Incorporating Adaptivity using Learning in Avionics Self Adaptive Software: A Case Study

Rajanikanth N Kashi *
Honeywell Technology Solutions Laboratories
Bangalore, India
rajanikanth.kashi@honeywell.com
(* Corresponding Author)

Meenakshi D'Souza
S Kumar Baghel
Nitin Kulkarni
IIIT-Bangalore, India

***Abstract*—Self Adaptive software is forging its way into avionics. Such software, while being adaptive, needs to meet safety, determinism, and real time responsiveness, like all avionics systems. We model avionics self adaptive software as a multiagent system. Each agent uses a BDI (Belief Desire Intention) model for adaptiveness and also incorporates learning to address several constraints. We illustrate our approach using a case study of adaptive Flight Management System (FMS). Our BDI model of adaptive FMS in Netlogo is a model that is adaptive while being deterministic and also responds in real-time. We propose a learning algorithm that agents use to adapt themselves better and also a way of measuring their adaptivity that provides quantitative gains illustrating the system's adaptability.**

Keywords—Avionics; Self Adaptive Software; Multiagent Systems; BDI Model; Reinforcement Learning; TD Learning; Measures for adaptibility;

I. INTRODUCTION

Changing user requirements coupled with dynamic changes in the user environment is one of the motivations to turn to self adaptive systems. In the context of our avionics case study, we identify how a user need to obtain best possible flight plan represents a changing user requirement and is coupled to uncertainties in the perceived user environment due to the continuous change in location of the user vis-à-vis the constraints that unfold. Traditional software engineering methods rely on a definitive set of rigid requirements and design principles to engineer systems for many such seemingly uncertain behavioral contexts [1].

Self adaptive systems provide a promising track to deal with attendant problems but need to address many issues. In the context of engineering self-adaptiveness, it is to be recognized that software is the primary means by which this adaptation is obtained. Therefore, the focus of this paper is self-adaptive software in the avionics domain and its mechanization via a learning mechanism. The aviation industry has been looking at adaptive software as a solution for addressing the needs arising from increased automation in the avionics systems of modern aircraft. The attendant complexity of such systems poses many research challenges. Many areas within the civil aviation sector have been identified for such systems that operate as ground and on-board applications. The attention is on the latter category of applications which needs to address the unique challenges related to safety, determinism, and real-timeliness. The associated problem of obtaining assurance in such systems is a matter of current research. One of the important questions before an adaptive system is proposed as a candidate system for the safety critical domain of avionics is whether it can provide considerable benefit compared to traditional approaches. Considering the physical and mental workload of the pilots [2], the flight management system on board represents a suitable candidate where an adaptive system can possibly aid the decision making process. Determinism and real-timeliness are aspects closely related to dependability, meaning that the system should provide sufficient assurance that it can be used in such contexts. Before we start dealing with these challenges, we present a brief background on the topic of self adaptive software and systems. The narrative cites relevance of these different aspects to our case study.

"Adaptation is change of system properties and behavior at runtime in response to dynamically varying user needs, resource constraints, and changing environments" [3]. The spectrum of software adaptation ranges from conditional expressions at the bottom (of the spectrum), moving onto online algorithms, generic/parameterized algorithms, algorithm selection methods to evolutionary and machine-learning techniques at the top (of the spectrum) [4]. In our approach for the case study example, we operate at the top of this spectrum. Different degrees of adaptation are recognized: Behavioral, Component, and Architectural adaptation. The case study example utilizes the behavioral adaptation approach. A term that is generally associated with adaptive systems is 'Evolution': Open-adapted systems allow adaptation of behaviors arising due to information shared by peer level software components (at runtime, adaptation policies updated dynamically) while closed-adapted systems do not incorporate such a mechanism and adapt in isolation (fixed adaptation policies). The exemplar case study uses the open adaptation policy. Generally speaking, an adaptation loop consists of four processes: Detection, Decision Making, Acting, and Monitoring. This loop has close parallels to adaptive control loops in control systems, where there are a number of methods to verify system behavior [5]. There are a number of techniques for engineering adaptive systems: dynamical systems, game theoretic models, control-theoretic models, evolutionary computation, state transition systems, social

978-1-5090-2029-4/16/$31.00 @2016 IEEE

network analysis, agent based simulations, rule based systems, and Multiagent systems. In our approach, the adaptive software for the representative avionics system is engineered using Software Agents, the rationale being [6] : (a) Agent approaches provide a higher degree of autonomy and distributed decision making (b) Agents are effective in a highly dynamic and incompletely known environment (c) the pattern of Monitor/sense followed by Analysis/Reasoning and subsequent enactment of appropriate behaviors (d) the concept of utility for each element and the endeavor to achieve individual goals which will collectively contribute towards systemic goals.

We choose a representative avionics system – namely a hypothetical Flight Management Aiding System in the terminal area as a case study. A use case scenario is described for the case study which provides the requirement and background necessary to outline an adaptive system. Thereafter, an agent based model is conceptualized for the hypothetical system. There are different types of agent architectures: Deduction/Logic based, Reactive, BDI, and Hybrid. Specifically, a BDI [7] model is chosen as the agency mechanism and is used to engineer the adaptivity; the reasons for this choice is outlined in a subsequent section of this paper. An agent performs more efficiently when learning is incorporated. However, the pure traditional BDI architecture is weak in the context of learning [8]. However, the area of learning in BDI is continuously evolving and recent advances [9, 10] present representative methods incorporating the same. We present a slightly different method to address this important aspect in the context of dependability assurance for the system. A consequence of this approach with learning is an interesting insight gained for the system and its relationship to the environment. A proposal is made of a measure that characterizes the degree of cooperativeness that exists among a society of agents while they co-operatively solve a problem /subtasks or equivalently adapt the system to its environment. We also provide data that describes how the co-operation varies with disturbances or changes in the environment. In this context, the relative co-operation that exists between a pair of agents is captured. With a view to emphasize the overall utility of our approach, we also provide a measure that characterizes the gain that one can obtain using this approach.

The outline of the paper is as follows: Section 2 captures important considerations for incorporating learning in avionics self-adaptive software. Section 3 provides a short background for the problem of adaptive flight planning and introduces a representative Flight Management System which forms the case study exhibit. The system is adaptive and performs appropriate actions to changing environmental conditions in subject airspace. This Section also lays out broad requirements in the context of safety, determinism, and real time responsiveness in combination with learning specification for the case study. Section 4 introduces the model of case study cast as a distributed multi-agent system with each agent structured on the BDI architecture. We discuss aspects of learning within the BDI model. Aspects that embody dependability are discussed. Section 5 provides the motivation for learning and the various methods available. The incorporation of learning within agents that compose the system is also discussed. Some measures related to adaptivity/co-operation is provided and results obtained with experiments during system simulation is elaborated. Section 6 details related work. Section 7 provides a summary of our work and presents directions for future work.

II. AVIONICS SELF ADAPTIVE SOFTWARE

A. Considerations for Adaptation and Learning in Avionics

We present seven different considerations that touch upon aspects of adaptation and learning since the two are closely related.

- Reference [11] asks a fundamental question: Does the learning (algorithm) provide a significant benefit (in terms of safety or performance) compared to current approaches? We answer this question in Section 5 and indeed show that our approach provides significant benefits.

- The adaptation approach is an important aspect. A system that incorporates learning from and which adapts itself to its environment [12] may possess more capabilities than one that cannot. We answer this question too in Section 5.

- An adaptive system is described by its self* properties: Self-managing, self-awareness, self-configuration, self-healing, self-protection, self-optimization [4]. According to [4], self-* properties are those that provide some degree of variability, and consequently, help to overcome deviations from expected goals. Examination of two of these properties leads to the following arguments: (a) The property of self optimization is one of the primary system goals and since we have engineered the system to be composed of various agents, each agent can choose a desire (BDI Model) and provide a partial plan to solve the problem on hand for the most optimal desire applicable at that instant using a learning function. (b) The conceived system can be highly self aware since each individual agent adjusts its behaviors with every cycle. There is a possibility to build 'learning' into each agent that instructs which desire to choose from in a context.

- The Adaptation Loop is the main mechanism engineering the self* properties. The adaptation loop comprises the general processes of monitoring, detection, decision, and acting. The decision process could be driven by a learning engine embodying a learning algorithm that is fast (converges quickly) considering real time needs of the solution by generating desires categorized into specific levels, and adjusted according to the situation.

- In the context of choosing an adaptation type, we have leaned towards using a model free adaptation wherein there is no predefined model of the environment or system by using Reinforcement Learning. This approach helps us to generalize the method for problems that are similar, since knowledge gained from previous attempts are utilized to arrive at decisions.

- Learning is closely related to AI (Artificial Intelligence) and usage of planning is a prime area of our focus. Our agent based system uses continuous planning/re-planning versus an algorithmic approach.
- Certification plays a significant role for the usage of technologies available for adaptive systems [11]. The considerations for certification bring forth issues that are encountered when dealing with avionics systems, particularly those that are planned for civil aircrafts, primarily due to the safety expectations and the attendant stringent requirements and processes. The decision making process with the avionics adaptive systems assumes a central focus since this is the area where a numbers of issues arise. The adaptation stems from decision making after learning that accrues over time and experience. The decision making therefore is entwined with non-determinism that is associated with the learning. In section 5, we discuss about the different types of learning that is suitable for our application and also present a design philosophy that attacks the problem of non-determinism of the adaptive algorithm.

III. ADAPTIVE FLIGHT PLANNING: A CASE STUDY

A. *Adaptive Flight Planning System (AFPS) Overview*

Aircrafts flying in the terminal airspace need to get clearance from the subject Air Traffic Controller (ATC) to complete the transitions that they wish to perform. An exemplar situation is one when an aircraft using instrument aids and coming into land at a runway, will need to generally adhere to a Standard Arrival Chart (STARS) on which is indicated various segments of the flight path and the altitude levels that it has to adhere to. The pilot needs to evaluate clearances from the ATC and then accept or reject such clearances. The pilot needs to evaluate conditions that will prevent the pilot from accepting the clearance – Weather, approaching Traffic, and obstacles (Terrain) and follow the route to the runways indicated on the STARS chart. Therefore the path that the aircraft needs to take will need to be adapted suitably when constraints of weather, traffic, and terrain may be in place for such decisions. We make two assumptions for our hypothetical system: One, the pilot is free to choose an endpoint on the STARS chart as the final destination once he enters the terminal area. The end point so chosen will continuously change based on weather, traffic and terrain constraints. Two, we allow the aircraft to choose a different route when conditions on a route chosen become unfavorable due to any of the constraints of weather/traffic/terrain. The associated pilot workload [2] under such circumstances imposes a need for automation. A general flow chart of such a system is shown in Figure 1.

In the case study that we propose, the aircraft enters the Chennai (India) Terminal airspace [13] at the top right corner of the STARS chart (north east) around the HYDOK Waypoint (See Figure 1). The goal is to reach any of the endpoints MM515 or MM513 or MM 512 or MM510.

B. *Requirements Specification for Adaptation and Learning*

In this section, we elicit broad requirements related to adaptation and learning for the adaptive flight planning system outlined in the left hand side of Figure 1. Generally speaking, a pilot uses knowledge of the environment obtained by sensors on board the aircraft and plans a path using judicious decision making skills, after examining an ATC Clearance that has conflicts with one or more criterion. The sensors provide information about weather, traffic and terrain. The proposed system is required to reconcile all conflicts to choose the optimal route.

- Introduction of the notion of elements/entities for adaptation: The system shall find a path to the next waypoint in the terminal area that is free of conflicts from weather, traffic, and terrain constraints. Each constraint may be handled by a separate functional element/entity.
- Introduction of the notion of what to adapt: Each constraint handled by an element/entity can be solved by associating a desire level in which the constraint is solved. Therefore, we can state a requirement as follows: A route shall be chosen by an element that is optimal w.r.t each constraint that is considered, without compromising safety. Optimality is synonymous with highest desire level. As an example, the highest desire may correspond to a route with least distance and heading to a waypoint, moderate corresponding to a route with least distance only, and the lowest desire corresponding to that with least heading.
- Introduction of the notion of a mechanism to adapt: Each element/Entity that is responsible for handling a constraint shall negotiate and agree upon a mutually acceptable solution by changing/adjusting its desire level.
- Introduction of the notion of learning during adaptation for a rapid agreement between the entities leading to a solution: A utility value shall be associated with each desire level and each element/entity handling the constraint will choose a desire that maximizes this utility value.
- Introduction of the notion of learning convergence for assuring safety property of the system: The system shall issue a warning (to divert to an alternate airport) when with the fuel remaining onboard the aircraft, it appears improbable to find a route (terminal area waypoint) that is weather, traffic, and terrain constraint free.
- Introduction of the notion of learning convergence for meeting realtimeliness in the system: The system shall issue a warning (to divert to an alternate airport) when distance trending checks show that the route is diverging from its intended goal for a specified amount of time.

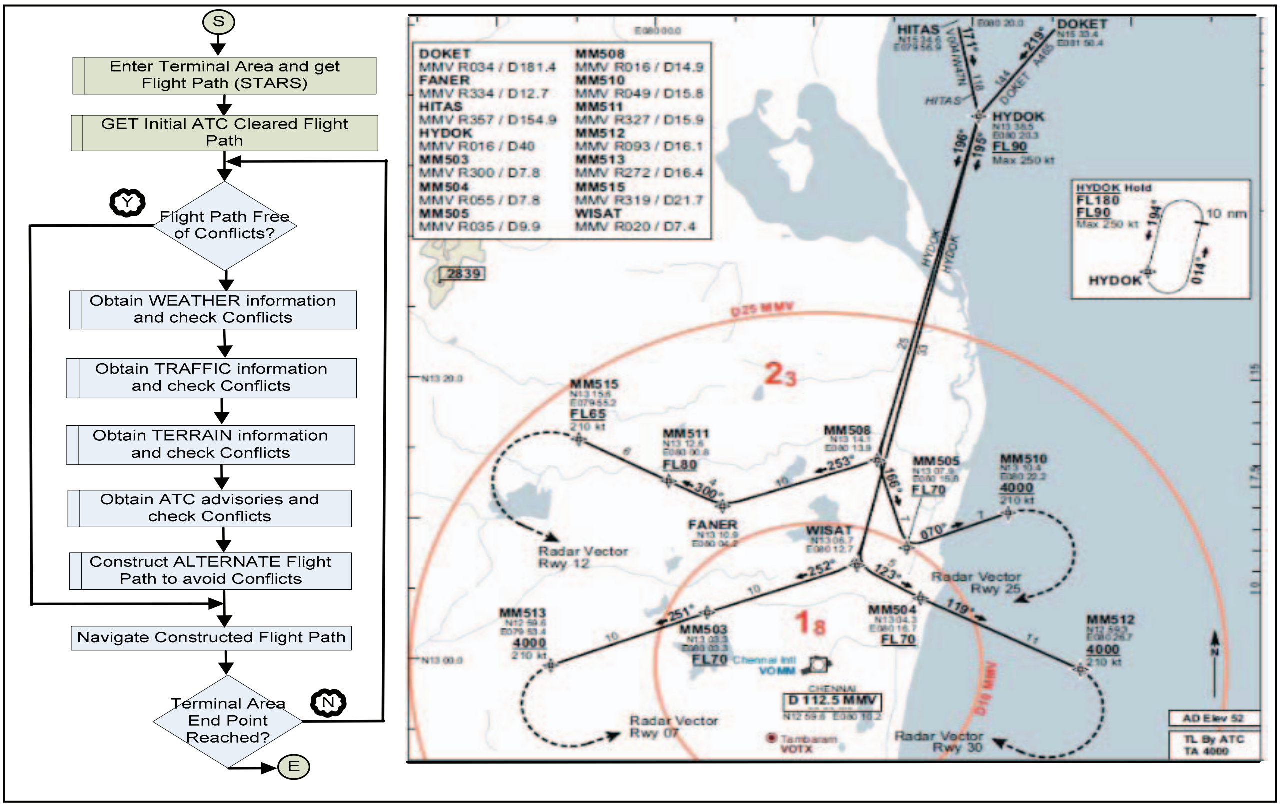


Fig. 1. Terminal area adaptive flight planning problem

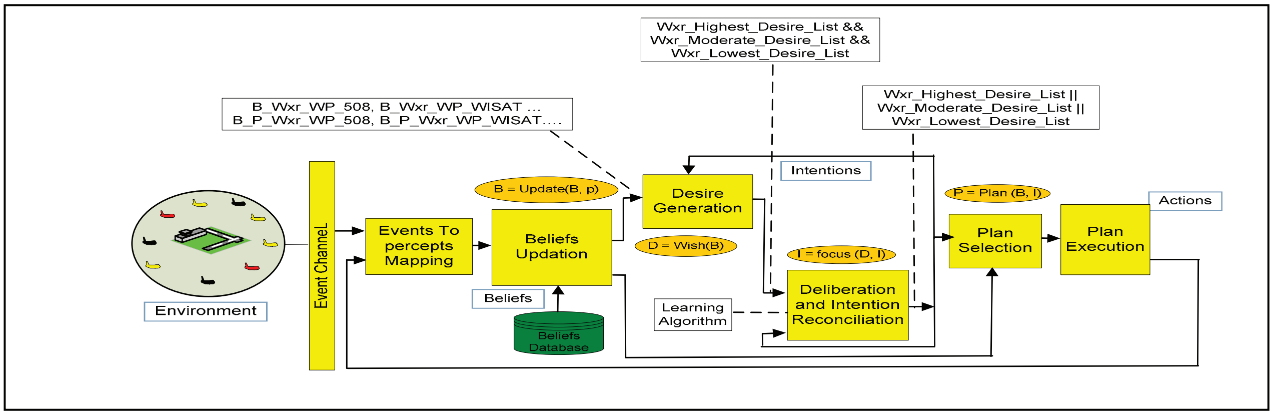


Fig. 2. The BDI Model of agency for the adaptive flight planning problem. (*figure caption*)

IV. BDI MODEL OF ADAPTIVE FLIGHT PLANNING

A software Agent is an autonomous software entity that interacts with the environment and adjusts itself and its actions based on the inputs from the environment, with reactivity, proactivity, and cooperation being the salient characteristics. The advantage of casting adaptive software as an agent based system is that agent based systems support higher degree of autonomy and distributed decision making. The Literature cites several models of agency exist, but we have chosen the Belief-Desire-Intention (BDI) model of Agency [7], the reasons being: (a) it is parallel to psycho-logical and philosophical human behavior (b) its ability to be formulated and reasoned in terms of logical semantics [7] (c) response in uncertain environments with a right balance of pro-active goal seeking behavior and reactive response (d) existence of many practical implementations like PRS and dMARS [14]. The second characteristic above provides the means to perform model checking which forms the basis of proving correct behavior. The generic BDI model adapted to our AFPS is shown in Figure 2. Beliefs are facts representing what an agent believes about the world, including the environment, obtained by each agent from the environment and other agents. Desires are the goals or end states that the agents wish to achieve. With beliefs and desires as inputs, agents deliberate and reconcile to come up with a plan, part of which also depicts an agents set of intentions. Intentions are committed desires. Agents then need to execute their plan as actions or re-plan if a particular plan is found to be infeasible.

Examination of the entire flight planning problem helps us arrive at the list below for the various subsystems. Each subsystem except for the Controller receives relevant data from its associated sensors and is responsible to provide a flight path that is free of conflict for the constraint that it handles. The Controller is then responsible to aggregate data from each of these sensors and provides an overall solution that will solve all of the constraints. The mapping of subsystems to agents is straightforward applying principles of functional identification, cohesion, separation of concerns, and data exchange at the interfaces.

Weather Subsystem → Weather Agent
Traffic Subsystem → Traffic Agent
Terrain Subsystem → Terrain Agent
ATC Subsystem → ATC Agent
Controller Subsystem → Negotiation Agent

Weather, traffic, terrain and ATC agents have a belief revision function that updates the beliefs. The desire function within each agent generates a set of partial plans based on the different desires (goals). One such partial plan from each agent is provided to the negotiation agent for adaptive flight planning. Within each agent, the decision to use one such partial plan is arrived at, through a learning algorithm. The learning algorithm that we use is discussed in the next section.

The negotiation agent acts on the partial plans produced by each agent and has a filter function that computes a negotiated set that is the final plan. This negotiated final plan represents the 'negotiated set of desires' and indicates a safe route to an end point in terminal area through appropriate way points. There is another filter function that computes the waypoint that needs to be navigated to, in the immediate context. This latter decision forms the intention of the negotiation agent.

We have implemented the above BDI model in an open source tool NetLogo [15], a popular tool to model several multi-agent systems. Our BDI model has agents as described above and also implements the functions for beliefs, desires along with the learning and filter algorithms as briefed above.

As an exemplar description of one agent, we describe the weather agent characterization in our NetLogo implementation. Other agents follow a similar approach. The weather agent maintains two kinds of beliefs about intermediate and terminal waypoints. The first is about whether a waypoint is weather constraint free and is available for navigation to it currently. The second is about whether a waypoint is weather constraint free and is available for navigation for the next (say) 5 minute window. The latter is a set of predicted beliefs. Both of these aspects are shown in the box with a dashed line attached to the output of the 'Belief Updation' function in figure 2.We assume that sensors provide the 'percepts' of the current and predicted weather situation. The weather agent directly maps these percepts onto beliefs. Feasible waypoint lists are generated examining these base beliefs and segregated according to the desires. This is shown in the box with a dashed line attached to the output of the 'Desire Generation' function in Figure 2. In our case, these are highest, moderate, and lowest desire lists. The desire generation and selection thus operates in two stages: In the first stage it generates the partial plans (options generation function, (Opf-1)), and in the next stage chooses the 'Best Desire List' among the three using a learning algorithm (Opf-2). This can be seen in Figure 2, at the output of the 'Deliberation and Intent Reconciliation' function. The Learning algorithm is used to attach a value to each of the generated lists within the individual agents looking at 'Negotiated Final Desire List' that is provided by the Controller. Looking at the 'Negotiated Final Desire List', each Agent learns what would be the most preferred list to send to the Negotiation Agent, so that its list is accepted. Therefore, for each individual agent handling the constraint, generation of the 'Best Desire List' is equivalent to intention generation.

The 'Best Desire Lists' (waypoint lists from multiple agents are reconciled within the negotiation agent which uses a simple one-shot negotiation for real-time considerations. The Negotiation Agent is formulated with two filter functions filter-Neg (for Negotiation) and Filter-Int (for Intention). Filter-Neg implements the one shot negotiation between weather, traffic, and Terrain agents, wherein the waypoint lists from each agent is compared with the other to find agreeable common waypoints. This common waypoint list forms the 'Negotiated Waypoint List' During each execution cycle, the current waypoint to which the aircraft is heading is set to the head of the 'Negotiated List'. The Intention function (commitment) checks if there are changes brought about during an execution cycle to the negotiated list that introduces a new waypoint in the list that is currently not in it. Only when this happens the intention changes. Also, during execution of an agent cycle, temporal constraints are checked and a warning is generated if it is exceeded. The fuel remaining within the system is also tracked within a global cycle that evaluates it continuously and a warning is issued if there is an exceedance. A distance trending check (mechanism) is also evaluated to monitor whether the final solution is converging. A simple execute function navigates the current waypoint (housed within negotiation agent or with a separate and distinct entity).

V. LEARNING IN THE BDI MODEL

In order to incorporate a learning mechanism, with our BDI agents, we need to consider the following aspects: (a) Which aspect of functionality or performance is best addressed by the learning mechanism (b) What parts of the BDI model should host the learning mechanism (c) What is the best representative structure or mechanism to Integrate learning within the BDI (d) How do we engineer the environmental and internal feedbacks into an appropriate learning mechanism (e) convergence of the algorithm for real-time response.

We recognize that there are three distinct types of machine learning identified in the literature: Supervised learning, Unsupervised learning, and Reinforcement learning (RL) [17]. For the problem on hand of adaptive flight planning system, we have chosen the last category of learning. This follows from our observations that every flight is unique when considerations of weather, traffic, terrain, ATC clearances are considered. Reinforcement learning offers a promising route since this type of learning utilizes the concept of a reward that a system receives for the response it provides. The flight planning system that we propose is engineered as a composition of agents, each of which receive a reward which is a 'feedback' for the system. This is used to 'adapt' the behavior of the system. Therefore each agent in the adaptive flight planning system tries to maximize the reward in trying to find a solution for its constraint. This is in essence our basic principle for incorporating learning into the AFPS. The learning is incorporated on the open/closed belief sets of waypoints in the terminal airspace. As stated earlier, we incorporate the learning mechanism into the selection of Desire generation process. A judicious choice of a 'feature' of the environment is used to engineer the feedback mechanism. An appropriate algorithm is chosen for real time response and convergence.

A. *Temporal Difference Learning*

Temporal Difference (TD) learning algorithm [18] is characterized by the following equations:

$$\text{sample} = R\,(s, \pi, s') + \gamma\, V\pi(s') \qquad (1)$$

$$V\pi(s) = V\pi(s) + \alpha\,(\text{sample} - V\pi(s)) \qquad (2)$$

where R is a reward function, s ϵ S, as set of states and a ϵ A, a set of actions that are taken in moving from one state to another, s′ is the next state of s, π is a policy ∍ π: S→A, γ is a discount factor, and α is the learning rate. Equation (2) is called the update equation and updates the value of states each time a new experience is obtained (s, a, s′, r). V is a value function that associates a value to each state.

We have chosen the Temporal Difference (TD) algorithm for incorporating learning within our proposed Flight Management System. The reasons for this choice are:

- TD learning is model free – where we do not know a model (T) or reward function (R) {A model T can be defined as T(s,a,s′), where s ϵ S, as set of states and a ϵ A, a set of actions; Reward function can be defined as R(s, a, s′). Short term weather, traffic, and terrain (when position changes) in the terminal area are stochastic processes and cannot have a precise model.

- The policy π can be mapped to the method/plan to satisfy the goal/desire of an agent represented by: HDxx, MDxx, LDxx, where xx is the constraint as noted earlier in section III B, and HD, MD, LD are highest, moderate and lowest desire qualifiers. Each Agent (Weather, Traffic, Terrain) uses and examines these policies. It is desirable that each agent operate with the best policy (optimal flight path planning).

- The value function for a policy is most appropriate, where the exploration policy π is executed to estimate the value of states that will provide the highest value. Our mechanization of the value function (provided in the next section) is such that, when a temporal difference is used, we will observe that this has an effect of moving the state values towards whatever successor state occurs. This running average will direct the system towards its final goal of reaching the terminal endpoints

- The TD algorithm converges quickly [18, 19] and is therefore suitable for the real-time response of the system

B. *Mechanizing TD learning within BDI Agents for Adaptive Flight Planning*

In order to understand the mechanization of TD learning algorithm within our system, we first introduce the abstract finite state transition system model. Figure 3 shows the finite state model abstracted from our NetLogo [15, 16] implementation. This model exhibits predominantly Boolean behavior that is sound with respect to the BDI model and is used for the purposes of verification by model checking [20] which is not discussed further in this paper.

A cycle of the system starts at the top with a check of the flight plan and fuel sufficiency at the top (s0). A check is made in s1 to ensure that all sensor updates (weather, traffic, and terrain) are captured. States (s2, s3, s4, s5, s6) within the extreme left swimlane constitute the weather agent mechanization, while states (s7, s8, s9, s10, s11) in the middle swimlane constitute the traffic agent mechanization, and the states (s12, s13, s14, s15, s16) in the right swimlane constitute the traffic agent mechanization. The function brf-Xx-Conf-free-WPTs$|_t$ (produces the set of beliefs that indicates waypoints that are conflict free with respect to the constraint Xx, at time t), is associated with states s2, s7, and s12. Similarly, the function Opf-1-Xx$|_t$ is associated with states {s3, s4, s5}, {s8, s9, s10}, and {s13, s14, s15}. The function Opf-2-Xx$|_t$ is incorporated with states s2, s7, s12. The function Filter-Neg$|_t$ is associated with state s20 while function Filter-Int$|_t$ is associated with state s22.

The central focus for this paper is the mechanization of Opf-2-Xx$|_t$. Each Agent evaluates a value function associated with a desire - classified as high, moderate and low. The desire classifications are correspondingly mapped to the requirements of waypoint prioritization order ((Distance + Heading), (Distance), (Heading)) mentioned in Section III B. When an agent first finds a set of waypoints as the solution set {Beliefs-Xx-Conf-free-WPTs}$_t$ (the set of beliefs that indicates waypoints that are conflict free with respect to the constraint Xx, at time t), we define the value of the states s3, s4, s5, with:

$$V(s) = K1\,/\,(\|\{\text{Beliefs-Xx-Conf-free-WPTs}\}_t\|) \qquad (3)$$

Where

$$\|\{\mathrm{Beliefs} - \mathrm{Xx} - \mathrm{Conf} - \mathrm{free} - \mathrm{WPTs}\}t\| = \sum_{n=1}^{NCFW} (\mathrm{DIST} - \mathrm{WPTS} - \mathrm{RWY})n \qquad (4)$$

Note that {Beliefs-Xx-Conf-free-WPTs}$_t$, can be segregated into three sets based on the waypoint prioritization order (D+H, D, H) and therefore this results in generation of {Desires-Values}$_t$. These map to the policies π1, π2, and π3. K1 is a constant (set equal to 100 in our experiments) and (DIST-WPTS-RWY) denotes the distance of each of the conflict free waypoints from the runway, considering the constraint Xx. We then perform a summation of distances of all such waypoints (n), the total number being denoted by NCFW. Observe that we have used a 'feature' based representation, which is a function that maps a state to a real number.

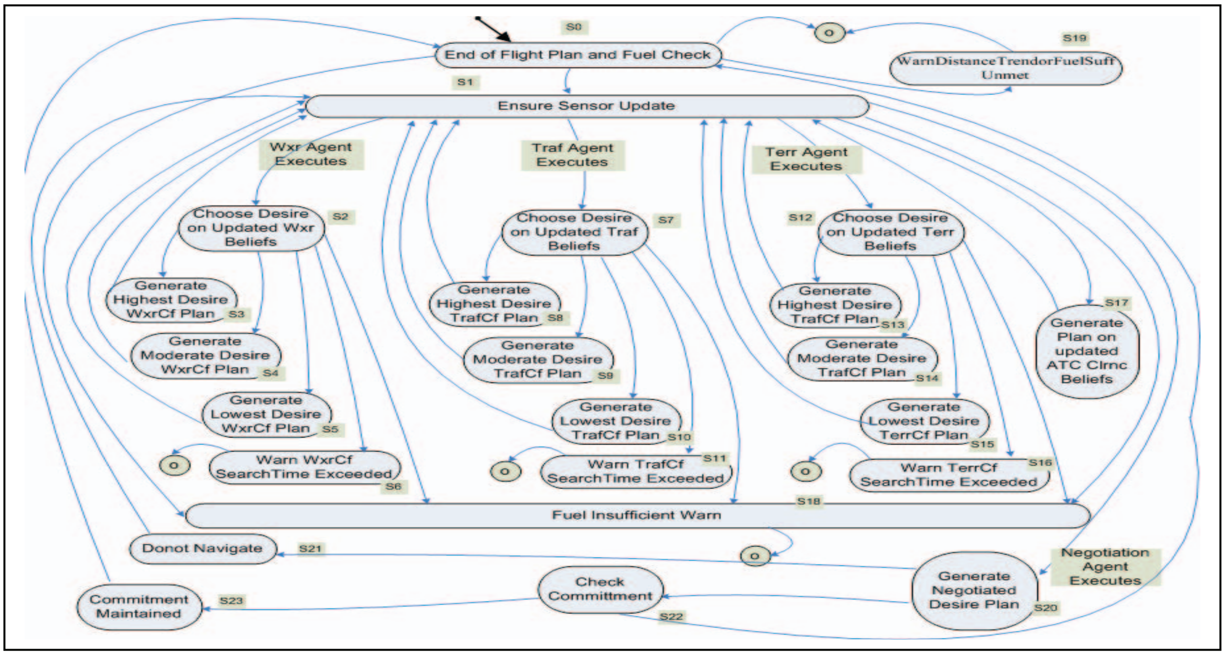

S0
End of Flight Plan and Fuel Check
S19
WarnDistanceTrendorFuelSuff Unmet
S1
Ensure Sensor Update
Wxr Agent Executes
Traf Agent Executes
Terr Agent Executes
Choose Desire on Updated Wxr Beliefs
S2
Choose Desire on Updated Traf Beliefs
S7
S12
Choose Desire on Updated Terr Beliefs
Generate Highest Desire WxrCf Plan
S3
Generate Moderate Desire WxrCf Plan
S4
Generate Lowest Desire WxrCf Plan
S5
Warn WxrCf SearchTime Exceeded
S6
Generate Highest Desire TrafCf Plan
S8
Generate Moderate Desire TrafCf Plan
S9
Generate Lowest Desire TrafCf Plan
S10
Warn TrafCf SearchTime Exceeded
S11
Generate Highest Desire TrafCf Plan
S13
Generate Moderate Desire TrafCf Plan
S14
Generate Lowest Desire TerrCf Plan
S15
Warn TerrCf SearchTime Exceeded
S16
S17
Generate Plan on updated ATC Clrnc Beliefs
O
S18
Fuel Insufficient Warn
Donot Navigate
S21
Negotiation Agent Executes
Generate Negotiated Desire Plan
S20
Check Committment
S22
Commitment Maintained
S23

Fig. 3. Terminal area adaptive flight planning problem.

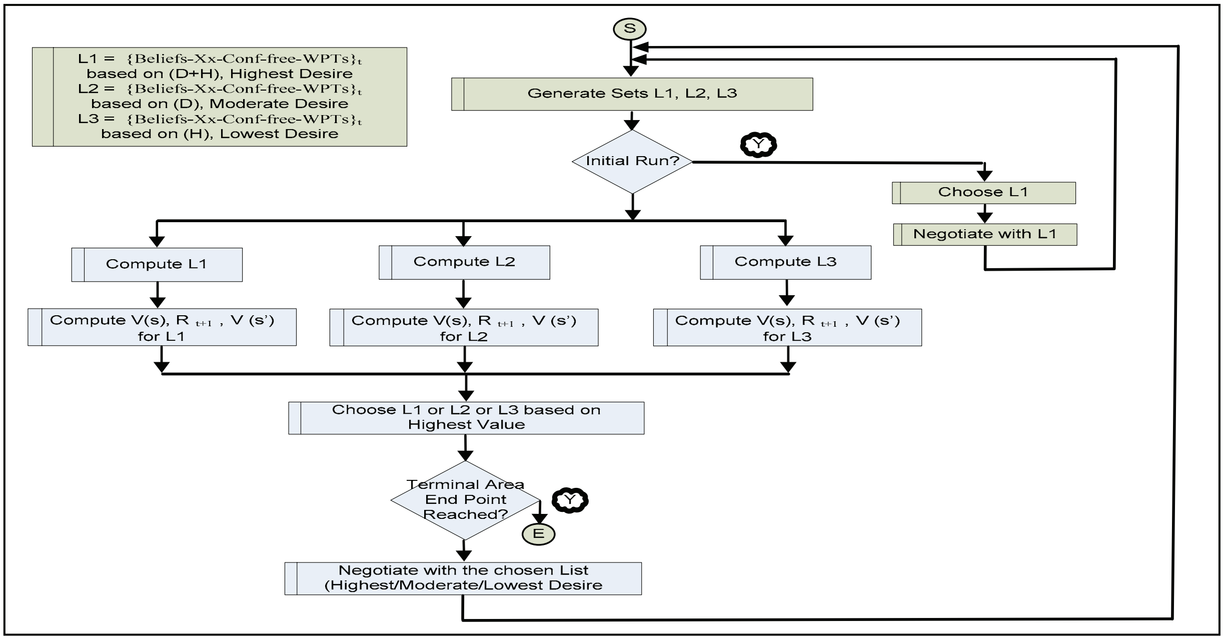

L1 = {Beliefs-Xx-Conf-free-WPTs}t based on (D+H), Highest Desire
L2 = {Beliefs-Xx-Conf-free-WPTs}t based on (D), Moderate Desire
L3 = {Beliefs-Xx-Conf-free-WPTs}t based on (H), Lowest Desire
S
Generate Sets L1, L2, L3
Initial Run?
Y
Choose L1
Negotiate with L1
Compute L1
Compute L2
Compute L3
Compute V(s), R t+1 , V (s') for L1
Compute V(s), R t+1 , V (s') for L2
Compute V(s), R t+1 , V (s') for L3
Choose L1 or L2 or L3 based on Highest Value
Terminal Area End Point Reached?
Y
E
Negotiate with the chosen List (Highest/Moderate/Lowest Desire

Fig. 4. The mechanization of TD Learning within an Agent.

The calculation of the reward for each agent is given by:

R_{t+1} = K1 / [(||{Beliefs-Xx-Conf-free-WPTs}$_t$||) ~ (||{Beliefs-Neg-Conf-free-NAV-WPTs}$_t$ || (5)

Where {Beliefs-Neg-Conf-free-NAV-WPTs}$_t$ represents set of waypoints found conflict free with respect to all of the constraints, ATC cleared (navigable waypoint set)

We recognize that:

$$V(s') = K2 / (||\{Beliefs\text{-}Xx\text{-}Conf\text{-}free\text{-}WPTs\}_{t+1}||) \quad (6)$$

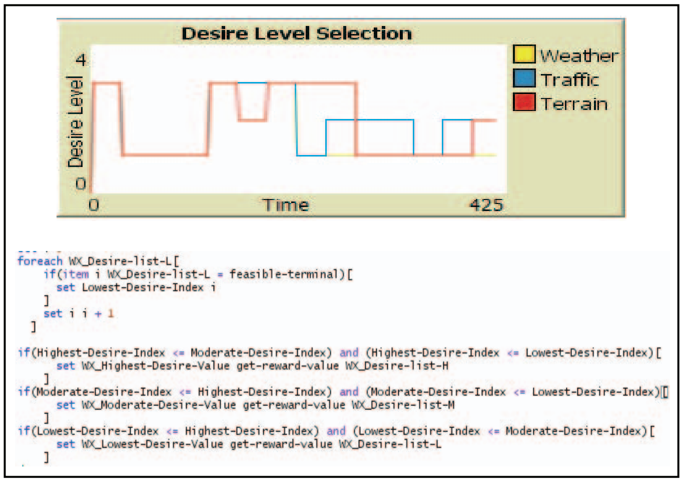
In calculating ||{Beliefs-Xx-Conf-free-WPTs}$_{t+1}$||, we use the same formula provided in (4) but the distances are decreased by an amount that the aircraft would have travelled in $\Delta t = 1$ time unit. The same arguments and propositions are valid for states {s8, s9, s10}, and {s13, s14, s15}. The function opf-2-Xx|$_t$ is the learning function and implements equations (3) through (6). K2 is a constant (100 in our experiments). Filter-Neg is a negotiation function and implemented as a simple match and retain algorithm within Negotiation Agent.

Figure 4 shows the flowchart that outlines the complete mechanization of the TD learning algorithm within an agent.

The negotiation process starts with each agent choosing the highest desire initially and adjusting the desire levels based on the learning as time progresses. The values of K1 and K2 were chosen based on the maximum distance typically represented on STARS charts.

C. Measures for adaptibility

Our NetLogo Implementation simulates the AFPS wherein every agent is invoked in a simulation cycle. The GUI (Graphical User Interface) implementation provides a method to set the 'Availability' of the waypoints for each constraint. This 'Availability' is dynamically changed by random means, based on a slider setting that conceptually maps to the severity of the constraint. By this method we can make dynamic changes in the environment related to weather, traffic and terrain. The GUI implementation also provides an option to set own aircraft speeds in the terminal area. Yet another option provides the ability to switch on/off a random selection of the terminal chosen for path navigation (Intention selection) within the final negotiated list. We have experimented with three implementation variations: one without the use of any form of learning, another using the temporal difference learning; In yet another implementation of the system, we have added another algorithm that provides additional reinforcement learning that uses the experience gained from episodes. An episode is one complete run of the system from start to a terminal end point. The method involves remembering (via writing into a file) the shortest navigable path to the terminal waypoints after arriving at a node. This 'desire.txt' file is updated after every episode to update any of the lists (navigable strings) that need to be changed due to discovery of new shorter paths. We have performed multiple runs with our NetLogo Implementation of the agent based AFPS with these different implementations. Figure 5 below is a snapshot of the desire changes in a single episode with TD learning and the associated code snippet showing Opf-2 implementation (Wx Agent).



```
foreach WX_Desire-list-L[
    if(item i WX_Desire-list-L = feasible-terminal)[
      set Lowest-Desire-Index i
    ]
    set i i + 1
  ]

if(Highest-Desire-Index <= Moderate-Desire-Index) and (Highest-Desire-Index <= Lowest-Desire-Index)[
      set WX_Highest-Desire-Value get-reward-value WX_Desire-list-H
    ]
if(Moderate-Desire-Index <= Highest-Desire-Index) and (Moderate-Desire-Index <= Lowest-Desire-Index)[
      set WX_Moderate-Desire-Value get-reward-value WX_Desire-list-M
    ]
if(Lowest-Desire-Index <= Highest-Desire-Index) and (Lowest-Desire-Index <= Moderate-Desire-Index)[
      set WX_Lowest-Desire-Value get-reward-value WX_Desire-list-L
    ]
```

Fig. 5. The NetLogo Implementation

In order to answer questions that we posed in Section II A, we analyzed the experimental data which recorded the aircraft speed, weather condition, terrain condition, ATC clearance, the values of K1, K2, α, and γ. The important output measures of Time Ticks of simulation (leading to time of flight), Distance travelled by the aircraft was also recorded. The following measures are proposed to answer the question related to benefit (performance) in the context of usage of learning within the system:

Measure of Cooperative Gain 1 = Time to complete an episode without Learning / Time to complete an episode with Learning (7)

Measure of Cooperative Gain 2 = Distance travelled to complete an episode without Learning / Distance travelled to complete an episode with Learning (8)

Measures in equations (7) and (8) provide a simple way to quantitatively answer questions on performance and efficiency of a learning algorithm (denominators in equations (7) and (8) will have different values for different learning algorithms).

In order to analyze how the system is adapting to changes, the following measures are proposed

Coeff. of Cooperation for an Agent (for an episode) =

[(Highest Desire Level * Time ticks) - (Instantaneous Desire Level * Time ticks)] / (Highest Desire Level * time ticks) (9)

Relative Coeff. of Cooperation (between agents M,N for an episode) =

Coeff of co-operation for an Agent M / Coeff. of co-operation for an Agent N, $M \neq N$ (10)

The 'Desire Level's are numerically provided values of 1, 2, and 3 corresponding to the lowest, moderate and highest desires that we have described in Sections III B and V B. Measures in equations (9) and (10) are examined in the context of varying α, γ, and most importantly, availability. Equation (9) provides a measure for the amount of adjustment that an agent had to accommodate in its desire level. This can be easily understood from Figure 6. The diagram shows how an agent that initially operates at the highest level, lowers its desire level to operate at the moderate desire level t1, holds the desire level until t3 whereupon it lowers it further down and at t4, it raises to the highest desire level. Considering time until t4, we see that the co-operation offered by this agent is directly proportional to the area under the instantaneous desire level. Therefore, over a sum period of time, this can be expressed as the ratio of the area under the instantaneous desire level to that under the idealized highest desire level

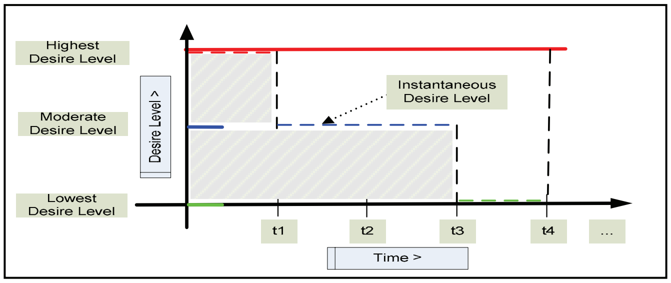


Fig. 6. Coefficient of co-operation

Having obtained a measure for an individual agent, it is cogent to propose a measure (equation (10)) that compares co-operation between a pair of agents. This provides a quantitative measure of the amount of mutual co-operation that exists relative to the disturbances in the environment for an episode and helps compare the learning algorithms. In our initial Monte Carlo type experiments for 67 episodes, we captured experimental data for which K1 and K2 were set to 100, α, and γ were each set to 0.9. The availability of waypoints in the STARS chart (Figure 1) for each constraint (weather, traffic, and terrain) was varied from 100% (always available) to 55 % (half available). Our NeLogo implementation also has built in the functions to record the number of time ticks, and compute the distance travelled, and the individual coefficients of co-operation for each episode. The results were populated into a spreadsheet and the relative coefficients of cooperation for each pair of agents was computed and tabulated. Table I shows the summary of our experimental results.

TABLE I. EXPERIMENTAL RESULTS FOR PROPOSED MEASURES

Measures for Adaptability		
Measure	***Mean Value***	***Standard Deviation***
Coeff. of Cooperation – TD Learning	0.294 (Wx Ag) 0.287 (Tr Ag) 0.278 (Te Ag)	0.141 (Wx Ag) 0.139 (Tr Ag) 0.135 (Te Ag)
Coeff. of Cooperation – TD Learning + Add RL Learning	0.282 (Wx Ag) 0.281 (Tr Ag) 0.265 (Te Ag)	0.118 (Wx Ag) 0.124 (Tr Ag) 0.124 (Te Ag)
Relative Coeff. of Cooperation – TD, ADD RL	1.097,1.059(Wx/Tr) 1.121,1.36 (Wx/Te) 1.13,1.393(Tr/Te)	0.51,0.33(Wx/Tr) 0.50,1.73(Wx/Te) 0.59, 2.06(Tr/Te)
Cooperative Gain 1 & 2 - TD	2.99 (G1) 2.99 (G2)	7.16 (G1) 7.16 (G2)
Cooperative Gain 1 & 2 – TD + Add RL Learning	4.42 (G1) 4.42 (G2)	13.579 (G1) 13.579 (G2)

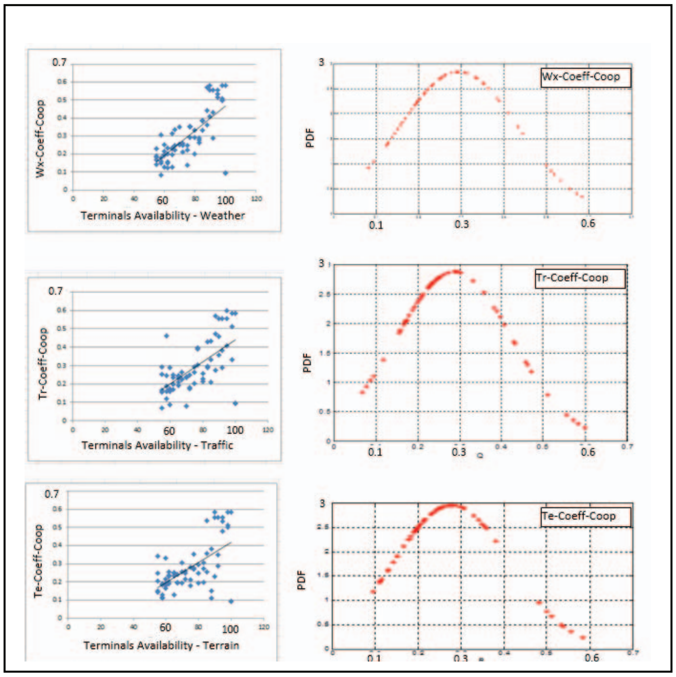


Fig. 7. Coefficient of co-operation plots

The left hand side of Figure 7 shows the plots of coefficient of co-operation (Y-axis) against Availability (X-axis) of terminals (constraints of weather, traffic, and terrain). Interpreting these plots, we arrive at the conclusion that we can obtain about 30% co-operation from every agent on the average. Maximum co-operation of about 60% is when the availability of terminals is high (85-100%) i.e, when the terminals are not heavily constrained. Minimum co-operation of about 10% is obtained when the availability of terminals is low (55%) i.e, when the terminals are heavily constrained. The right hand side of Figure 7 shows that the Coefficient of co-operation follows a normal distribution and the specific values are tabulated in Table 1.

Figure 8 provides a plot of time taken to reach the final waypoint versus the test case number. We progressively decreased the terminal availability with increasing test case numbers. The plot with highest peak indicates the case where no learning was employed and the other plot indicates the one where TD learning was employed. The initial test cases show conditions where availability was high. The terminal availability drops to 90% at test case 13, to 75 % around test case 25, to 67% around test case 40, and to 60% around test case 40, and finally to 55% around test case 67. We see that as the terminal availability decreases, the time taken to reach the final waypoint shows significant decrease for TD learning.

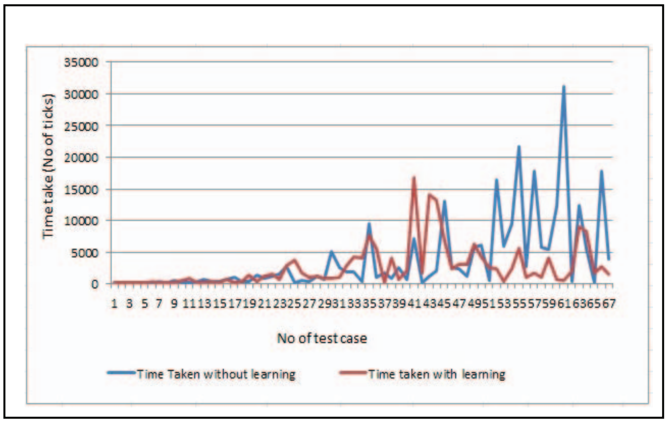


Fig. 8. Co-operative Gain plots

We see from the recorded experimental data from Table I that gain of 3 is obtained for the TD learning case while a gain of 4 is obtained for the case where additional reinforcement learning was added. Similar results were obtained when the distance travelled to reach the final waypoint versus the test case number was plotted.

VI. RELATED WORK

Work in the area of adaptive flight path/trajectory planning has been continuously evolving. We present and contrast our work with other important work in this area. AGENTFLY is a simulation platform that can simulate UAVs (unmanned Aerial Vehicles) and uses the free flight concepts [21]. The base is a modified A* algorithm. Path Planning is done in two phases: spatial and time planning. Our work differs in the fact that we integrate both of them in a single cycle. [22] is a recent publication on the use of adaptive path planning for UAVs using bi-level planning in place of real-time adaptive path planning and variable planning step techniques to build paths or new reference waypoints only when necessary. Our work differs from the fact that we do not anticipate the kind of performance variations that take into account the ownship performance variations; Also objectives are setup to adapt the flight path using a hierarchical principal-subordinate relationship – the leader's objective is to converge by providing a holistic trajectory while the follower optimizes locally – we can see a similar pattern in our work. [23] is a fairly recent treatise on generation of safe and reliable trajectories using hierarchical motion planning, in which trajectory generation and optimization is decomposed into two layers: geometric layer and dynamics layer. Our work, though focused on both these aspects, does not separate them, but incorporates both concepts within a single agent. Specifically, multi-resolution motion planning using wavelet-based cell decompositions has been shown to be efficient. The work represents the environment as a graph – we use a list. The trajectory generation uses an H-Costs algorithm with a discrete path planner that lifts partial graphs to satisfy constraints. Our arrangement of individual agents with the negotiation agent provides a similar effect in generating the total solution. [24] presents an adaptive trajectory planner for emergency landing scenario, using planning and reactive agents. The adaptive trajectory planning is built into the planning agent using three agents 'Pilot', 'Hybrid' , and 'FMS Agent' which use rule based approaches. We employ a more loosely coupled and generic approach in contrast. The use of RL in safety critical systems is provided in [25].

VII. CONCLUSION AND FUTURE WORK

Our work in formulating and simulating the Adaptive Flight Planning System with agents, has demonstrated how "Learning" could be introduced for an avionics domain application. During the initial stages we captured specifications particularly related to adaptability and learning for the system with intent to provide later the algorithms and methods to ensure safety, timeliness and deterministic aspects of the domain. Therefore ensures a means to articulate the system better for design transformation for the learning method. The choice of using Reinforcement Learning (TD) with reasoning and experimental results is provided. This work clearly shows how learning provides significant benefits (cooperative gains of 3 to 4.5 times). Also measures to characterize and analyze the system are proposed along with experimental data. We are carrying out further research towards formulating similar frameworks and their verification with aspects of safety, determinism, and realtimeliness in focus.

REFERENCES

[1] Ashley Aitken, Vishnu Ilango, "A Comparative Analysis of Traditional Software Engineering and Agile Software Development". 46th Hawaii International Conference on System Sciences, 2013, IEEE Computer Society, DOI 10.1109/HICSS.2013.31.

[2] Richard Stocker, Neha Rungta, Eric Mercer, Franco Raimondi, Jon Holbrook, "An Approach to Quantify Workload in a System of Agents", Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey.

[3] Mahdi Bashari and Ebrahim Bagheri, "Engineering self-adaptive systems and dynamic software product line", SPLC '13 Proc. 17th International Software Product Line Conference, Tokyo, Japan, 2013.

[4] Mazeiar Salehie, Ladan Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges". ACM Transactions on Autonomous and Adaptive Systems, Vol. V, No. N, March 2009.

[5] Yuriy Brun. et al., "Engineering Self-Adaptive Systems through Feedback Loops, Self-Adaptive Systems". LNCS. 5525 (2009), pp. 48–70 , Springer-Verlag Berlin Heidelberg, 2009

[6] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents", In Agent technology, Pages 3-28, Springer-Verlag,USA, 1998.

[7] Anand S. Rao and Michael P. George, "BDI agents: From theory to practice". In Proceedings of the First International Conference on Multiagent Systems (San Francisco, USA, June 12-14, 1995.

[8] Alejandro Guerra-Hernandez, Amal El Fallah-Seghrouchni, and Henry Soldano, "Learning in BDI Multi-agent Systems". Proceedings of CLIMA 2003. Springer Verlag, 2004.

[9] Dhirendra Singh, Sebastian Sardina, Lin Padgham, and Geoff James. "Integrating learning into a BDI age.nt for environments with changing dynamics", IJCAI 2011,p 2525-2530

[10] Dhirendra Singh, Sebastian Sardina, and Lin Padgham, "Extending BDI plan selection to incorporate learning from experience". Journal of Robotics and Autonomous Systems, 2010.

[11] Bhattacharyya, S., Cofer, D., Musliner, D., Mueller, J., Engstrom, "Certification considerations for adaptive systems". NASA Report, NASA/CR–2015-218702, Langley Research Center, USA. March 2015.

[12] Gratch, J., DeJong, G. "A Statistical Approach to Adaptive Problem Solving", Artificial Intelligence, Vol. 88, Iss. 1-2, Decr 1996.p. 101-142.

[13] RNAV-I (GNSS/DME/DME/IRU) SIDs and STARs, Chennai International, India, website: www.aai.aero/public_notices/29-2009.pdf

[14] Mark d'Inverno, Michael Luck, Michael P. George, David Kinny, and Michael Wooldridge. 2004. The dMARS architecture: A specification of the distributed multi-agent reasoning system In AAMAS, 2004.

[15] NetLogo multi-agent programmable modeling environment, https://ccl.northwestern.edu/netlogo/.

[16] Uri Wilensky and William Rand. "An Introduction to agent-based modeling: Modeling natural, social and engineered complex systems with Netlogo". MIT Press, 2015.

[17] K.-L. Du and M. N. S. Swamy. 2014. "Fundamentals of Machine Learning", Chapter 2, In Neural Networks and Statistical Learning. Springer-Verlag London, 2014, DOI: 10.1007/978-1-4471-5571-3_2.

[18] Richard S. Sutton and Andrew G. Barto. 2005. "Temporal-Difference Learning", Chapter 6, In Reinforcement Learning: An Introduction. MIT Press, 2005.

[19] Ferenc Beleznay, Tamas Grobler, Csaba Szepesvari, "Comparing Value-Function Estimation Algorithms in Undiscounted Problems", Technical Report TR-99-02, MindMaker Ltd, 1999.

[20] Rajanikanth N Kashi, Meenakshi D'Souza, S Kumar Baghel, Nitin Kulkarni, "Formal verification of avionics self adaptive software: A case study", ACM India Software Engineering Conference 2016, Goa, India.

[21] David Sislak, Premysl Volf, Michal Pechoucek, "Flight Trajectory Path Planning", http://www.rtca.org/

[22] Wei Liu, Zheng Zhenga, Kaiyuan Caia, "Adaptive path planning for unmanned aerial vehicles based on bi-level programming and variable planning time interval", Chinese Journal of Aeronautics, 2013.

[23] Panagiotis Tsiotras, Eric Johnson, "Advanced Methods For Intelligent Flight Guidance and Planning In Support Of Pilot Decision Making", Report, Intelligent Systems Division, NASA Ames Research, 2012.

[24] Igor Alonso-Portillo, Ella M. Atkins, "Adaptive Trajectory Planning for Flight Management Systems", AIAA Aero Sciences Conference, 2002

[25] Clement Gehring, Doina Precup, "Smart Exploration in Reinforcement Learning using Absolute Temporal Difference Errors", AAMAS 2013.