Modelling Team Adversarial Actions in UAV Operations

João Borges de Sousa
Departamento de Engenharia Electrotécnica e Computadores
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto
PORTUGAL

jtasso@fe.up.pt

Anouck Girard
Department of Aerospace Engineering
The University of Michigan at Ann Arbor
1320 Beal Ave, Ann Arbor, MI 48109
USA

anouck@umich.edu

ABSTRACT

The problems of modelling and control design for teams of adversarial forces in unmanned air vehicle (UAV) operations are discussed. A team is a group of cooperating entities which may evolve through different organizations, named configurations. To each configuration there corresponds a set of different properties for the entities in the team. The problem of configuration control is addressed by structuring the design in a hierarchical control structure where the properties of the system can be formally analyzed. The overall system is modelled in the framework of dynamic networks of hybrid automata. This is discussed in the context of the attack of a Blue force of unmanned air combat vehicles against a Red's ground force of SAM sites and radars.

1.0 INTRODUCTION

The problem of modelling the operations of opposing forces in a battlefield is challenging, for several reasons. One is the combinatorial nature of the problem. Other difficulties include the complexity of interactions (e.g. by task coupling and uncertainty), the dynamic nature of the situation, the fact that decisions must be made with partial, limited information, and the need for stochastic modelling.

Game theory uses mathematical models to model human decision making in competitive situations. It is ideally suited for analyzing military situations because it depicts the realistic situation in which both sides are free to choose their "best" moves and adjust their strategy over time (see [1,4] for example). This has been widely reported in the literature and several solution concepts have been introduced [1,3], as well as numerical methods for problem solving. Typically discrete-time stochastic games suffer from the "curse of dimensionality", as the cost of computing the player's expectations over all possible future states increases exponentially in the number of state variables. This is why techniques from game theory typically concentrate on stylized small scale problems, where complex interactions among the adversarial and friendly entities are not taken into consideration.

Borges de Sousa, J.; Girard, A. (2007) Modelling Team Adversarial Actions in UAV Operations. In *Platform Innovations and System Integration for Unmanned Air, Land and Sea Vehicles (AVT-SCI Joint Symposium)* (pp. 24-1 – 24-14). Meeting Proceedings RTO-MP-AVT-146, Paper 24. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

In order to cope with the challenges posed by real operations, manned operations are organized to make them more manageable to human decision-making, using hierarchy as an attack on complexity, thus restricting degrees of freedom for single units with the objective of getting control over ensembles. This does not solve the problem of dimensionality, but makes it more tractable and amenable to some form of organization. Moreover, it has led to a layered (and hierarchic) structure of decision-making. The hierarchic organization comes from several sources: 1) modular mix of capabilities into basic units (e.g. Platoons, regiments, etc…); 2) modular and nested composition of groups of units; 3) patterns of command and control in these nested structures; and 4) patterns of motion/actions for organizations.

The forms of organization for manned operations have evolved with time to face the challenges posed by new operational scenarios. However, the evolution of human organizations into new forms of organization and control has a steep learning curve. What happens with unmanned vehicle systems?

The organization and control of unmanned vehicle systems in adversarial scenarios represents a new modelling and control challenge. The interesting questions are: what is the most adequate form of organization for a particular pattern of adversarial behaviour? How can the form of an organization adapt to changes in the adversarial behaviour?

A significant body of recent research on the control of unmanned combat air vehicles in adversarial environments does not address these issues directly. However, a closer look at the operational environments tells us that their structure may provide guidelines for the design of automated organizations. In fact, these environments have some structure and we should be able to take advantage of this fact in order to propose forms of organization for combat air vehicles that are not only best adapted to each specific situation presented by the adversary, but that are also able to adapt the organization to changing situations.

Here we are concerned with the development of team strategies in the presence of team adversarial behaviour. This is the case, for example, of Suppression of Enemy Air Defence (SEAD) missions where one team attempts to eliminate the assets of the other team. By convention we will designate adversarial forces as "Red" and friendly forces as "Blue".

The developments in [14] discuss the design of planning and execution control framework for the attack of the Blue force of unmanned air combat vehicles (UAV), against Red’s ground force of SAM sites and radars. The design is structured in a two level hierarchy of planning and execution. The planning and execution control framework borrow much of their structure from the spatial organization of the problem and from the available capabilities in each vehicle. Each plan prescribes a set of constraints for execution. However, it does not prescribe a sequence of events and/or actions which would lead to poor execution performance in a non-deterministic world. This is why feedback strategies are used. There take the available information and the constraints from the plan, command UAVs to execute actions conforming to these constraints and leading to plan completion, while seeking to maximize some performance criteria. The concepts for execution control build on experience in the modular design of distributed control hierarchies described in [12], [13].

Here, we report on our investigations to the developments in [14] for problems where both forces present forms of organization, named configurations, which are intended to model the properties resulting from synergistic interactions in these forces. Configurations are a powerful way of modelling both the Blue and the Red forces. This is the case of an integrated air defence system where independently operated SAMs typically spend more missiles as a whole and are less effective in detecting Blue UAVs.

This leads to an abstract description of an adversarial game. The problem for each force in this adversarial game is to maximize its payoff. The state of the game is given in the product space of the configurations of the

two adversaries. The space of controls concerns changes of configuration. The control constraints are modelled as a graph where each node represents one configuration and the transitions model the admissible changes of configuration. There are payoffs associated to each transition. We are only concerned with feedback strategies from the state of the system to the control space at each state.

We will consider as a simple example for our developments that the Blue force is composed of three UAVs (jammer, bomber, sensor) and that the Red force is composed of a long range radar, 3 SAM sites with short range radars and a communication centre. When the 5 Red entities are connected to communication centre, the probabilities of detection increase and the firing strategy changes so that fewer missiles are fired (in comparison to the case of independent behaviour).

This paper is organized as follows. In Section 2 we review models of systems with evolving structure and whose properties depend on their current structure. In Section 3 we further discuss systems with evolving structure in the context of multi-vehicle control architectures. In Section 4, we introduce our modelling framework with the purpose of deriving control strategies for SEAD missions, and a simple problem to illustrate these concepts.

2.0 MODELS OF SYSTEMS WITH EVOLVING STRUCTURE

The control of distributed systems with evolving structure has presented a new challenge to control theory. This challenge entails a shift in the focus of control theory -- from prescribing and commanding the behaviour of isolated systems to prescribing and commanding the behaviour of interacting systems.

The nature of systems with evolving structure requires a new description language. The control and computer science communities address this challenge in the context of different applications, and contribute complementary views and techniques. Meanwhile, control engineers have developed a collection of idioms, patterns and styles of organization that serves as a shared, semantically rich, vocabulary among them [5]. However, this shared vocabulary is still deeply rooted in the underlying mathematical framework – differential equations and dynamic optimization – and lacks some semantically rich concepts invoked by distributed computing.

The notion of dynamic reconfiguration is an essential element for the control of the information flow, as one can infer from the following quotation from Robin Milner [15]:

> "Dynamic reconfiguration is a common feature of communicating systems. The notion of link, not as a fixed part of the system, but as a datum that we can manipulate, is essential for understanding such systems. Is there a common notion of link which subsumes pointers, references, channels, variables (in the programming sense), locations, names, addresses, access, rights, …, and is it possible to take this notion as basic in a computational model?... What is the mathematics of linkage?"

Here, we are concerned with models of systems with evolving structure. We are especially interested in models for distributed hybrid systems. The key idea is that links among components change during system execution. The constitution of the team of vehicles, along with their roles, may change over time due to faults, power saving strategies or due to requests to perform different kinds of missions. Therefore, the links should not be assumed as static.

Labelled transition systems allow for a simple representation of system behaviour. Furthermore, practical techniques have been developed for verification of finite-state automata. When transition systems model

infinite dimensional systems the problem of verification is typically undecidable. This is the case with general hybrid automata.

In [6], the hybrid automaton is specialized by augmenting it with the notion of input and output variables/actions (for the continuous and discrete case respectively). This framework defines the notion of external behavior, i.e. the discrete and continuous interactions between the automaton and the environment. It defines a composition operation and, for abstraction, notions of implementation and simulation using the concept of component interface, i.e., the externally visible behaviour of the component. The framework defines conditions to avoid the existence of Zeno-behaviour.

The theories of computation are evolving from notions like value, evaluation and function to those of link, interaction and process. π-calculus seems to be a first step in this direction. The π-calculus [7] treats linkage and mobility as basic, and takes interaction at a link as its only action. The computational world of π-calculus contains two kinds of entities: processes and channels. Processes (sometimes called agents) are the active components of a system; they interact by synchronous rendezvous on channels, also called names or ports. When two processes synchronize, they exchange a single data value, which is itself a channel.

In [8], the authors present the ***Dynamic I/O Automaton*** (DIOA) model in order to allow the definition and analysis of dynamic systems, in the sense that its components can be created and destroyed as computation proceeds. The authors claim some advantages over current methodologies for these kind of systems, namely the π-calculus [7]. The primitives used in the DIOA model are actions and automata while in the π-calculus the basic notions are channels and names. However, it is remarked that the approach is primarily a mathematical model, rather than a formal language. In the context of the DIOA model, automata which can create new automata are called ***signature automata*** (SIOA). The signature which characterizes each automaton is a function of the current state and indicates the current input, output and internal actions. It can change as state transitions are performed. In the limit, an automaton "self-destructs" (it cannot be destroyed by other automata) by changing its signature to an empty set. This is opposed to regular I/O automata where an action is always input, output or internal. The set of "existing" SIOA along with the current state of each one is tracked by a ***configuration automaton*** (which is itself a SIOA). The creation of new automata is modelled by a mapping from each state of the configuration automaton (termed "configuration") to the universal set of SIOA (i.e., all automata of the system model).

Notice that the signature cannot be used as an identifier of a particular instance of an automaton, i.e., it cannot be used to discriminate "clones" of the same automaton.

Another model that has some support for the dynamic creation and destruction of entities is the history dependent automata (HDA) [9] which evolved from an algorithmic structure for checking bi-similarity of π-calculus agents. The authors define HDA as automata which perform actions that can carry information generated in the past history of the system. The states, transitions and labels of the HDA are enriched with sets of local names. Thus, each transition can refer to the names associated to its source state but can also generate new names, which can then appear in the destination state. In this manner, the names are not global and static, as in ordinary labelled transition systems, but they are explicitly represented within states and transitions and can be dynamically created. The formalism provides the notion of bi-similarity and automata minimization. The authors describe translations of the π-calculus and Petri-nets to this formalism. In [10], extending the HD-automata as supporting model, the authors define a logic for dynamic creation and destruction of entities denominated **allocation temporal logic**, with the standard LTL operators and features such as assignment of entities to variables and assertion of whether two variables refer to the same entity. The authors show that model-checking for this logic is decidable and present an algorithm for that purpose.

In [11], the authors define the concept of **dynamic networks of hybrid automata** (DNHA). DNHA describe systems consisting of components which can be created, interconnected and destroyed as the system evolves. Each component is a hybrid automaton. A hybrid automaton consists of control locations with edges between the control locations. The control locations are the vertices in a graph. A location is labeled with a controlled ordinary differential equations (or a differential inclusion), and every edge is labeled with a guard, and a jump and reset relation. The state of a Hybrid Automaton is a pair (l, x) where l is the control location and $x \in R^n$ is the continuous state. Informally, a DNHA is a collection of hybrid automata that interact through the exchange of data and messages. Obviously, interactions are mediated by means of communication. Hence, a model for dynamic interactions has to include a description of the mechanisms by which automata interact. The DNHA computation model admits a compact representation for dynamic scheduling and network reconfiguration. The 1-to-1 synchronization is sufficient to model a static network. However, when the network is dynamic – that is, nodes in the network may appear/disappear – then more sophisticated data structures are needed. DNHA provides such constructs based on first-order predicate logic. DNHA allow for interacting automata to create and destroy links among themselves and for the creation and destruction of automata.

DNHA are also quite suitable to express properties resulting from the composition of hybrid automata. This way we are able to express the fact that a system, in a given configuration, possesses some properties. With DIOA we could also model this feature of composed systems by defining functions which map configurations to properties.

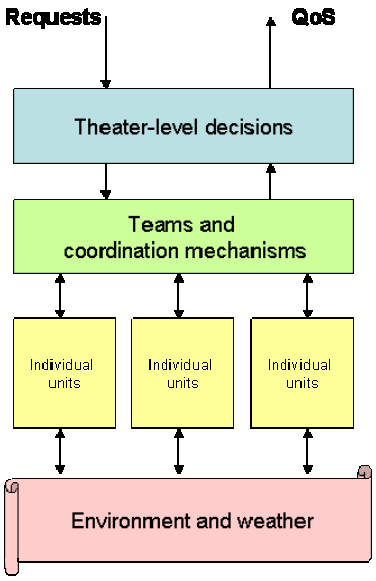
3.0 ORGANIZATION AND PATTERNS OF COORDINATION

Our approach here deals with the modelling of adversarial actions in UAV operations. As such, we are not primarily interested in the details of single vehicle operation; rather, we are interested in the organization and behaviour of a (possible large and heterogeneous) network of assets, and its ability to perform tasks over a region and time of interest [13].

The standard approach to control architectures for unmanned vehicles has been inspired from the parallel structures for manned operations [12]. Similarly, we use manned operations as a template for the design of organization patterns for the control of networks of assets.

There are many difficulties associated with such a problem, including ***complexity*** (for example, as generated by large design spaces, and/or task couplings), incomplete or delayed information, and a lack of understanding of the effects of scale. We start with the consideration of methods for alleviating complexity. The most common approach is “divide and conquer”, that is, to organize the functions into ***hierarchical layers***. This way, a complex design problem is partitioned into a number of more manageable sub-problems that are addressed in separate layers. The decomposition also allows for modular design and testing and the incorporation of plug-and-play components. A vast majority of current multi-vehicle systems and experimental demonstration setups are organized into hierarchical control architectures [12]. Tractability is obtained at the cost of global optimality.

Similarly, we organize the problem of team adversarial actions in a hierarchy. At each level in the hierarchy, we model components by using DNHA, as described in section 2.0.



Team and adversarial actions modelling and organization.

Individual units (by which we mean a "building block" for military operations, that is, an independently operating element) are modelled as DNHA. This allows model instances to be created or removed "on-the-fly" as operations progress. Individual units are subject to constraints of the environment and weather (which impact mobility, and therefore effectiveness).

Teams and coordination mechanisms are also modelled through DNHA. Teams involve two or more individual units. The team structure also evolves, as teams may be formed or dissolved as operations progress. For large models, more than one "team and coordination mechanism" level might be necessary, without loss of generality. Theatre-level decisions are modelled through DNHA; the input to this level is user requests, and the user is given feedback on the obtained quality of service – that is, the number of requests fulfilled, and the time taken to satisfy requests [13]. Here "theatre" is used to denote a specific geographic region within which military operations occur.

4.0 MODELS

4.1 Concepts

Our model includes both isolated entities and groups of entities operating as a team. We adopt some definitions from [14]. *Target* and *UAV* are generic terms for Red force and Blue force entities respectively. There are two finite set of types of targets and UAVs, called respectively *TargetTypes* (e.g. SAM launchers, command centres and radars) and *UAVTypes* (e.g. jammer, bomber and sensor). The type describes the properties presented by each entity when operated in isolation and how the parameters of these properties change when the entity is operated in conjunction with other entities (when operating in a team).

A prototype of an action/motion description for a *Target* or for a *UAV* is called a manoeuvre. The manoeuvre is the atomic component of all execution concepts. Each entity may also provide services to other entities. Each entity is abstracted as a provider of manoeuvres and services.

This is a first step towards tackling automated and adaptable organization in a systematic fashion. But, there is more. First, entities in both forces are heterogeneous and have complementary capabilities. Second, the correct use of capabilities contributed by different entities introduces sequence constraints which can be abstracted by patterns of execution (e.g. sense, attack, sense). Third, the structure of the world also leads to some constraints for the evolution of the Blue force (open corridors, etc). This means that we can organize groups of UAVs into teams and that there are patterns of motion/action (team manoeuvres) that can be used for each specific situation. This leads to a considerable reduction of the dimension of the decision-space and to a high level description of execution control. At this level we are able to formulate the problem facing the Blue force in the framework of dynamic optimization. The decisions concern team manoeuvres, which are then decomposed into single vehicle manoeuvres in the control hierarchy.

Targets and *UAVs* can operate in isolation or integrated in a team. This is modelled with local controllers and with team controllers in the framework of dynamic networks of hybrid automata. There is one local controller for each entity. This controller is termed the supervisor of the entity. The team controller has control links to the local controllers of the constituent entities. The team controller deals with configurations.

4.2 Teams

A *Target* is characterized by its type and its (two-dimensional) location (x,y). A Red force with N targets is thus described by a set of the form

$Targets = \{target_1 = (\ type_1, (x_1, y_1)), \ldots, target_N = (\ type_N, (x_N, y_N))\}$

Likewise the Blue force with M *UAV*s is described by a set of the form

$UAVs = \{uav_1 = (\ type_1, (x_1, y_1)), \ldots, uav_M = (\ type_M, (x_M, y_M))\}$

There is a local controller, the supervisor, for each *UAV,* and one for each *Target*. The supervisor T_V is modelled as a hybrid automaton:

$T_V = (Q_V, \rightarrow, I_V, O_V, V_V, Init_V, Final_V)$

Q_V = {Idle, Team, Isolated, Error} – the discrete states

I_V = {list-of-manoeuvres} – the input events (commands for execution)

O_V = {list-of-messages} – the output events (state messages)

V_V = {variables(discrete-state)} – the valuations of internal variables (which depend on the type of entities the UAV or Target interacts with)

$Init_V$ = {Idle} – the initial state

$Final_V$ = {Stop} – the final state

$\rightarrow$ – the transition relation (that encodes the control logic).

The interactions with the team controller are modelled by using the input and output events. A simple protocol governs the interactions between the supervisor and the team controller: the team controller sends a

manoeuvre command to the supervisor; the supervisor either accepts the command and executes the manoeuvre, or it does not accept the command and sends an error message to the team controller; the supervisor sends a 'done' message or an 'error' message to the controller depending on whether the manoeuvre terminates successfully or fails.

The list of variables for a *Target* includes, for example, *probability of detecting a UAV, number of weapons* and *effectiveness of weapons*. The list of variables for an *UAV* includes *current manoeuvre*, *probability of being detected* and *effectiveness of weapons*. We observe that valuations for these variables depend on the discrete state.

The notion of configuration is introduced at the team level. The simplest way of thinking of a configuration is to consider the synchronous composition of the vehicle supervisors. A configuration is basically one or more states of the product automaton, together with valuations of a set of the variables in this automaton. For example, a team composed of one jammer and one bomber UAV is in an attack configuration when their supervisors are both in the team state and the jammer is executing a jamming manoeuvre to protect the bomber. However, it is not practical to work with this product automaton, since we are only interested in a small number of configurations. This is why we introduce the notion of team controller, the states of which include the configurations of interest.

In general, there is one team controller for each team in each force. For our application of interest, and for the sake of clarity, we consider only one team controller for each force, as we are assuming relatively small numbers of UAVs and targets for SEAD. In general, each team controller is associated with an entity (or unit) in the force. This means that the elimination of the team controller results in the elimination of the team 9or vice-versa). The models for the Red and Blue team controllers T_{CR} and T_{CB} are hybrid automata with a similar structure T_C:

$T_C = (Q_C, \rightarrow, I_C, O_C, V_C, Init_C, Final_C)$

Q_V = {Idle, $Conf_1$, …, $Conf_C$, Error} – the discrete states (*$Conf_1$, …, $Conf_C$*. are configurations)

I_V = {list-of-commands} – the input events (commands for execution)

O_V = {list-of-team-commands} – the output events (commands for the vehicle supervisors)

V_V = {variables(discrete-state)} – the valuations of internal variables (which depend on the type of entities the team controller interacts with)

$Init_V$ = {Idle} – the initial state

$Final_V$ = {Stop} – the final state.

$\rightarrow$ – the transition relation (that encodes the control logic).

T_{CB} (T_{CR}) encodes part of the Blue (Red) force control strategy: configurations (forms of organization) and control for each configuration (what can be done in each configuration). There are two types of input events (commands) for the team controller: switches among configurations (discrete states) and switches among team manoeuvres in the same configuration. For example, in the *attack* configuration of the *jammer+bomber* team there is just one team manoeuvre, *attackjam(target)*. The role of the team controller is to send a list of team

commands (one for each UAV) to the corresponding supervisors (i.e. to 'translate' high level commands). A *jammer+bomber+sensor* team could perform one additional manoeuvre: *attackjamconfirm* (where the sensor UAV is tasked with the assessment of the state of the target after the engagement).

The model for each force results from the synchronous composition of the team controller with the vehicle supervisors. For example, the model of the Blue force is

$F_B = T_{CB} \parallel T_{V1} \parallel \ldots \parallel T_{VM}$

The elimination of the team controller leaves each entity in the *Isolated* discrete mode and no coordination is possible.

The modelling observation here is that some of the UAVs may be rendered inoperative. This means that we would need to consider the transition of the automaton resulting from the above composition to another, where one or more vehicle supervisors are missing. The formalism of configuration automata can be used to alleviate this.

4.3 Control formulation and SEAD example

In more abstract terms what we have is two team controllers playing against each other. In what follows, and for the sake of simplicity, we consider that the team controller for the Red Force is just a command centre which provides targeting information to each SAM site and coordinates the launch of SAMs by the launchers. This results in higher lethality for each SAM and minimizes the number of SAMs launched. The launch strategy is encoded as a team manoeuvre.

The Blue force is composed of UAVs (jammer, bomber, sensor) and the Red force is composed of a long range radar, SAM sites with short range radars and a communication centre. The Red force team controller integrates the radar information so that the detection probabilities of UAVs increase. Moreover, there is an integrated firing strategy for the SAM sites which results in having fewer missiles fired (in comparison to the case of independent behaviour). If the long range radar is rendered inoperative the probabilities of detection for the team are decreased. If the command centre is rendered inoperative the probabilities of detection for each SAM site are further reduced and they change to the isolated firing mode (where more SAMs are fired). The number of SAM available to each SAM site is limited.

The objective of the Blue Force is to render the 3 SAM sites, which are located in a geographic configuration to protect both the command centre and the long range radar (which are located inside the triangle), inoperative.

We take two different approaches to the control formulation. One is basically an extension of the planning and execution techniques proposed in [14]. This approach finds the optimal sequence of secondary targets to engage in the process of opening a path to the primary targets. Optimal here is defined in the sense of minimizing the maximum risk incurred by each UAV. The planning procedure is based on the properties of UAVs operating in isolation and also on the assumption that the state of the target after engagement was immediately available to the controllers. The extension of this procedure considers the properties of the team engaged in the operations. These include not only the lower risk incurred by the UAVs but also the capability to perform Battle Damage Assessment (BDA) which enable us to relax the second assumption. The execution control for a given plan is based on sequencing team manoeuvres in the light of was what done with UAV manoeuvres.

The second approach to the control formulation results from the application of dynamic programming to our problem. This is an extension of what was proposed in [16]. The problem is to go through a sequence of targets.

State Equation

The state is formed the number of vehicles in the Blue team at time step k, N_k, the fuel reserve available to each vehicle, $r_{1,k},\ldots, r_{N,k}$, the effectiveness of the enemy E_k (which can take values between zero or one, depending on whether the Red team is acting as a completely connected team (1), in an isolated fashion (0), or some configuration in between.

The state update is:

- Fuel reserves get reduced proportionally to the distance to the next target $d_{i,k}$, where k is the time step and i is the vehicle number.

$$r_{i,k+1} = r_{i,k} - d_{i,k}$$

- The number of vehicles might be reduced because of enemy fire, which causes a number of casualties c_k at step k.

$$N_{k+1} = N_k - c_k$$

The probability of being shot down depends on

- The distance travelled between targets (normalized by D)
- The effectiveness of the Red force (which changes in time, depending on what targets have been destroyed)
- A random variable, w_k, that models uncertainties in the detection and targeting of the Blue force by the Red force, caused by imperfect sensors and weapons, weather conditions, etc…

$$c_k = Ew_k \sum_{i=1}^{N} \frac{d_{i,k}}{D}$$

State constraints:

$$0 \le r_k, \quad 0 < N_k$$

Fuel reserves and numbers of available vehicles must always be positive.

Control constraints

The decision variables include whether to release a bomb or not, and whether to perform BDA using the sensor or not. This can be captured by one variable, u, which belongs to a discrete set with four

components. In this paper, for this preliminary presentation, we assume bombing and BDA are done in the same step as a first cut.

$$u_k \in \{0,1\}$$

If $u_k = 0$, do not release bomb, do not perform BDA at step k, if $u_k = 1$, release bomb, use sensor to perform BDA.

Expected cost function:

$$V(r_1, N_1, E_1 \mid w_1) = E\left[\sum_{i=1}^{N_1} g(w_1)u_i\right]$$

The cost function concerns the benefits g related to bombing and classifying each target. The cost function is an additive function representing the sum of the expected conditional reward (in terms of "number of targets processed") at each step. The expectation is taken over the random variable w_k.

The optimal cost function would be:

$$V^*(r_1, N_1, E_1 \mid w_1) = \max_u V_u(r_1, N_1, E_1 \mid w_1)$$

Stochastic Dynamic Programming (SDP)

Applying the SDP algorithm, we first need to compute $V_N(r_N)$:

$$V_N(r_N, N_N, E_N \mid w_N) = \begin{cases} \bar{g}(w_N) & if\ r_N \geq d_{i,N} \quad and \quad N_N > 0 \\ 0 & otherwise \end{cases}$$

Substitute the above equations and solve for the random variable:

$$w_N^* = \frac{N_N}{E_N \sum d_{i,N} / D}$$

Then by inspection:

$$u_N^* = \begin{cases} 1 & if\ w_N < w_N^* \\ 0 & otherwise \end{cases}$$

One can then apply the SDP recursion and obtain the cost function and optimal decision for all time steps.

$$V_k(r_k, N_k, E_k \mid w_k) = \max\left\{\bar{g}(w_k) + V_{k+1}(r_{k+1}, N_{k+1}, E_{k+1}), V_{k+1}(r_k)\right\}$$

5.0 CONCLUSIONS

We consider modelling formalisms for operation of opposing forces in the battlefield. This is a difficult problem, due to the combinatorial nature of the problem, the complexity of interactions (e.g. by task coupling and uncertainty), the dynamic nature of the situation, the fact that decisions must be made with partial, limited information, and the need for stochastic modelling. Interesting questions are: what is the most adequate form of organization for a particular pattern of adversarial behaviour? How can the form of an organization adapt to changes in the adversarial behaviour?

We have reviewed models of systems with evolving structure, whose properties depend on their current structure. We have discussed systems with evolving structure in the context of multi-vehicle control architectures, and we introduced a modelling framework with the purpose of deriving control strategies for SEAD missions, and a simple problem to illustrate these concepts. Future work includes refinements to the modelling and control framework and simulation.

ACKNOWLEDGEMENTS

João Borges de Sousa gratefully acknowledges the support given by the AsasF project from Porto University and by Fundação Luso-Americana para o Desenvolvimento.

REFERENCES

[1] Cruz, J.B., Simaan, M.A., Gacic, A. Jiang, H., Letellier, B. and Li, M. “Modelling and Control of Military Operations Against Adversarial Control”. Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia, December 2000, pp. 2581-2586.

[2] Attie, P. C. and Lynch, N. A., “Dynamic input/output automata: a formal model for dynamic systems,” MIT Laboratory for Computer Science, Cambridge, MA, 02139, Tech. Rep. MIT-LCS-TR-902, July 2003.

[3] Cruz, J.B., Simaan, M.A., Gacic, A. and Liu, Y. “Moving Horizon Game Theoretic Approaches for Control Strategies in a Military Operation”. Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, FL, December 2001, pp. 628-633.

[4] Cruz, J.B. and Simaan, M.A. “Ordinal Games and Generalized Nash and Stackelberg Solutions”. Journal of Optimization Theory and Applications, Vol. 107, No. 2, pp. 205-222, November 2000.

[5] Shaw, M. and Garlan, D. Formulations and Formalisms in Software Architecture, Computer Science Today: Recent Trends and Developments, Editor: J. van Leeuwen, Springer-Verlag, 1996, pp 307-323.

[6] Lynch, N., Segala, R. and F. Vaandrager, “Hybrid I/O Automata Revisited”, Lecture Notes in Computer Science, Volume 2034, 2001.

[7] Milner, R. Communicating and Mobile Systems: The π-Calculus. Cambridge University Press, 1999

[8] Attie, P.C. and Lynch, N.A. Dynamic Input/Output Automata: A Formal Model for Dynamic Systems. MIT Technical Report, Laboratory for Computer Science, MIT-LCS-TR-902, Cambridge, MA 02139, July 2003.

[9] Montanari, U. and M. Pistore. History-Dependent Automata. Technical Report, Dipartimento di Informatica University of Pisa, TR-98-11, October 1998.

[10] Distefano, S., Rensink, A. and J.P. Katoen, “Model Checking Birth and Death”, in Proceedings of the 2nd IFIP World Conference on Theoretical Computer Science (TCS 2002), Kluwer Academic Publishers, Montreal, Canada, 2002.

[11] Deshpande, A., Gollu, A. and L. Semenzato, “The SHIFT Programming Language for Dynamic Networks of Hybrid Automata”, IEEE Transactions on Automatic Control", Volume 43, Number 4, pages 584-587, April 1998.

[12] Girard, A., Sousa, J. and J.K. Hedrick, “A Hierarchical Control Architecture for the Mobile Offshore Base”, Marine Structures Journal, Vol. 13, No. 4-5, July 2000, pp.459-476.

[13] Sousa, J., Girard, A. and J.K. Hedrick, “Elemental Manoeuvres and Coordination Structures for Unmanned Air Vehicles”, Proceedings of the International Conference on Decision and Control, IEEE CDC 2004, Paradise Island, The Bahamas, December 2004.

[14] Borges de Sousa, J., Simsek, T. and Varaiya, P, “Task planning and execution for UAV teams”, Proceedings of the International Conference on Decision and Control, IEEE CDC 2004, Paradise Island, The Bahamas, December 2004.

[15] Milner, R., Semantic Ideas in Computing, in Computing Tomorrow: future research directions in computer science, Cambridge University Press, 1996, pp. 246—283.

[16] Girard, A., Dharba, S., Pachter, M. and Chandler, P., “Stochastic Dynamic Programming for Uncertainty Handling in UAV Operations”, Accepted for Publication, American Control Conference, ACC 2007.