HIERARCHICAL REINFORCEMENT LEARNING APPLIED TO AUTONOMOUS UNDERWATER VEHICLES

PEDRO LIMA, GEORGE SARIDIS

Electrical, Computer and Systems Engineering Department

Rensselaer Polytechnic Institute

Troy, NY 12180-3590

Abstract

A general methodology for performance improvement of Intelligent Machines based on Hierarchical Reinforcement Learning is introduced. Machine Decision Making and Learning are based on a cost function which balances reliability and computational cost of algorithms at the three levels of the hierarchy proposed by Saridis. Despite this particular framework, the methodology intends to be sufficiently general to encompass different types of architectures and applications.

Novel contributions of this work include the definition of a cost function combining *reliability* and *complexity*, recursively improved through feedback, a Hierarchical Reinforcement Learning and Decision Making algorithm which uses that cost function, and a coherent joint definition of algorithm cost and reliability.

Results of simulations show the application of the formalism to an Intelligent Robotic System mounted on an Autonomous Underwater Vehicle.

1 INTRODUCTION

Most of the work done in the last few years towards building Hierarchical and Goal Directed Intelligent Machines (HGDIM) [15] quite often mentions the need for a methodology of designing the IM and a measure of how successful the final result is. An analytic design based on measures of performance recursively improved through feedback, assures some degree of certainty about the measurability and robustness of that design. Previous results within the framework of the *Analytic Theory of Intelligent Machines* developed by Saridis *et al* [17] established a general architecture for the IM and detailed this architecture for the different levels. However, the flow of feedback through the hierarchy with the purpose of improving the overall performance by updating the decision making structure, has never been detailed for the complete hierarchy.

The present work proposes a methodology for performance improvement of Intelligent Machines based on *Hierarchical Reinforcement Learning*. Different options to accomplish a goal or a subgoal may be found at all levels of the IM: the **Organization Level** has to decide among different *tasks* capable of executing a given goal (*command*) sent to the machine; given the chosen task, composed by subgoals (*events*), the **Coordination Level** has to determine, for each event, the best among the set of *primitive algorithms* capable of solving each subgoal.

To compare the different alternatives at each level we further need a *cost function*. The proposed procedure recursively estimates a cost function combining *reliability* and *computational cost* of tasks, events and primitive algorithms. This approach has the advantage of providing a cost measure applicable to several different problems, since it is based on *reliability* (defined as the probability that an algorithm will meet some set of specifications in a given state of the environment) and *complexity* of a problem, i.e. minimum computational cost of the algorithm which solves the problem, here considered not only in terms of computation time (time-complexity) but also more general features such as memory and other resources usage (space-complexity).

When dealing with very large systems, some amount of uncertainty exists in the model of the system to be controlled. Hence there is always uncertainty about the result of a given command sent to the controlled system. *Uncertainty* is present at all levels of a HGDIM, as defined by Saridis: at the **execution level**, there is uncertainty in terms of features like rise-time or overshoot, since mathematical models never match exactly the real controlled system. At the **coordination level**, there is uncertainty in terms of the success of each of the *primitive events* (e. g. strut grabbed, path planned, manipulator did not move) composing a *task*. At the **organization level**, there is uncertainty in terms of the success of the *task* executed.

The different algorithms used at the Execution Level of an Intelligent Machine are frequently designed in order to meet a set of specifications or, without loss of generality, in order to keep the error of a set of involved variables below some desired *accuracy* $\epsilon \geq 0$. The uncertainty involved in the design of these algorithms is mostly due to approximate or incomplete modeling, and statistical fluctuations around nominal parameters. Hence it can be modeled statistically. Previous work in this area by McInroy and Saridis (1990)[11] and Musto and Saridis (1993)[12] describe a model-based approach where the most reliable from a set of different algorithms is selected by an entropy-based technique. However, the most reliable algorithm may have a non feasible computational cost, in terms of the time it takes to complete, the amount of memory it uses or the number of resources (e.g. processors) required. No attempt is made in those works to deal with this problem, with the exception of plan execution time, modeled as a specification by McInroy and Saridis (1990)[11]. Also, reliability is not learned from experience, but model-based.

Autonomous Underwater Vehicles (AUVs) are Unmanned and Untethered Mobile Robots capable of carrying out exploratory missions at sea. The diversity of subsystems composing AUVs (positioning, obstacle detection and recognition, control, path planning, communications, mission management) and the vehicles autonomy requirements make them particularly well suited for applications of Intelligent Control methodologies. The problem addressed by this particular instance of our methodology is: given a set of alternative tasks (sequences of calls to any subsystem but the Mission Management) capable of implementing commands coming from the Mission Management subsystem, and a set of alternative algorithms capable of solving the problems associated to the other subsystems, learn the best translations at each level to improve the overall performance, defined as a balanced sum of reliability and cost.

The paper is organized as follows: after this introduction, we briefly outlook in section 2 the joint definition of reliability and complexity of an algorithm. Section 3 explains the hierarchical decision making and learning algorithms, including the definition of the cost function. Afterwards, section 4 describes the application to a manipulator mounted on an Autonomous Underwater Vehicle, and section 5 presents preliminary conclusions and directions of future work.

2 COMBINED DEFINITION OF RELIABILITY AND COST

In order to coherently combine the definitions of cost and reliability for a given problem, the key is the desired accuracy or error specification ϵ for the problem, which must be the same in both definitions. This may be done using the background of the Theory of Information-Based Complexity[16]. To save space, we briefly summarize our formulation next. A more detailed and rigorous mathematical treatment may be found in [7].

We desire to compute a *solution approximation* $\underline{U}(f,\phi)$ of $\underline{S}(f)$, where $\underline{S}(f)$ is called a *problem solution.* $\underline{U}(f,\phi)$ is the result of an algorithm ϕ, capable of solving the problem, when operating over the data or *problem element* f. Typically, $\underline{S}(f)$ is a vector of specifications for a given problem, for example the desired joints position overshoot of control algorithms that can solve the move robot problem, and the problem element f is a vector with the output and set point signals used to compute the actual overshoot $\underline{U}(f,\phi)$ when algorithm ϕ is applied. To measure the distance between $\underline{S}(f)$ and $\underline{U}(f,\phi)$ we use an absolute error criterion, $\|\underline{S}(f)-\underline{U}(f,\phi)\|$, where $\|(.)\|$ represents some norm. $\underline{U}(f,\phi)$ is an *ϵ-approximation* of $\underline{S}(f)$ iff $\|\underline{S}(f)-\underline{U}(f,\phi)\| \leq \epsilon$. We call ϵ the *accuracy* of the approximation. In a probabilistic setting[16] the specification error is required to be below ϵ except in a subset with a small measure.

The *ϵ-cost* (*cost* for short) of a problem as defined in [9] as the lowest cost among the available algorithms capable of approximating the problem solution to some desired accuracy ϵ. The cost includes the prices of getting information and processing it. Depending on the model used, different features are weighted (CPU time, memory space, number of processors).

Given some desired accuracy ϵ, *Reliability* of algorithm ϕ is defined as:

$$R(f,\phi) = \Pr\{\|\underline{S}(f)-\underline{U}(f,\phi)\| < \epsilon\} \tag{1}$$

The *cost* of an algorithm ϕ given a problem element f has two components:

$$\text{cost}(\phi,f) = c_i(\underline{\mathcal{I}}(f),f) + c_p(\phi,\underline{\mathcal{I}}(f)) \tag{2}$$

where c_i is the cost of getting information about f needed by algorithm ϕ, and c_p is the combinatorial cost of processing that information by algorithm ϕ. The term c_i is inherent to information-based complexity. Information is gathered to reduce uncertainty. c_p would be the only term in the absence of uncertainty.

Given ϵ and δ, we define the cost of algorithm ϕ for the most unfavorable problem element f whose approximated solution $U(f)$ still belongs to the subset of G with measure $1-\delta$:

$$\begin{aligned} f^* &= \arg\inf_{f\in F}\{\Pr\{\|S(f)-U(f,\phi)\| < \epsilon\} \ni \Pr\{\|S(f)-U(f,\phi)\| < \epsilon\} \geq 1-\delta\} \\ \text{cost}(\phi) &= \text{cost}(\phi,f^*) \end{aligned} \tag{3}$$

or, taking into account (1) and making $R_d = 1 - \delta$, where R_d is some desired reliability:

$$\begin{aligned} f^* &= \arg \inf_{f \in F} \{R(\phi, f) \ni R(\phi, f) \geq R_d\} && (4) \\ C(\phi) &= \mathrm{cost}(\phi, f^*) && (5) \end{aligned}$$

that is, among all $f \in F$ capable of keeping the specification error for algorithm ϕ below ϵ with reliability at least R_d, the one leading to the worst-case, i.e. the f leading to the largest probability of error, is picked. Here and henceforth, the reliability will be denoted as $R(\phi) = R(\phi, f^*)$.

The link between the definitions of reliability and cost is the assumption that all algorithms are designed to meet an error specification ϵ of the problem they can solve. Given some desired reliability for the problem, the cost of obtaining that reliability can be determined for each of the algorithms, according to the cost measure defined (number of operations, elapsed CPU-time, memory used) for the problem. Conversely, if the cost measure is fixed at different values for the different algorithms, this will correspond to different reliabilities for each of them.

For example, N image frames or more need to be averaged to increase past a certain value R_d the probability that the error of locating an object in a noisy image is below ϵ. Every image resulting from the average of a different number frames is a problem element. If the cost of processing that information is not considered, the overall cost will be equal to c_i and proportional to the number of averaged frames. Among the number of image frames which have to be averaged, N corresponds to the worst-case specification error. A greater number of averages will decrease the error probability, while a smaller number will push the corresponding approximated problem solution to the subset of G with measure δ, for which $R(\phi) < R_d$.

Tasks implemented by Intelligent Robotic Systems may generally be decomposed on primitive events. Among these the most typical events are perhaps Move Robot, Locate Object, Plan Path, Grasp Object. Algorithms capable of solving these problems belong to the areas of Motion Control, Computer Vision, Trajectory Generation and Compliant Grasping. Next we will give a few examples on how the performance of some of these algorithms may be computed under the paradigm just formulated. Emphasis is put on cost measures other than execution or computing time, to enhance the flexibility of the definition.

2.1 Motion Control

The dynamics of a n-degree of freedom robot manipulator can be expressed by the following compact form of Euler-Lagrange's equations of motion:

$$D(\underline{\theta})\ddot{\underline{\theta}} + \underline{NL}(\underline{\theta}, \dot{\underline{\theta}}) = \underline{u} \qquad (6)$$

where $\underline{\theta} \in \Re^n$ is the joint angles vector, $\underline{u} \in \Re^n$ is the control torques vector, $D(\underline{\theta}) : \Re^n \rightarrow \Re^{n\mathrm{X}n}$ is the inertia matrix, and $\underline{NL}(\underline{\theta}, \dot{\underline{\theta}}) : \Re^n \mathrm{x} \Re^n \rightarrow \Re^n$ is the vector representing nonlinear coupling of Coriolis, centrifugal, gravity and friction torques. Luo and Saridis (1985)[10] formulated the optimal control solution for the problem of making the manipulator track a desired trajectory. They identified the system state with $\underline{x}(t) = (\underline{\theta}(t)\ \dot{\underline{\theta}}(t))^T$ and suggested the performance index

$$J(\underline{u}) = \frac{1}{2}\underline{e}^T(t_f)G\underline{e}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}[\underline{e}^T(t)Q\underline{e}(t) + \dot{\underline{e}}^T(t)S\dot{\underline{e}}(t)]dt \tag{7}$$

where $S = \begin{bmatrix} 0 & 0 \\ 0 & S_0 \end{bmatrix}$, G is a $2n$x$2n$ and S_0 a nxn real symmetric, positive definite matrix, Q is a real non-negative $2n$x$2n$ matrix, $\underline{e}(t) = \underline{x}_d(t) - \underline{x}(t)$ and $\underline{x}_d(t) = (\underline{\theta}_d(t)\dot{\underline{\theta}}_d(t))^T$ is the desired state vector. When $t_f \to \infty$, the control law reduces to

$$\underline{u}^* = D(\underline{\theta})\{\ddot{\underline{\theta}}_d(t) + K_p[\underline{\theta}_d(t) - \underline{\theta}(t)] + K_v[\dot{\underline{\theta}}_d(t) - \dot{\underline{\theta}}(t)]\} + \underline{NL}(\underline{\theta}, \dot{\underline{\theta}}) \tag{8}$$

which has the same form of the *Computed Torque Method*, with $K_p = S_0^{-1}P_{12}$ and $K_v = S_0^{-1}P_{22}$. $P = \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix}$ is the solution of a continuous algebraic Ricatti equation.

If we assume complete information about the state, but measurements are noisy and cancellation of non-linear terms is not perfect, and if we further model these uncertainties as additive gaussian noise, we will end up with a particular case of a Linear Quadratic Gaussian problem, where information about the state is complete. Given the optimal control law, the discretized closed loop model becomes:

$$\underline{x}((k+1)T_s) = A_{dcl}\underline{x}(kT_s) + B_{dcl}\underline{u}_d(kT_s) + D\underline{v}(kT_s) \tag{9}$$

where $\underline{v}$ is a gaussian noise vector with $E[\underline{v}(kT_s)] = 0$, $E[\underline{v}(kT_s)\underline{v}(kT_s)^T] = C_v$, $\underline{u}_d = (\underline{\theta}_d^T \dot{\underline{\theta}}_d^T \ddot{\underline{\theta}}_d^T)^T$ and T_s is the sampling period.

The performance index has to be modified when the noise is actually added to the open loop system, and it becomes $I(\underline{u}) = E[J(\underline{u})]$. For this motion control problem (event move robot) we identify the algorithms cost with the optimal value of I:

$$C = I(\underline{u}^*) = \underline{e}(0)^T P\underline{e}(0) + \sum_{k=1}^{N} \mathrm{tr}(PDC_vD^T) \tag{10}$$

where P is the solution of a discrete algebraic Ricatti equation (Lewis, 1986) [6], and N the number of samples in the trajectory.

A lower bound for the Reliability can be obtained based on a method described by McInroy and Saridis (1990) [11], when the specifications are quadratic in the tracking error $\underline{e}(kT_s)$:

$$\underline{e}(kT_s)^T Q_s\underline{e}(kT_s) \leq \epsilon,\ k = 1,\ldots,N,\ Q_s \geq 0 \tag{11}$$

If

$$C_e^{-1}(kT_s) - Q_s(kT_s) \geq 0, \forall k = 1,\ldots,N \tag{12}$$

then

$$R \geq [\chi_d^2(\epsilon)]^N \tag{13}$$

where χ_d^2 is a chi-square distribution with d degrees of freedom, $C_e(kT_s)$ is the covariance of the tracking error, N the number of points the specifications are concerned with, and d the dimension of the state vector ($d = 2n$ for a n-degree of freedom manipulator). $C_e(kT_s)$ can be determined by solving the difference equation

$$C_e((k+1)T_s) = A_{dcl}C_e(kT_s)A_{dcl}^T + DC_v(kT_s)D^T \tag{14}$$

Given Q_s and ϵ, the reliability lower bound is given by (13) for all different C_e which satisfy (12). The value of C_e depends on A_{dcl} which in turn is a function of the weighting matrices Q, S, G in the performance index. Hence, for different lower bound reliabilities, different Costs C will be obtained, and a performance function balancing the two would help selecting the best choice of Q, S, G. Such a performance function is introduced in section 3.2. In this example, it would balance error penalty and cost of control (by penalizing joint accelerations) to track a given trajectory, and the reduction of uncertainty due to measurement noise and incomplete modeling.

2.2 Computer Vision

The event locate object for AUVs endowed with a stereo vision system, can be translated by algorithms that require moving an AUV to different viewpoints to improve vision measurements. Hence, uncertainty associated to the event results from incomplete modeling of AUV dynamics, measurement noise on AUV speed and position sensors, camera calibration process, inaccuracies in stereo matching due to bad lighting, disparity errors due to pixel resolution and bad lighting.

Assuming that the camera calibration is reliable enough and spot noise is filtered, pixel truncation error is the main cause of errors for the vision subsystems. The pose estimate degrades with the distance of the object from the cameras due to this irreducible (from the vision point of view) uncertainty. However McInroy and Saridis(1990)[11] use N_v different viewpoints to estimate the pose of an object by stereo vision, and reduce the estimation uncertainty by averaging the N_v estimates. Therefore, one natural measure of Cost, given a desired Reliability, is N_v. Lower bounds for the Reliability may be found in the above reference. The average over a number of viewpoints does also reduce spot noise, thus it influences the quality of the matching process. Algorithms translating this event are thus distinguished by the viewpoints and the number of viewpoints they use.

2.3 Compliant grasping

A manipulator comes in contact with the environment while performing many useful tasks. Thus it may be required to exhibit a particular functional relation between the force it exerts and the displacement that results. One possible control strategy to achieve that is impedance control. Impedance control involves issuing a position command and assigning a relationship between the interaction forces and deviations from the desired position command. Thus, impedance control consists of a position control loop with the assigned impedance determining the stiffness of the manipulator[4].

Let x_0 be the nominal end-effector trajectory and x be the actual end-effector trajectory. Let f be the forces on the manipulator due to contact with the environment.

$$\underline{f} = K(\underline{x} - \underline{x}_0) + B(\underline{\dot{x}} - \underline{\dot{x}}_0) + J(\underline{\ddot{x}} - \underline{\ddot{x}}_0) \tag{15}$$

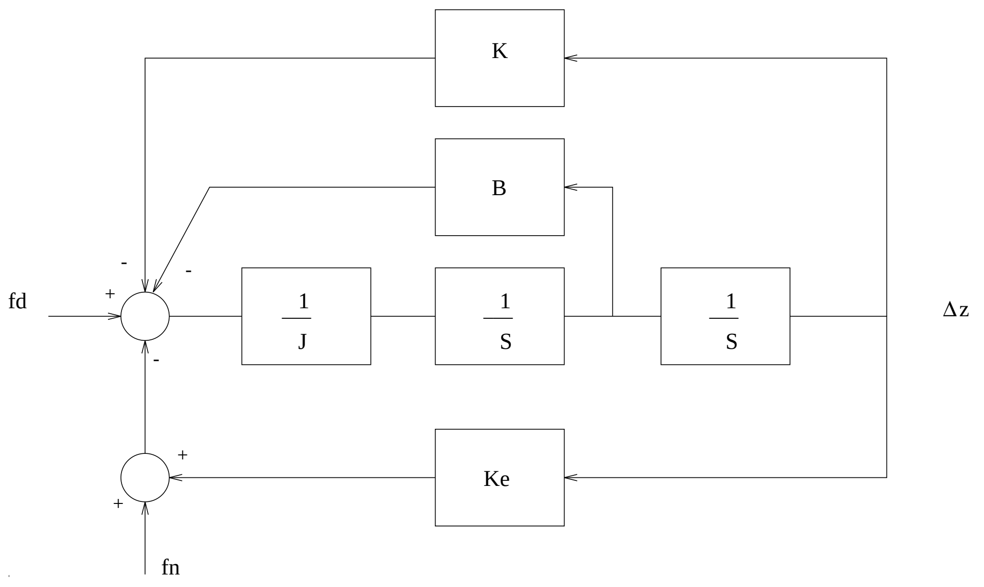

Figure 1: Continuous Mass, Spring and Damper block diagram.

Equation (15) represents a relationship between the force at the end-effector and motion about a nominal trajectory. If $\underline{x} = \underline{x}_0$ the force $\underline{f}$ is zero. Thus $\underline{x}_0$ can be considered the non-contact trajectory. The choice of the parameters K, B, and J depend upon the response desired from the system. If the specifications require that the actual force f is in a neighborhood ϵ of some desired force f_d, the uncertainty about the actual forces on the manipulator due to force sensor noise will lead to a compliant grasping not completely reliable.

Suppose a manipulator has to grasp some object using impedance control. After getting to a position above the object with the required tool pose and xy position, the manipulator tip (tool) must approach the object with a vertical downward movement along the z axis. Once the object is reached, the manipulator will try to grasp it after some desired force in the positive z direction is obtained or the pre-established duration time for the movement expires, whichever occurs first. In this study, compliance is assumed to work for all other components of $\underline{x}$, and K, B and J are scalars.

If the downward movement is exclusively due to a desired force f_d, the closed loop manipulator-environment can be roughly modeled as in Figure 1. The environment is modeled as a spring of constant K_e and errors from the manipulator position controller are ignored. $K_e = 0$ before contact, and $K_e \gg K$ after contact (very stiff object). The initial position of the manipulator is the nominal position. Measurement noise f_n is added to the force sensor. This is a reasonable model for all situations except immediately after contact, where a non-linear system behavior has been experimentally observed[14].

We may assign a zero cost to the algorithms if we are just interested on distinguishing them by their grasping reliability only. Other possible measures for the cost are the *delay-time* or *rise-time* of the deviation from the nominal position, if the concern is about the time taken by the movement before contact. From Figure 1, and ignoring the force sensor noise, the closed loop transfer function is

$$\frac{\Delta Z(s)}{F_d(s)} = \frac{\frac{1}{J}}{s^2 + \frac{B}{J}s + \frac{K'}{J}} \tag{16}$$

where $K' = K + K_e$ and $\Delta Z(s)$, $F_d(s)$ are the Laplace Transforms of the displacement from the nominal trajectory along z and the desired force f_d respectively.

Hence the following expressions are obtained for the natural frequency w_n and the damping ξ:

$$w_n = \sqrt{K'/J}\ ,\ \xi = B/(2\sqrt{K'J}) \tag{17}$$

A reasonable approximation for the delay-time (time elapsed while the system response raises from its initial value to 50% of the final value) when $0 < \xi < 1.2$ is given by [5]

$$t_d \simeq \frac{1 + 0.6\xi + 0.15\xi^2}{w_n} = \sqrt{\frac{J}{K'}} + \frac{0.3B}{K'} + \frac{0.0375B^2}{K'\sqrt{K'J}} \tag{18}$$

The delay-time depends on the 3 parameters J, K', B. Assuming a fixed J, t_d increases with B for a fixed K' and decreases with K' for a fixed B.

After contact, the main concern is about the time taken by the force sensed at the manipulator tip to settle down to the desired force f_d. If specifications require the force error in the z direction, $|f_z - f_{zd}|$ to be less than some accuracy by the time the object should be grasped, the system *settling-time* together with the force sensor noise will affect the reliability.

One definition of settling-time as the time the response takes to go from its initial value to within 5% of the final value leads to the following approximation:

$$t_s \simeq \frac{3}{\xi w_n} = \frac{6J}{B} \tag{19}$$

Hence the settling-time and consequently the reliability do not depend on K'. Again there is a tradeoff between cost and reliability: for some fixed K and J, if B is increased, the cost (identified here with the delay-time) will increase, but settling-time will decrease and the system will have more chances to attain the desired force before timeout, thus increasing its reliability.

The following instantiations of the definitions above for this particular example summarize and clarify the application of the formalism:

- problem element $\underline{f} = (f_z\ f_{zd})$
- problem solution $S(\underline{f}) = f_{zd}$
- solution approximation $U(\underline{f}, \phi) = f_z + f_n$, as obtained by algorithm ϕ
- algorithm $\phi = \Delta z$, from the position accommodation controller.

3 HIERARCHICAL DECISION MAKING AND REINFORCEMENT LEARNING

When dealing with large complex systems such as mobile robots, common questions are "how to measure performance?" and "how to improve performance". Energy consumption is an usual performance measure. However, it may be very difficult, if not impossible, to relate energy consumption with the performance of the underlying subsystems.

Here, we assume that each subsystem is designed to achieve its best possible performance, in the sense of not failing to meet its specifications the maximum possible number of times, without using too many resources. Assuming a fixed cost along time, in terms of resources consumption, if we can monitor whether a subsystem fails to meet its specifications each time its service is required, then it is possible to learn along time the best subsystem among a set of pre-designed alternatives, for each subgoal. Overall, the best task which accomplishes the main goal, is chosen based on the performance of the subsystems composing the task. The approach provides a cost function to compare different designs, which will be distinguished by the quality of the pre-designed algorithms and the way they are composed into tasks. It also provides the methodology to obtain convergence to the best possible solution given a design. Better designs will converge to smaller cost functions. Furthermore, it provides a simple way of improving performance along time through feedback, here consisting of success/failure signals only.

The design process follows a *bottom-up* approach, where the alternative primitive algorithms capable of implementing the problem represented by an event and the different events feasible for a given command are pre-specified. Subsequently, the *planning* problem (not considered here) consists of **composing** these events to build a task, as opposed to a *top-down* approach, where an original goal is recursively decomposed until a feasible task is obtained. On the other hand, the execution process flows both top-down (for decision-making) and bottom-up (for feedback). For each command, a task is selected based on the current probability distribution over the set of tasks. A task, in turn, is a sequence of events. The primitive algorithms translating each event are selected based on the current probability distribution over the set of algorithms for the event. Feedback from the environment where the HGDIM operates consists of successes and failures of the algorithms to meet their specifications. These are used to update in a bottom-up fashion the probability distributions at the decision levels. Figure 2 shows a diagram of the hierarchy just described. In the figure, LSA denotes a *Learning Stochastic Automaton* (see following section).

3.1 Hierarchical Decision Making

There are 2 stages of decision in the hierarchy considered here: the interface between the Organization and Coordination Levels, where commands are translated into tasks, and the interface between the Coordination and Execution Levels, where primitive events composing a task are translated into primitive algorithms. At each stage we need a performance measure to assist the decision making process. To avoid local minima, there must be, at each step, a chance of exploring decisions which are not the current best action. With this purpose, the interfaces of the IM are modeled by a 2-stage *Hierarchical Stochastic Automaton*(HSA) [2, 13](see Figure 2). At each stage, and at each step, the IM chooses the task or primitive algorithm to

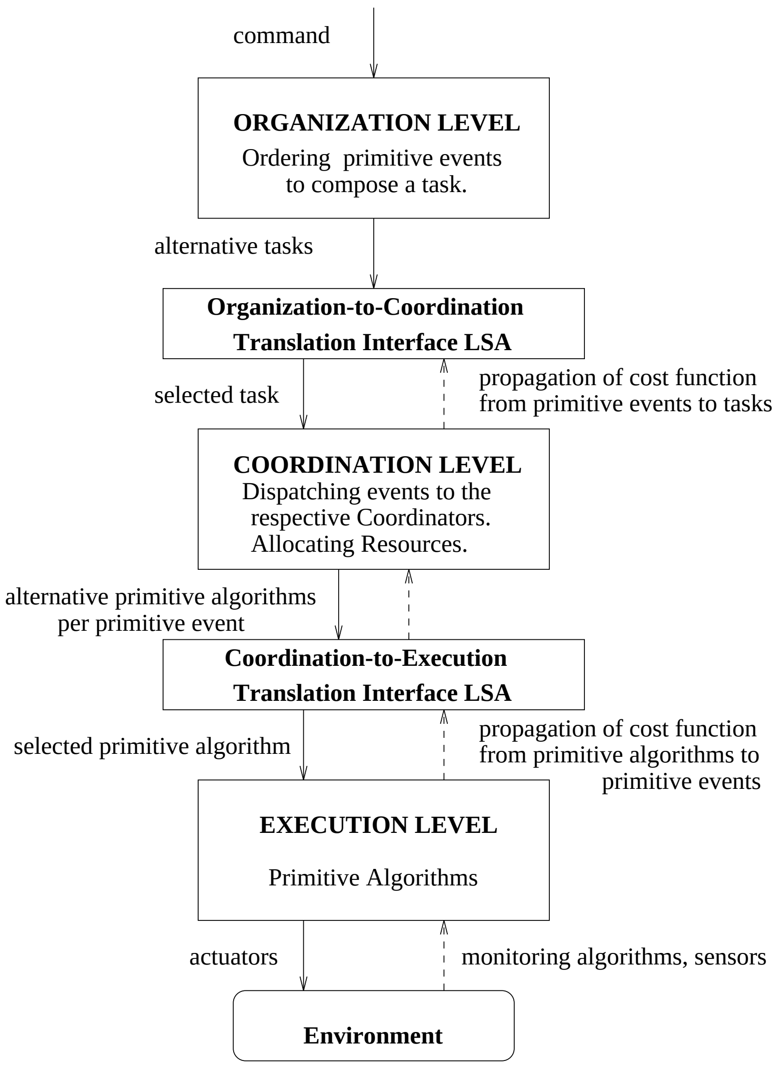
command
ORGANIZATION LEVEL
Ordering primitive events to compose a task.
alternative tasks
Organization-to-Coordination Translation Interface LSA
selected task
propagation of cost function from primitive events to tasks
COORDINATION LEVEL
Dispatching events to the respective Coordinators. Allocating Resources.
alternative primitive algorithms per primitive event
Coordination-to-Execution Translation Interface LSA
selected primitive algorithm
propagation of cost function from primitive algorithms to primitive events
EXECUTION LEVEL
Primitive Algorithms
actuators
monitoring algorithms, sensors
Environment

Figure 2: Diagram of the IM and its decision interfaces.

apply by **random decision**, based on the probability distribution function over the corresponding set. The probabilities are updated along time, based on the update of the cost function estimates.

3.2 Cost Function

The coherent definition of reliability and complexity introduced in the previous section allows the definition of a cost function combining the two, assuming that each algorithm is designed to meet a set of specifications:

$$J = 1 - R + \rho C \tag{20}$$

where R is the reliability, C the cost and ρ a weight factor which balances the two. In general ρ will be such that $\rho C \in [0, 1]$, so that the cost does not overwhelm the reliability. Examples of ρ are $\rho = \frac{1}{\max_{a \in A} C(a)}$ or $\rho = \frac{1}{\sum_{a \in A} C(a)}$, where A is the set of algorithms capable of solving a problem.

Equation (20) applies to all levels of the HGDIM, i.e., the performance of an algorithm, primitive event or task can evaluated by (20) if the cost and reliability are appropriately propagated bottom-up through the hierarchy.

A task t is composed by several events $e_k \in E^t$, where E^t is the set of events composing task t, occurring in sequence or in parallel. For each event e_k there exist a set of alternative algorithms A^k capable of solving the problem represented by the event. The propagation equations are:

Cost of event $e_k \in E^t$ is the minimum cost among all algorithms translating the event:

$$C(e_k) \triangleq \min_{a \in A^k} \{C(a)\}$$

Action probability p_a **of algorithm** $a \in A^k$ is the current probability of a being applied.

Reliability of event e_k is the average reliability among all algorithms translating the event:

$$R(e_k) \triangleq \sum_{a \in A^k} p_a R(a)$$

where $R(a)$ is the reliability of algorithm a.

The **cost of parallel execution of events** e_1, e_2 is

$$C(e_1 // e_2) \triangleq \max_{e_1, e_2 \in E^t} \{C(e_1), C(e_2)\}$$

while the **cost of** n **events** $e_1, \ldots, e_n \in E^t$ **executed in series** is

$$C(e_1 | \ldots | e_n) \triangleq \frac{1}{n} \sum_{i=1}^{n} C(e_i).$$

The successive application of these rules leads to the **cost of a task**, $C(t)$.

The parallel execution of events is not logically parallel from the reliability point of view. In fact, **all** events must be successful to complete a task. Hence, the **reliability of task** t is

$$R(t) \triangleq \prod_{e_k \in E^t} R(e_k).$$

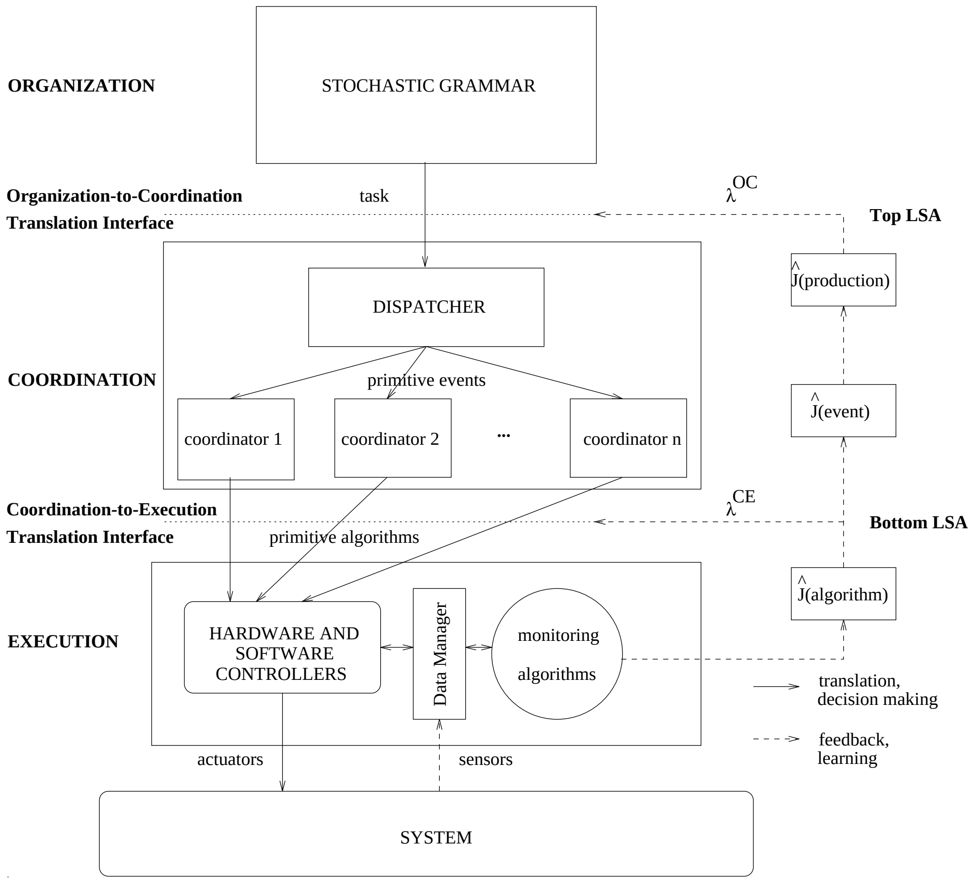

Figure 3: HLSA model of the IM decision interfaces.

3.3 Hierarchical Reinforcement Learning

The *a priori* knowledge of the actual cost function, using for example a model-based approach, is impractical in most cases. Hence we are interested on improving along time an estimate of the cost function, using feedback from the environment. A recursive estimate also provides some capability of adaptation to environment changes, thus reducing the need to predict all environment states. A model-based determination of Reliability assumes a stationary environment.

At the lowest stage of the HSA, the procedure recursively estimates Reliability of the primitive algorithms. This estimation is subsequently propagated to the upper stage, where its complement is added to the Cost to obtain the cost function estimate. These estimates of J are used by *Reinforcement Learning Algorithms* to update the *subjective probabilities* of the different algorithms capable of translating a *primitive event* and the different *tasks* capable of translating a command sent to the machine. The HSA becomes a HLSA (Hierarchical *Learning* Stochastic Automaton). A diagram showing the modeling of the IM translations interfaces as an HLSA is depicted in Figure 3.

The stochastic approximation reinforcement learning scheme used here at both stages was proposed by Fu[3] for its expediency. First, the reliability of primitive algorithms is estimated by

$$\hat{R}(N_{ij}+1) = \hat{R}(N_{ij}) + [z(N_{ij}+1) - \hat{R}(N_{ij})]/(N_{ij}+1) \tag{21}$$

which is the recursive version of the sample mean of the instantaneous performance function $z \in \{0,1\}$, 0 being a penalty (algorithm failed to meet the problem specifications) and 1 a reward. N_{ij} is the number of occurrences of environment state i and algorithm j. It is known that the sample mean estimate converges with probability one (w.p. 1) to the actual value: $\Pr\{\lim_{n\to\infty} \hat{R}(n) = R\} = 1$. From (21), $\hat{J}(N_{ij}) = 1 - \hat{R}(N_{ij}) + \rho C$. Next, $\hat{J}(N_{ij})$ is propagated bottom-up, using the propagation equations of subsection 3.2.

To update the probability density function over the set of tasks or algorithms, Fu's stochastic approximation reinforcement learning algorithm is also used:

$$p_{ij}(N_i+1) = p_{ij}(N_i) + \gamma(N_i)(\lambda_{ij}(N_i) - p_{ij}(N_i)) \tag{22}$$

where $N_i = \sum_j N_{ij}$, $0 \le \lambda_{ij}(N_i) \le 1$, $\sum_j \lambda_{ij}(N_i) = 1$, $i = 1, \ldots, d$ denotes environment states, $j = 1, \ldots, r$ denotes tasks or primitive algorithms. Given the estimates of the performance function at each time instant, $\lambda_{ij}(N_i)$ is defined by

$$\lambda_{ij}(N_i) = \begin{cases} 1 & \text{if } \hat{J}(N_{ij}) = \min_k \hat{J}(N_{ik}) \\ 0 & \text{if } \hat{J}(N_{ij}) \neq \min_k \hat{J}(N_{ik}) \end{cases} \tag{23}$$

We use $\gamma(N_i) = \frac{1}{N_i+1}$, which satisfies the condition of Theorem 2 in [3]. Hence, the probability density function of the action probabilities at state i will converge w.p. 1 to zeros for all the actions except the optimal action m, i. e., $\Pr\{\lim_{N_i\to\infty} p_{im}(N_i) = 1\} = 1$, where action m for state i is such that $J(N_{im}) = \min_k\{J(N_{ik})\}$, and J is the cost function at the decision stage under consideration. Using Fu's stochastic approximation reinforcement learning algorithm at both stages, the HLSA will converge w.p.1 to the optimal decisions at the two stages[7].

4 APPLICATION TO AN INTELLIGENT ROBOTIC SYSTEM

A manipulator mounted on an AUV has to grasp a cable whose 3D pose (position + orientation) is roughly known. The AUV has a pair of cameras overviewing the working space of the manipulator and used by a stereo vision system to determine more accurately the 3D pose of objects to be grasped. The manipulator motion may be compliant. Muddy waters may deteriorate the accuracy of vision algorithms. Hence, we consider a two-state environment: state 0 corresponds to clear waters and state 1 to muddy waters. Only one command is available: grasp cable. The event set E is composed by 5 events: e_1 : move-manipulator, e_2 : grasp-object-compliant, e_3 : locate-object, e_4 : plan-path, e_5 : grasp-object-hard.

Event e_1 represents a movement along a pre-planned path. Two computed-torque control algorithms are capable of translating e_1 corresponding to distinct choices of matrices Q and S in the performance index of equation (7). Both are prone to errors resulting from incomplete modeling of the manipulator dynamics. Event specifications state that at the end of the movement along the path the end-effector frame of the manipulator must be located within a boundary ϵ_1 of the pose set point - established by the vision system or by the initial rough estimate of the cable pose.

Event e_2 assumes that the manipulator is close enough to the object to be grasped. The manipulator gripper is opened, and compliantly approaches the object, until the desired 3D force and torque components are obtained. Then, the gripper is closed and the object grasped. Here, different algorithms correspond to different parameterizations of the impedance controller resulting in different equivalent mass-spring-damper (MSD) systems. The four algorithms translating e_2 are MSD1, MSD2, MSD3, MSD4, in *increasing* order of stiffness of the damping parameter. The stiffer the impedance controller the harder is to grasp an object if its location is not accurately known or if the AUV motion controller does poorly (for example in the presence of strong currents). On the other hand, with a very accurate location estimate and a stable AUV position, one may desire compliant control, but not too underdamped, so that grasping is more precise. The specification for e_2 requires an absolute steady state force error below some ϵ_2 specification. In the simulation, the reliability increases with stiffness under environment state 0, since the vision system may provide a more accurate pose estimation, but decreases with stiffness under state 1, where only the rough pose estimate is available.

Event e_3 determines the pose of an object using stereo vision algorithms. Both algorithms estimate the pose by averaging the results of a number of viewpoints N_0 for algorithm 0 and N_1 for algorithm 1. The specification for e_3 requires that the trace of the error covariance matrix of pose estimation is inside some tolerance interval $[-\epsilon_3, \epsilon_3]$. The trace will normally increase from state 0 to state 1 for both algorithms. Assuming $N_1 > N_0$, and making the cost proportional to the number of viewpoints, we simulate a better reliability and a larger cost for algorithm 1.

Event e_4 plans a path in cartesian space, conveniently samples that path, and using inverse kinematics routines, maps the path into joint space of the manipulator. This path goes to a memory space shared with e_1. This memory space is part of the World Model, if one exists. e_4 needs an argument stating the destination of the path, given the current departure pose. To simplify, we assume a completely reliable and zero cost path planner. It is also implicit that the AUV has moved to a position such that the cable is within the workspace of the manipulator.

Event e_5 is a non-compliant version of e_2. All compliance will be passive, i. e., a result of the manipulator mechanical compliance (assuming a "stiff" AUV position controlling system, which is not completely possible). Specifications are also related to the steady state force error, which is required to be less than $\epsilon_5 = \epsilon_2$. The reliability of the only algorithm used, a PI controller with no explicit compensation of dynamics, is necessarily smaller than that of any of the active compliance algorithms translating e_2, since no force control exists.

Table 1 shows the reliabilities and costs assigned to the simulation models of each primitive algorithm (except $R(e_4) = 1$ and $C(e_4) = 0$). The values were assigned based on the physical considerations above. Notice that specifications are made for each event, not for each algorithm. Cost does not change with the environment state. During the simulation, the reliabilities of the algorithms are estimated from the rewards/penalties resulting from successes/failures of their application over the environment, respectively. The algorithm rate of successes/failures is simulated by a Monte Carlo method from the tabulated reliabilities for each pair state/algorithm,

Table 1: Event specifications and simulated reliabilities and costs for the translating algorithms.

algorithm	ctc1	ctc2	msd1	msd2	msd3	msd4	N_0	N_1	pi
specs	$\|$ss-err$\| < \epsilon_1$		$\|$force ss-err$\| < \epsilon_2$				err var $< \epsilon_3$		$\|$force ss-err$\| < \epsilon_2$
reliab/st0	0.85	0.95	0.80	0.85	0.90	0.95	0.85	0.95	0.65
reliab/st1	0.85	0.95	0.95	0.90	0.85	0.80	0.70	0.80	0.65
ρ cost	0.1	0.3	0.0	0.0	0.0	0.0	0.3	0.5	0.1

and it is unknown to the HGDIM. Alternative tasks differ by the inclusion or not of compliance and by the refining or not of the knowledge of the strut pose. The stochastic regular grammar[1, 8] for the command is

$$\begin{array}{llcll}
0.5 & \text{p0} & \text{S} & \rightarrow & e_3\ \text{A} \\
0.5 & \text{p1} & \text{S} & \rightarrow & \text{A} \\
1.0 & \text{p2} & \text{A} & \rightarrow & e_4\ e_1\ \text{B} \\
0.5 & \text{p3} & \text{B} & \rightarrow & e_2\ \text{C} \\
0.5 & \text{p4} & \text{B} & \rightarrow & e_5\ \text{C} \\
1.0 & \text{p5} & \text{C} & \rightarrow & e_4\ e_1
\end{array}$$

The numbers to the left of the productions are the initial production probabilities.

Equations (21) and (22) were used to recursively learn the algorithms and task that minimize the cost function (20). The production probabilities are updated after a task is applied. A recursive sample mean with *forgetting factor*[7] was used to improve the adaptation of Reliability estimate, necessary when there is an unforeseen change in the state of the environment.

Figure 4 shows an interesting example of adaptation of the IM to an environment change not acknowledged by the IM. The figure shows the sample mean over 50 runs. At step 400 the environment reliabilities in table 1 are switched from state 0 (clear water) to state 1 (muddy water). Events e_2 and e_3 suffer dramatic changes, but only the latter influences the production sequence and thus the chosen task. The change in e_2 is only internal, i. e., when the environment state changes, so does the choice of the best algorithm translating the event. The change in the reliability estimate of e_3 provokes a task change from estimating the pose using stereo to the use of the apriori estimate, as expected, since stereo cost function is poor under bad lighting conditions. Notice that the reliability of applying production 1 is kept constant and equal to 0.5. For the initial environment states, the algorithms and productions learned the best translations. After the state change, the best algorithm of event e_2 is not selected with maximum probability during the time window plotted, but its probability is slowly converging to 1.

5 CONCLUSIONS

In this paper we formally stated a solution for the problem of Decision Making and Learning in Intelligent Machines, based on a cost function which includes reliability

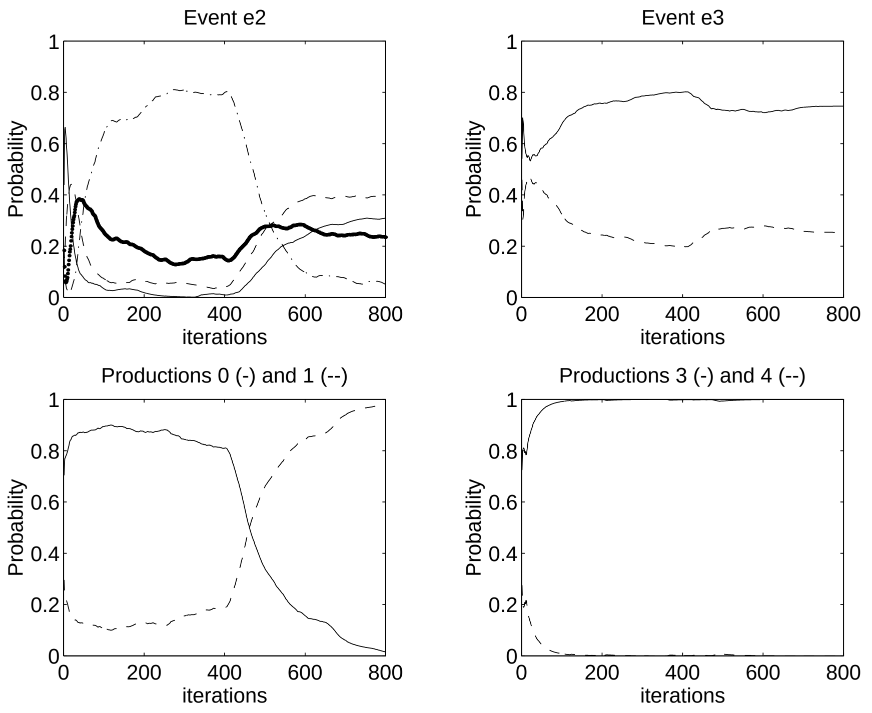

Figure 4: Evolution of probabilities for algorithms translating events e_2 and e_3 (top), and pairs of alternative productions (bottom).

and cost of the involved algorithms at the 3 levels of the hierarchy proposed by Saridis. A case study was presented where the proposed formalism is applied to a robotic system requiring coordination among control, vision and path planning. Results show the convergence of the reliability estimation and reinforcement scheme algorithms at the different levels of the IM hierarchy, and some degree of adaptation to changes in the state of the environment.

The methodology seems promising as a design tool for hierarchical controllers of large autonomous systems. It relies on existing knowledge to build the various subsystems, and provides a measure to compare different solutions and a mechanism to converge to the best solutions given a design.

Acknowledgments

The first author was supported by the portuguese research institution JNICT, under Grant #BD/1357/91-IA. The second author was supported by the NASA Grant #NAGW-1333.

References

[1] K. S. Fu and T. L. Booth. Grammatical inference: Introduction and survey – part II. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-5(4), 1975.

[2] K. S. Fu and J. M. Mendel. *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications.* Academic Press, 1970.

[3] K. S. Fu and Z. J. Nikoliç. On some reinforcement techniques and their relation to the stochastic approximation. *IEEE Transactions on Automatic Control*, AC-11(2):756–758, 1966.

[4] N. Hogan. Impedance control: an approach to manipulation: parts 1,2,3. *Journal of Dynamic Systems, Measurement and Control*, 107:1–24, March 1985.

[5] B. C. Kuo. *Automatic Control Systems.* Prentice Hall, 1987.

[6] F. L. Lewis. *Optimal Estimation.* John Wiley and Sons, 1986.

[7] P. U. Lima. *Intelligent Machines as Hierarchical Stochastic Automata.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1994.

[8] P. U. Lima and G. N. Saridis. Hierarchical reinforcement learning and decision making for Intelligent Machines. In *Proceedings of 1994 IEEE Int. Conf. Robotics and Automation*, May 1994.

[9] P. U. Lima and G. N. Saridis. A performance measure for Intelligent Machines based on complexity and reliability. In *Proceedings of SY.RO.CO 94*, September 1994.

[10] G. L. Luo and G. N. Saridis. Optimal/PID formulation for control of robotic manipulators. *IEEE Journal of Robotics and Automation*, 1(3), December 1985.

[11] J. E. McInroy and G. N. Saridis. Reliability analysis in Intelligent Machines. *IEEE Transactions on Systems, Man and Cybernetics*, 20(4), 1990.

[12] J. Musto and G. N. Saridis. An entropy-based reliability assessment technique for intelligent machines. In *Proceedings of 8th International Symposium on Intelligent Control*, Aug 1993.

[13] K. S. Narendra and M. A. L. Thathachar. *Learning Automata - an Introduction.* Prentice Hall, 1989.

[14] Michael Ryan. Implementation of robotic force control with position accommodation. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, 1992.

[15] Technical Committee on Intelligent Control. Report of task force on Intelligent Control, IEEE Control Systems Society. *IEEE Control Systems Magazine*, 14(3), June 1994. P. Antsaklis, editor.

[16] J. Traub, G. Wasilkowsky, and H. Woźniakowsky. *Information-Based Complexity.* Academic Press, Inc., 1988.

[17] K. P. Valavanis and G. N. Saridis. *Intelligent Robotic Systems.* Kluwier Publishers, 1992.