*Goal achieved*)
else
if(*All plans in the list executed*)
Serve stop fault assistance request

Inhibition relation : : $b0 \prec b1 \prec b2$

Table 5. Monitoring Agent Design

Percepts :: *sensors* ← *lower level agents* , *monitored agent* ← *current agent*
Beliefs :: *performance* ← *current agent, self state,*

Constraints :: [*bpi*] ← *golden beam orbit* , {*beam orbit : good performance band* (Δ_{good}), *average performance band* ($\Delta_{average}$), *poor performance band* (Δ_{poor}), *bad performance band* (Δ_{bad}), *constraint band* (Δ_{const})}.

Behavior :: *b0: Serve agent performance evaluation*
Inform all agents about the agent performance evaluation activity.
Formulate coalition for the participating agents and distribute their role.
Wait for different agents to finish their work and come to desired state.
Coordinate activity
Monitored agent ← *event: action request (suggested move)*
Lowest level agents ← *event: supply information*
$Per_{value} \leftarrow f_{cal\ per\ value}$ (*percepts, beliefs*)
Stop Coordinate activity
Performance ← $f_{fuzzify\ performance}$ (Per_{value}, *constraints*)
Update (*Current agent* ← *Performance*)

Behavior :: *b1: Serve reactive agent performance evaluation*
(*Event: evaluate performance*(*agent*))
Current agent ← *agent*
Serve agent performance evaluation

Behavior :: *b2: Serve proactive agent performance evaluation*
Current agent ← $f_{suspected\ agent}$ (*data base, percepts, policy*)

Inhibition relation : : *b0* ≺ *b1*≺*b2*

Table 6. COD agent design

Percepts :: *sensors* ← *lower level agents*
Events :: *get new orbit, get optimized corrector settings*

Beliefs :: {*system model: horizontal response matrix* (RM_x), *vertical response matrix* (RM_y)},
{*system data base : machine data base* (*static, dynamic*)}

Constraints :: [*bpi*] ← *golden beam orbit* , {*beam orbit : good performance band* (Δ_{good}), *average performance band* ($\Delta_{average}$), *poor performance band* (Δ_{poor}), *bad performance band* (Δ_{bad}), *constraint band* (Δ_{const})} , {*cvi* ⇔ [*free running: max ,min*], [*restricted mode: max ,min*]}

Capability: *C0: get new golden orbit*
Sector orbit into restricted and relaxed regions
[*restricted , relaxed*] ← $f_{sector\ orbit}$(*golden orbit, beliefs, percepts,*)
New golden beam orbit ← $f_{optimized\ orbit}$ ([*restricted , relaxed*], *constraints*)
Dispatch ← *new golden orbit*

Capability : *C1: get optimized corrector settings*
Sector orbit into restricted and relaxed regions
[*restricted , relaxed*] ← $f_{sector\ orbit}$(*golden orbit, beliefs, percepts,*)
Optimized corrector settings ← $f_{optimized\ cor\ settings}$([*restricted , relaxed*], *constraints, policy*)
Dispatch ← *optimized corrector settings*

Table 7. Orbit Control Agent Design

Percepts :: *sensors ← lower level agents*
Events :: ***user generated events***
e1: Start operation
e2: Stop operation
e3: Increase/decrease θ at BLi or IDi
e3: Increase/decrease P atIDi
e4: Do Not use BPIi
e5: Do Not use CVi
e6: Exp. at BLi is sensitive
e7: BLi is upcoming BL
e8: Force present orbit as the golden orbit
System generated events
e9: Beliefs updated
e10: Agent died
e11: Agent not responding
e12: Fault cannot be identified
e13: Seek operator suggestion request
e14: Agent status updated
e15: Update system mimic
Suggestions seek from **Operat**or ::
evaluate overall performance of the machine operation (good\average\poor)
evaluate present orbit (good\average\poor)
evaluate position at BLi (large hi\ slight hi\ good\ slight low \large low)
evaluate BPIi performance (good \ average\ poor\ faulty)
evaluate CVi performance (good \ average\ poor\ faulty)
clarify doubt is the BPIi faulty (ok\ faulty)
clarify doubt is the CVi faulty (ok\ faulty)
Beliefs :: {*system data base* (DB) : *machine data base* (*static, dynamic*), *accelerator physics database , scenario snapshot database* }
Capability: *C0: generate beam line control agent*
Capability: *C1: generate insertion device control agent*
Capability: *C2: generate trainer agent*
Capability: *C3: generate fault assistance agent*
Capability: *C4: generate proactive events*
Capability: *C5: handle events (ei)*

6. Constraint Gradient based Reinforcement Learning

The reinforcement learning (RL) is the common learning scheme used by the agent based systems. In this scheme an agent selects and engages in behavior with respect to sensory inputs obtained from sensors. As a result the learning is performed by repeating a cycle in which a reward from the environment and sensory input for the next state is given. The Q-learning and Temporal-Difference (TD) Learning are two common RL methods; the former learns the utility of performing actions in states, while the latter usually learns the utility of being in the states themselves. But apart from learning the system states and the action relationships in some situations like the local orbit control in accelerators discussed in this paper, there is also a need to identify the local environment's behavioral representation that is independent of the action selection by the agent. This can be thought of as the local environment representation (local

environment model) in agent beliefs that does not affect the agent's action selection decision but directly affects the outcome of the agent's actions on the environment. For example if one of the "beam line control agent" decides to correct the beam angle at the respective beam line and engages in behavior "serve attain goal (event: suggested ±ΔO to BL)" to attain the suggested ±ΔO value. Apart from action sequences and all other relevant things the outcome of this behavior also depends on the correctness of the calculated corrector strengths and this calculated corrector strength depends on how well the local environment is modeled by the agents skill. For beam line control agents, the skill is represented by the set [*Skill::CV1:CV2:CV3:BPM1:BPM2*] where the *CV1,CV2,CV3* are the corrector magnet set values and *BPM1*, *BPM2* are the beam position monitor values for the stated corrector values. Now for the case of agent with a good skill level the skill variables (*CV1, CV2* and *CV3*) will be having a definite ratio that satisfies the Eq. 2 to generate the non-leaky closed bump. Thus for changed accelerator environment this skill will not be able to generate the close bump and the amount of bump leakage will directly depend on the amount by which the system dynamics have changed the environment. Thus in such cases the relearning of the skill is required by the agent. As in this case the agent learning is not concerned with the action selection but is linked to the local system identification, we are referring this as the agent's skill learning and as in the highly distributed environment the agent itself does not contain the sufficient resources to carry out this learning on its own. The coordinated training of the student agent is carried out by the trainer agent who suggests the moves (actions) to the student agent, evaluates the outcome of this move and depending upon this outcome decides the next move and new skill level. The next move is decided, based on the gradient of the cost value (reward value). For simplicity we will view the distributed multiple agents performing the coordinated learning activity as a single unit.

Some systems like the accelerator orbit control in SR sources, demand that during the training the agent's action performed towards learning activity should not perturb the electron beam at nearby or other high priority beam lines to a level such that the experiment data gets affected or the complete beam gets killed. To overcome this problem the constraint based skill learning algorithm is proposed here. Complete skill learning by the agent in the presence of noise requires that the system should be perturbed such that at least a minimum level of bump height at the desired position is generated while keeping the disturbance in the beam at all other locations confined within the limits. Contrary to this in actual system such desired bump can only be generated while fulfilling the constraint requirements only if the correct model of nearby environment is available (*i.e.*, the fully trained skill set is available). So in order to solve this paradox the proposed algorithm in the beginning (when skill set is initialized to all zero or some pre-known skill set) starts the system identification with very small perturbation done to the environment through first corrector (*CV1*) such that the beam positions at constraint locations $_{(Pervalue)}$ remains within the constraints limit (Δ_{const}) *i.e.*, the Eq. 9 is satisfied. It then keeps on increasing the *CV1* values in steps till the Per_{value} just crosses the start limit band $_{(\Delta start)}$ (*i.e.*, the Eq. 10 is satisfied). It then tries various combinations for *CV2* and *CV3* and selects the one based on the steepest gradient method that reduces the Per_{value} the most.

(9) P_{vc} (10)

$_{u} >^{1} stc\ t$

(11) $Per_{vc\ ue}$ (12)

$G1_{s\ top}$

It then updates the skill level according to the best action performed in the iteration cycle. This process of refining the skill continues till the Per_{value} reduces below start limit $(\Delta start)$. It then uses this partially learned skill and calculates new values for *CV1*, *CV2*, *CV3* such that it satisfies the Eq. 10. It then again starts the skill refining process using gradient based method. This multi-step alternate bump height increasing and partial skill learning process continues till the desired bump height $(HBump)$ is achieved. Finally after the desired bump height is reached the skill refining process ends when the disturbance in beam orbit in the constraint region $(Pervalue)$ reduces below stop limit $(\Delta stop)$. This overall process for the proposed algorithm is shown by flowchart in Figure 6.

7. Early Implementation Simulation Results on INDUS-2 Model

The agents are developed according to the proposed scheme and the simulation trials are done for training of different beam line control agents and insertion device control agents so that they learn the sufficient skills for controlling the beam lines. Figure 7 shows the different beam positions at different BPM's throughout the INDUS-2 ring during one of the lower level agent ("DP3 agent") training using the proposed constrained gradient based reinforcement method. Figure 8 shows the different beam positions at different BPM's throughout the INDUS-2 ring for training of same agent using the general gradient based reinforcement method. Figure 9 shows the different skill states acquired by the agent during the training cycles for which the beam positions are shown in Figure 7 and 8. Figure 10 shows the skill states acquired by the agent ("DP3 agent") for the case of agent imparted with different initial skill levels. Figure 11 shows the skill learning by the agent ("DP3 agent") for the case of machine operating point changes due to different error introduced in the defocusing quadrupole (QD) family.

From Figure 7 one can clearly see that for the agent skill learning process using the proposed constraint gradient based reinforcement learning method is well capable of imparting the same skill level to the agent as general gradient based learning method while traversing almost the same path through skill set. Further it can be seen from Figure 7 that the proposed skill learning method is successful in limiting the beam perturbation in the constraint region (BPM0 toBPM6 and BPM10 to BPM55) within the constraint limits (±70μm) whereas the general gradient based method perturbs the beam to very large values (≈ ±2.5 mm) for similar skill set learning thereby making the beam lines unusable in this region at the time of training. From Figure 10 one can see that for the case when agent has acquired some skill (*i.e.*, the agent is having some initial skill learned according to some previously occurring state) and then in run time if the system changes to some other operating state then the agent's skill learning using the proposed method is well capable of converging to desired new skill from different initial skill sets. Further from Figure 11 one can see that the training iteration cycles required by the agent to acquire this new skill set (from the previously trained state) is limited to less than 30 cycles for the worst case considered of the parameter variations up to 0.75%. Thus for the practical case if the agent decides to go for training to improve the performance, only the related beam line will be affected for the period of less than a minute.

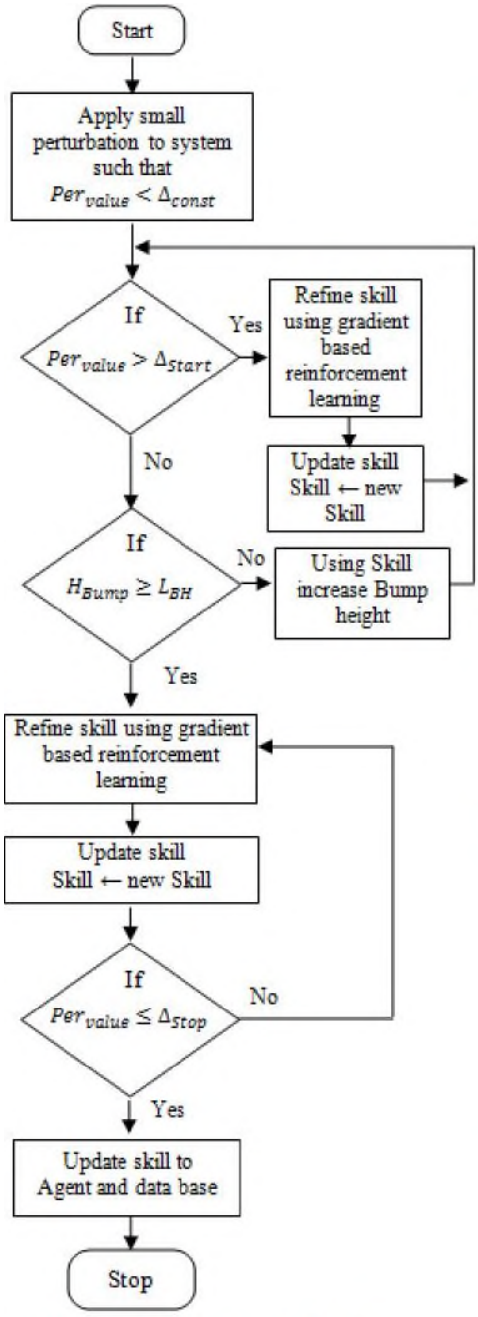


Figure 6. Flow chart for the constrained gradient based reinforcement skill learning

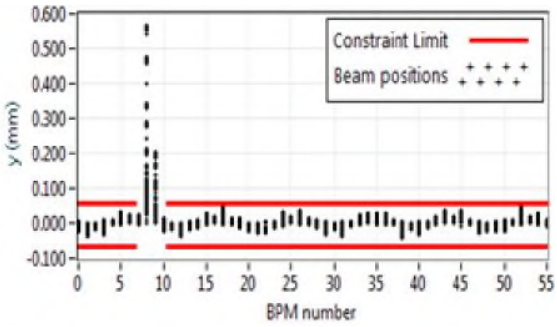


Figure 7. Beam positions on different BPM's during the skill learning using the constraint gradient based reinforcement learning

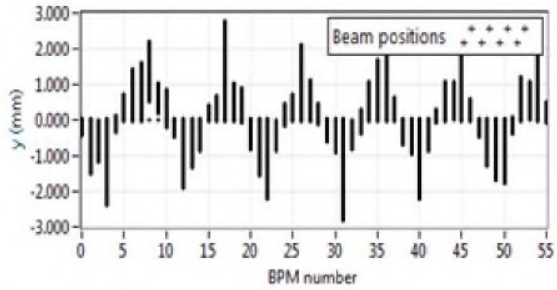


Figure 8. Beam positions on different BPM's during the skill learning using the general gradient based reinforcement learning

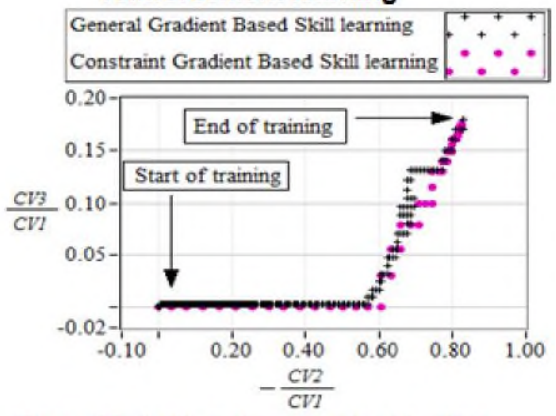


Figure 9. Skill states acquired by the agent during the training cycles for which the beam positions are shown in Figure 7 and 8

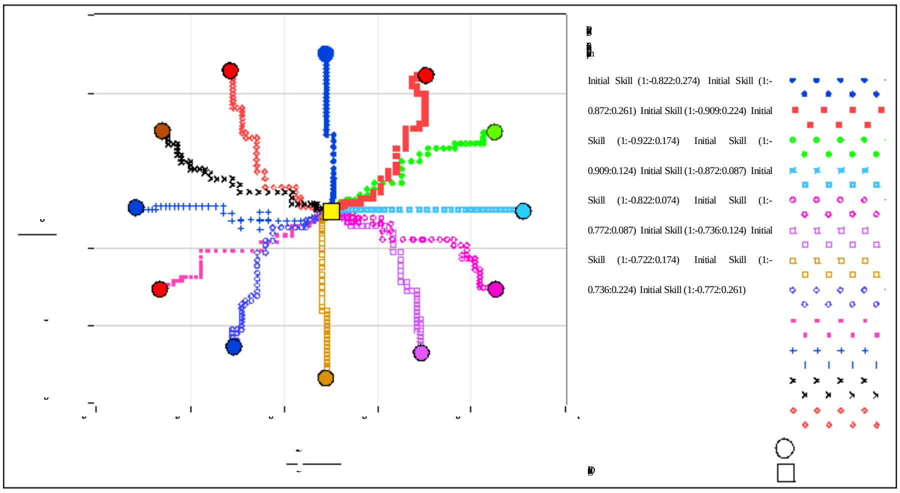
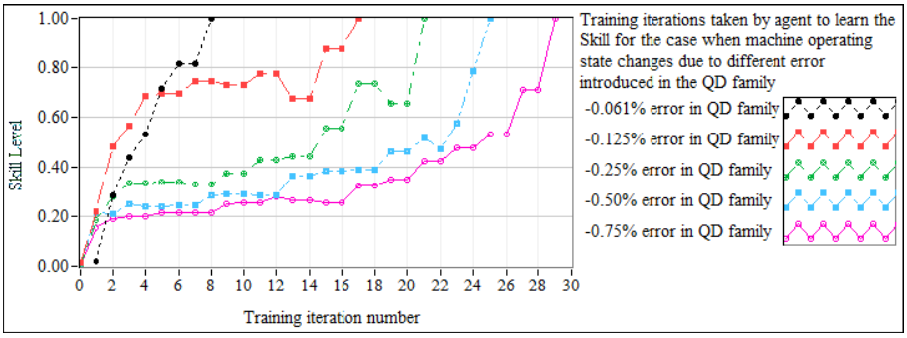


Figure 10. Skill states acquired by the agent ("DP3 agent") for the case of agent imparted with different initial skill levels



Initial imparted skill state

Figure 11. Skill learning by the agent ("DP3 agent") for the case of machine operating point changes due to different error introduced in the defocusing quadrupole (QD) family

8. Conclusion

Electron beam position and angle are controlled in SR sources for obtaining and maintaining the maximum photon flux at desired sample location in experimental SR beamlines. Towards this, a novel multi-agent based operator support and beam orbit control scheme is presented in this paper for improving the overall reliability and beam availability to the SR users. The design of different types of constituting individual agents is presented. This scheme is also helpful in improving the available beam quality to users through prioritized optimization for the cases when multiple beam lines demand the beam correction simultaneously. A novel gradient based cooperative multi-agent based reinforcement skill learning scheme is also presented for imparting training to lower level agents in case the skill level towards orbit control of individual agents degrades. The effectiveness of the proposed scheme is shown through simulation results obtained with the early implementation of the scheme with INDUS-2 model. The integration of the scheme with INDUS-2 machine control system is in

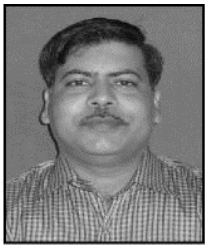
progress. Assuming that under dynamic environment the agents can successfully maintain their up to date skill levels with the presented skill learning method, then this distributed skill information can be further utilized for extracting the systems model which can be used to improve the system performance.

References

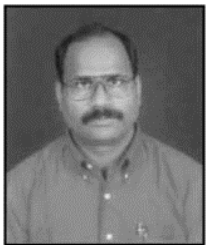
[1] H. B. Donald, E Pascal and W Edgar, "Review of third and next generation synchrotron light sources", Journal of Physics B: Atomic Molecularand Optical Physics, vol. 38, **(2005)**, pp. S773–S797.
[2] H. Wiedemann, "Particle Accelerator Physics", Springer, 3rd edition, **(2007),** pp. 750-788.
[3] J. Li, G. Liu, W. Li, B. Sun, C. Diao and Z. Liu, "Development of orbit correction system of HLS storage ring", Proceedings of the second Asian Particle Accelerator Conference, **(2001)** September 17-21; Beijing, China, pp. 681-683.
[4] Y. Chung, E. Kahana, J. Kirchman, A. Lumpkin, J. Meyer, E. Plouviez, K. Scheidt, E. Taurel, A. Ando, S. Sasaki and A. Taketani, "Local Beam Position Feedback Experiments on the ESRF Storage Ring", 1995 Particle Accelerator Conference, **(1996)** May 1-5; Dallas, Texas, pp. 2699-2701.
[5] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, vol. 2, no. 1, **(1986),** pp. 14-23.
[6] Y. Mo and Y. Xu, "A Navigation System Based on Subsumption Architecture", Proceedings of the 2010 International Conference on Measuring Technology and Mechatronics Automation", **(2010)** March 13-14; Washington, DC, USA, vol. 1. IEEE Computer Society, pp. 161-164.
[7] G. Morgan, "The application of multi-agent systems to the design of an intelligent geometry compressor", PhD thesis, Brunel University, London, **(2002)**.
[8] Y. Lesperance, "Foundations of a logical approach to agent programming", in Intelligent Agents II edited by Wooldridge M., Muller J. P., and Tambe M, Springer-Verlag, Berlin, Germany, LNAI, vol. 1037, **(1996)**, pp. 331-346.
[9] R. P. Yadav, P. Fatnani and P. V. Varde, "Intelligent agent based control of TL-1", In Proceedings of the 5th DEA-BRNS Indian Particle Accelerator Conference, **(2011)** February 15-18; Delhi, India.
[10] R. P. Yadav, P. V. Varde, P. S. V. Nataraj, P. Fatnani and C. P. Navathe, "Model-based Tracking for Agent-based Control Systems in the Case of Sensor Failures", International Journal of Automation and Computing, vol. 9, no. 6, **(2012)**, DOI: 10.1007/s11633-012-0680-y, pp. 561-569..
[11] J. D. Head, J. R. Gomes, C. S. Williams and K. Y. Lee, "Implementation of a multi-agent system for optimized multiobjective power plant control", North American Power Symposium (NAPS), **(2010)** September 26-28; Arlington, USA, pp. 1-7.
[12] G. Sheng, X. Jiang and Y. Zeng, "Optimal coordination for multi-agent based secondary voltage control in power system", IEEE/PES Transmission and Distribution Conference and Exhibition: Asia and Pacific, **(2005)** August 14-18; Dalian, China.
[13] F. Maturana, R. Staron, K. Hall, P. Tichy, P. Vrba, M. Vladimir, "Agent virtual machine for automation controllers", Robotics and Computer-Integrated Manufacturing, vol. 24, no. 5, **(2008)**, pp. 616-624.
[14] D. Chakraborty and S. Sen, "MB-AIM-FSI: A Model Based Framework for exploiting gradient ascent MultiAgent Learners in Strategic Interactions", Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Padgham, Parkes, Müller and Parsons (eds.), Estoril, **(2008)** May 12-16; Portugal, pp. 371-378.
[15] J. Baxter and P. L. Bartlett, "Infinite-Horizon Policy-Gradient Estimation", Journal of Artificial Intelligence Research, vol. 15, **(2001)**, pp. 319-350.
[16] T. Yamada and S. Yamaguchi, "Reinforcement Learning Using a Stochastic Gradient Method with Memory-Based Learning", Electrical Engineering in Japan, vol. 173, no. 1, **(2010)**.
[17] J. Choi, S. Oh and R. Horowitz, "Cooperatively learning mobile agents for gradient climbing", In the Proceedings of 46th IEEE Conference on Decision and control, **(2007)** December 12-14; New Orleans, Louisiana, USA, pp. 3139-3144.
[18] S. Magyary, M. Chin, C. Cork, M. Fahmie, H. Lancaster, P. Molinari, A. Ritchie, A. Robb, C. Timossi and J. Young, "The Advanced Light Source Control System", Nuclear Instruments and Methods in Physics Research A, vol. 293, **(1990)**, pp. 36-43.
[19] C. Stern, E. Olsson, M. Kroupa, R. Westervelt, G. Luger and W. Klein, "A Control System for Accelerator Tuning Combining Adaptive Plan Execution with Online Learning", International Conference on Accelerator and Large Experimental Physics Control Systems, **(1997)** November 3-7; Beijing, China.
[20] Webpage of Accelerator Control Section, RRCAT, **(2012)** November, http://www.cat.gov.in/technology/accel/indus/acs/i2ControlSystems.html.
[21] S. Y. Lee, "Accelerator Physics", World Scientific, Singapore, **(1999)**.

[22] W. Herr, "Algorithms and procedures used in the orbit correction package COCU", Technical report CERN SL/95-07(AP), **(1995)**.

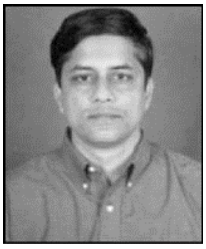
Authors



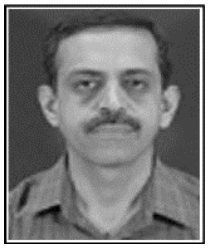
R. P. Yadav graduated from Sir Chhotu Ram State College of Engineering (CRSCE), Murthal, India, in 2000 and completed one year Orientation Course for Engineering Graduates and Science Postgr aduates (OECS) from Centre for Advanced Technology Training School Indore in 2001. He is currently working as Scientific Officer in Accelerator Control Section at Raja Ramanna Centre for Advanced Technology, Indore, India and is also pursuing Ph.D. degree from Homi Bhabha National Institute, India. His research interests include, feedback control system, accelerator control, agent based feedback control systems and intelligent control systems.



P. V. Varde joined Reactor Group, BARC in 1984 (27th batch of BARC Training School). He received the Ph.D. degree from Indian Institute of Technology, Bombay in 1996. He is currently working as Scientific Officer at Bhabha Atomic Research Centre (BARC), Trombay, Mumbai, India and heads Safety Evaluation & Manpower Training and Development Section (SE&MTDS). He is also a Professor at Homi Bhabha National Institute (HBNI), India.His research interests include Probabilistic Safety Assessment (PSA) of Research Reactors. He also works on application of AI tools like Artificial Neural Network and Knowledge Based Systems for developing PSA based systems for operations and fault diagnosis of nuclear plant.



P. S. V. Nataraj received the Ph.D. degree from IIT Madras, India in process dynamics and control in 1987. He is currently a Professor with the Systems and Control Engineering Group at IIT Bombay since 1988. He is the chief coordinator of the Control, Automation, Reliability, Instrumentation, Measurement, and Optimization (CARIMO) cell setup at IIT Bombay to foster interaction with the industry in the areas of Control, Automation, Reliability, Instrumentation, Measurement, and Optimization. He received a NASA/US Navy commendation and memento for the Best Applied Research Paper awarded at the International Symposium on Air Breathing Engines (ISOABE), Cleveland, Ohio, September 2003.His research interests include Robust and Fractional Order Control, Global Optimization, Reliable Computing, Parallel Computing (GPU), and Robust Statistics.



P. Fatnani graduated from Pandit Ravishankar Shukla University, Raipur, India, in 1985 and completed one year Orientation Course for Engineering Graduates and Science Postgraduates (OECS) from Bhabha Atomic Research Centre Training School in 1986. He is currently working as Scientific Officer at Raja Ramanna Centre for Advanced Technology (RRCAT), Indore, India and heads Accelerator Control Section. His research interests include, accelerator control systems, agent technology, Supervisory Control and Data Acquisition.