Measuring Complexity of Intelligent Machines

Pedro Lima, George Saridis
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

Abstract

In this paper we introduce a formalism which combines reliability *and* complexity *as performance measures for Intelligent Machines.*

For a given desired reliability, *different algorithms may be available which are reliable enough. Hence it is important to have a means of choosing the algorithm of least cost among the reliable ones. By cost we do not mean CPU time only but other features such as memory space. Information-Based Complexity provides a solid formalism to deal with different sources of information, thus with distinct algorithms at all levels of the machine.*

A case study related to image processing illustrates the method.

1 Introduction

The analytic approach to the theory of Intelligent Machines (IM) [4] has developed in the past few years a formalism for the architecture of an Intelligent Machine[5, 3, 7, 8] which is now well established. Efforts looking towards a practical implementation of the formalism expressed in those works took place recently[1] and are currently being pursued.

The closer one gets to the implementation problems, the stronger is the need for performance measures. However, an approach somewhat different from traditional control cost functionals is needed, since a myriad of different sources of information is present in such a machine. This is what we call a *wide-sense control problem*. The final stage of execution of a high level task includes vision, control and path planning algorithms, to name just the most relevant.

A first attempt to deal with this problem is described in [2], where a reliability-based measure is studied for pose-estimation problems. Since it is never possible to overpass all the uncertainty associated to the controlled system, it is important to study, for a given set of algorithms, the probability of success of the problem-solving task implemented by those algorithms. This allows an entropy-based measure to guide the system in its choice of the most reliable algorithm for a given task.

In this paper a complementary approach to the use of reliability as a performance measure is introduced. For a given desired reliability, different algorithms may be available which are reliable enough. Hence it is important to have a means of choosing the algorithm of least *computational cost* among the reliable ones. *Information Based Theory of Complexity* [6] provides a solid formalism to deal with different sources of information, thus with distinct algorithms at all levels of the machine.

The paper is organized as follows: after this introduction, section 2 briefly describes the formalism of Information-Based Complexity. The core of the paper is section 3 where the application of the theory to the 3-levels hierarchy (Organization, Coordination and Execution) of the Intelligent Machine is formalized. A simple case study using image processing algorithms is presented in section 4 to show how to apply the techniques described below, and finally section 5 concludes the paper.

2 Information-Based Complexity

The *computational complexity* of a *problem* may be defined as "its intrinsic difficulty as measured by the time, space or other quantity required for its solution" [6]. More formally this is equivalent to the cost of the optimal *algorithm* for the solution of the *problem* in the sense defined below.

We briefly summarize below the general formulation of information-based complexity [6]. Hopefully the details will be clarified by the formulation of the case study.

2.1 Problem formulation

For each $f \in F$, where F is a set and f is called a *problem element*, the problem is to compute an approximation $U(f)$ of $S(f)$, where $S : F \rightarrow G$ is called a *problem solution* and G is a normed linear space over the scalar field of real or complex numbers. In this interpretation, G is the *feature space* of the problem, $S(f)$ represents a vector of desired features for f and $U(f)$ computes the actual corresponding features from f. To measure the distance between $S(f)$ and $U(f)$ we use an absolute error criterion, $\|S(f) - U(f)\|$.

$U(f)$ is an *ϵ-approximation* of $S(f)$ iff $\|S(f) - U(f)\| \leq \epsilon \geq 0$. We wish a computed element $U(f)$ which is an ϵ-approximation in one of three settings: *worst-case, average* and *probabilistic.*

2.2 Information

It is assumed that the only initial knowledge we have about f is that it belongs to the set F, and also that we may gather more knowledge about f using computations of the form $L(f)$, $L : F \rightarrow H$ for some set H.

The information $\mathcal{I}(f)$ is then defined as

$$\mathcal{I}(f) = [L_1(f), L_2(f), \ldots, L_n(f)],\ \forall f \in F.$$

.

In the above sense, we may say that $U(f)$ computes the feature vector obtained when algorithm $\Phi : \mathcal{I}(f) \rightarrow G$ is used. Φ is an *algorithm* which computes an approximation of $S(f)$ given the information $\mathcal{I}(f)$.

2.3 Model of computation

The initial assumptions are: *i*) a sequential model of computation and *ii*) all information and combinatorial operations are performed with infinite precision and finite cost.

The model postulates a constant cost c for each information operation $L(f)$ and unit cost for each combinatory operation performed by Φ over $\mathcal{I}(f)$.

Cost is defined as

$$\operatorname{cost}(\Phi) = \sup_{f \in F}\{c_i(f) + c_p(f)\} \tag{1}$$

where $c_i(f)$ is the cost of getting information about f and $c_p(f)$ is the cost of processing that information by algorithm Φ.

The *error* associated to Φ is, in a *probabilistic setting*

$$e(U) = \sup_{f \in F}\{\|S(f) - U(f)\| : P(\|S(f) - U(f)\| < \epsilon) \geq 1 - \delta\} \tag{2}$$

In simple words, this means that we control the error of estimating $S(f)$, keeping it below ϵ, except in a subset of G with measure δ. The cost is obtained from the constraints imposed to the error.

Finally, *ϵ-complexity* (*complexity* for short) of a problem is the minimal cost among all algorithms which solve the problem with error at most ϵ: $\operatorname{comp}(\epsilon) = \inf_\Phi\{\operatorname{cost}(U) : e(U) \leq \epsilon\}$

The algorithm Φ^* that achieves the minimal cost is called an *optimal algorithm.*

3 A Performance measure for Intelligent Machines

A major concern of a control system designer is the *performance* measure for the controlled system. The design goal is such that the controlled system should maximize that measure.

When dealing with very large systems, some amount of uncertainty must be assumed 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 the Intelligent Machine [4]: 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. Zames [9] suggests that feedback reduces this uncertainty in conventional control problems; 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 didn't move) composing a *task* [7]; at the **organization level**, there is uncertainty in terms of the success of the *task* executed.

Thus, if we reduce the uncertainty about the actions taken by the Intelligent Machine, we are improving performance, since each of the actions at all levels will have a broader chance of success. This can be interpreted as a problem of improving *reliability*, by choosing at the **coordination/execution levels** the most *reliable* low-level algorithms for each primitive event [2], and at the **organization level** the most reliable ordered sequence of primitive events (task).

However, the price to pay when a high reliability is required may be unacceptable, either in terms of computational time or resources such as memory, number of image frames or sampling rate.

Thus it is desirable to measure the relative *costs* of the different algorithms capable of solving the problem represented by an event, with the required accuracy. This cost includes the prices of getting information and processing it. Depending on the model used, different features are weighted (CPU time, memory).

Given the above, maximizing the *performance* of the entire machine may then be formulated as a Discrete Optimization Problem:

$$\begin{array}{ll} \min & \text{cost}(a) \\ \text{s. t.} & \text{reliability}(a) \geq R_d(a) \end{array}$$

where R_d is the reliability required by the command sent to the machine and a is a task translation for the command.

Reliability is defined by

$$\text{R} = \text{P(worst case error w/r to specifications} < \epsilon) \quad (3)$$

for some $\epsilon > 0$.

Suppose we consider in (2) $S(f)$ as a vector of specifications for a given problem. The problem solution $S(f)$ is for example the desired overshoot of a control algorithm implementing a move robot event, and the problem element f is the output signal where $S(f)$ is computed from. Hence,the formulation of Information-Based Theory of Complexity is appropriate to help us in the design of the IM, expressed by the optimization problem above. If we make $R_d = 1 - \delta$, when determining the cost of U by equations (1) and (2) we guarantee that $e(U)$ is such that its probability of being below ϵ (i. e. the reliability of the solution) is lower-bounded by R_d.

In order to determine the cost of a task, we need to know how to determine it from the cost of its composing events. If we assume that the cost of one event (minimal cost of the algorithms which implement the event) does not depend on the cost of the other events:

$$\begin{array}{rcl} \text{cost(sequence of events)} & = & \sum \text{cost(each event in the sequence)} \\ \text{cost(concurrent events)} & = & \max \text{cost(concurrent events)} \end{array} \quad (4)$$

In view of the above, the problem of finding the sequence of actions that solve a problem by the Intelligent Machine with least cost given the current knowledge and a *desired reliability* may be presented as follows:

1. Design the low-level algorithms imposing some ϵ accuracy for each algorithm;

2. Wait for command specifying the problem and the desired reliability $R_d = 1 - \delta$ for its solution;

3. Translate command into n pre-stored *tasks* [7] $a_1, \ldots, a_n$ capable of solving the problem. We are not concerned with planning here: it is assumed that the tasks were planned prior to this. Thus, *learning* the correct reliabilities is an important issue. However, this will only be addressed in later work;

4. Translate each task a_i into m_i pre-stored ordered *events* $e_1, \ldots, e_{m_i}$, and each event e_j into l_j possible algorithms $g_1, \ldots, g_{l_j}$;

5. For each event e_j determine the reliability of the event as the reliability of the most reliable algorithm for that event, and the cost of the event as the cost of the least costly algorithm for the event;

6. For each task a_i, $i = 1, \ldots, n$ determine the cost of the task, using the rules of composition of event-cost (4), and the reliability of the task as the product of the reliabilities of the events composing the task;

7. Among the tasks with reliability greater than or equal to the desired reliability for the task, determine the task of least cost and apply it. Go to 2.

Without learning, this algorithm will only work in static environments if we initially know the correct reliabilities of each algorithm. Moreover, there is a risk of combinatorial explosion when commands are translated into tasks, these into events and these into algorithms. These are topics which will be addressed in the near future.

4 Study of an image processing problem

In order to provide an example of application of the above formalism, an image processing problem was formulated.

Given a rectangle inside a M x M pixels image, the problem was to estimate the position of the rectangle in the image (see figure 1), that is, its central pixel of coordinates (i_c, j_c). The pixels inside the rectangle were initially set to 1, while the outside pixels were set to 0. Zero mean gaussian noise was added to the initial value of each pixel in the whole image.

Several assumptions were made with the goal of simplifying the mathematical analysis and the simulation: *i)* The area of the rectangle is known, $A = (y_e - y_b)(x_e - x_b)$; *ii)* The whole rectangle is inside the boundaries of the image; *iii)* No more objects

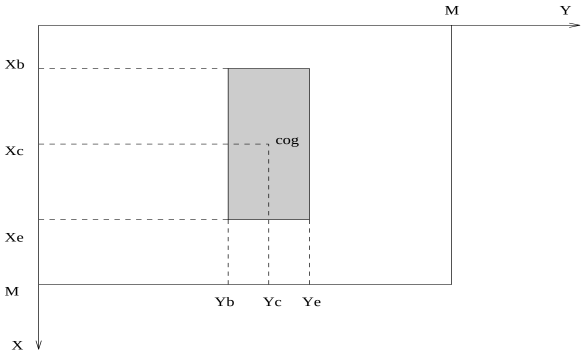

Figure 1: Image processed by the two algorithms

are present in the image; *iv*) Errors resulting from poor image resolution or computational roundoff were not considered, that is, without noise both algorithms should determine the rectangle position with infinite accuracy.

Two algorithms were proposed to solve the problem:

- The open loop algorithm determines the *center of gravity* of the total image, using the equations

$$\begin{aligned} \hat{x_c} &= \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} j b(i,j)}{A} \\ \hat{y_c} &= \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} i b(i,j)}{A} \end{aligned} \tag{5}$$

  where $b(i,j)$ is the brightness of pixel (i,j).

- The closed loop algorithm correlates the image (*feedback*) with a pattern rectangle (*reference*) equal in size to the original noise-free rectangle of the image and with the same orientation. The rectangle is assumed to be centered inside the P x P pixels pattern image. The coordinates (x_c, y_c) of the pixel with the greatest correlation coefficient are the estimates of the rectangle position in the image.

The objectives were twofold: *i*) apply the proposed methodology to study and compare the cost of the *open loop* and *closed loop* algorithms; *ii*) compare in terms of cost a wide-sense *open loop* solution with a wide-sense *closed loop* solution for the problem.

4.1 Problem formulation

F is the set of rectangles with size $(x_e - x_b)$ by $(y_e - y_b)$ that fit inside the M x M image.

The goal is to compute an ϵ-approximation for $S : F \to \mathcal{R}^2$, that is, we want to determine an estimate $U(f)$ of $S(f)$ such that

$$||S(f) - U(f)|| < \epsilon$$

where $S(f) = [x_c\ y_c]'$ and $U(f) = [\hat{x}_c\ \hat{y}_c]'$.

To simplify the analysis, and since what happens in one of the directions is similar to what happens in the other, we will only check the error estimate of x_c.

$$|\hat{x}_c - x_c| < \epsilon.$$

4.2 Information

The computations $L_{ij}(f),\ L_{ij} : F \to \mathcal{R}$ have the format

$$L_{nij}(f) = \begin{cases} 1 + n_{ij} & \text{if pixel}(i,j) \in f \\ n_{ij} & \text{if pixel}(i,j) \notin f \end{cases}$$

and so $\mathcal{I} = [L_{111}(f), \ldots, L_{NMM}(f)]$ where n denotes the nth average, N is the total number of averages, M the number of pixels on each side of the image and n_{ij} is a random variable representing the noise at pixel (i,j), $n_{ij} \sim \mathcal{N}(0, {\sigma_b}^2)$, i.i.d.

4.3 Model of computation

In this simple approach, we restrict the set of algorithms to those that can solve the problem in, say polynomial time. This means that we are more concerned about the cost of getting information. For example, if a mobile robot has to stop and get several frames of a scene in order to speedup posterior computations of its locations, we prefer a slowest algorithm that however requires less stopping time for the robot. The algorithm may run while the robot is performing other tasks.

Hence, $\text{cost}(\Phi) = \text{cost}(\mathcal{I}, f) = cN_{min}$, that is, cost is proportional to the minimum number of averages needed by algorithm Φ to get the error below ϵ.

4.4 Study of the algorithms

The open loop algorithm estimates the center coordinates of the rectangle using equations (5). Since the brightness of each pixel is a gaussian distributed random variable

$$p_{B_{ij}}(b_{ij}) \sim \begin{cases} \mathcal{N}(1, \sigma_b^2) & \text{pixel}(i,j) \in f \\ \mathcal{N}(0, \sigma_b^2) & \text{pixel}(i,j) \notin f \end{cases}$$

and the B_{ij}'s are uncorrelated from pixel to pixel (given that they are independent), it can be deduced that, after N averages of distinct frames of the same image and if we assume independent noise from frame to frame, $p_{\hat{X}_c}(\hat{x}_c) \sim \mathcal{N}(\mu_{\hat{X}_c}, \sigma^2_{\hat{X}_c})$ with $\mu_{\hat{X}_c} = x_c$ and

$$\sigma^2_{\hat{X}_c} = \frac{M^2(M+1)(2M+1)\sigma_b^2}{6A^2N} \quad (6)$$

Now, given an accuracy ϵ and a desired reliability R_d, we want to determine N such that $P(|\hat{x}_c - x_c| \leq \epsilon) \geq R_d$

From the table of standard normal we get η in $P(\frac{|\hat{x}_c - x_c|}{\sigma_{\hat{X}_c}} \leq \eta) = R_d$

Making $\eta = \epsilon/\sigma_{\hat{X}_c}$ and using (6) we finally get

$$N \geq N_{min} = \frac{\eta^2 M^2(M+1)(2M+1)\sigma_b^2}{6A^2\epsilon^2} \quad (7)$$

The inequality in (7) comes from the fact that we want the reliability to be lower bounded by R_d.

The closed loop algorithm looks for the pixel where the noisy output of the correlator achieves a maximum when the pattern is displaced around the image. Due to noise, there is some probability that the wrong pixel is chosen. In order to make the problem tractable, we have to make some assumptions, such as working at 1D again, and considering errors of 1 pixel displacement at most.

If the correlator input $u_{corr}(x) = r(x) + n(x)$, where $r(x)$ is a rectangle of length $x_e - x_b$ and $n(x)$ is gaussian noise of zero mean and variance σ_b^2/N, and the impulse response of the correlator is $r(x_c - x)$, then $y_{corr}(x) = y_r(x) + y_n(x)$ is the correlator output, where $y_r(x)$ is an isosceles triangle of length $2(x_e - x_b)$, centered at x_c and of height $x_e - x_b$, and $y_n(x)$ is gaussian noise with mean x and variance $\sigma_n^2 = \sigma_b^2(x_e - x_b)/N$.

Now, if we assume errors of 1 pixel at most and we say that, for convenience, 1 pixel corresponds to an ϵ displacement, the reliability $P(|\hat{x}_c - x_c| < \epsilon) = P(y_{corr}(x_c + \epsilon) - y_{corr}(x_c) < 0$ and $y_{corr}(x_c - \epsilon) - y_{corr}(x_c) < 0)$.

Let us assume that the output noise of the correlator is independent from pixel to pixel. This is not actually true, but it allows us to proceed. Since the two random variables are correlated, under this assumption we will obtain a smaller probability, hence obtaining an upper bound for N_{min}. Noticing that the sum of two random variables jointly and marginally gaussian is another gaussian distributed random variable, we finally get $R_d = P^2(z < 0)$, $z \sim \mathcal{N}(-\epsilon, 2\sigma_n^2)$, or $\sqrt{R_d} = P(\frac{z+\epsilon}{\sqrt{2\sigma_n^2}} < \eta)$, where η can be read from a table of standard normal and is made equal to $\frac{\epsilon}{\sqrt{2}\sigma_n}$, hence the upper bound is

$$N_{min} = \frac{2\eta^2\sigma_b^2(x_e - x_b)}{\epsilon^2} \quad (8)$$

Comparing with N_{min} for the open loop algorithm (7), it may be noticed that the closed loop upper bound on the number of averages does not depend on the size of the image M, while the minimum number of averages for the open loop algorithm increases with M, thus it is possible, with a reasonable ratio of image size to pattern size, to show that the closed loop algorithm upper bound for N_{min} will be below the actual value of N_{min} for the open loop case. Simulations show that (8) is a loose upper-bound and that in practice the cost of the closed loop algorithm is much smaller, for the same reliability. If the cost of processing information is also considered, other intermediate solutions between the two algorithms will have less cost, since the open loop algorithm is computationally fast.

4.5 Simulation results

The open loop algorithm was tested with different sets of parameters as follows:

- $R_d = 90\%$ and $\epsilon = 0.1$ and 0.2.
- $\epsilon = 0.3$ and $R_d = 90\%$ and 95%.

Each of the setups was tested with standard deviation of pixel noise $\sigma_b = 0.1$ and $\sigma_b = 0.3$. Each side of the image had 32 pixels and the rectangle had 13 pixels in the x direction, 9 in the y direction.

σ_b	0.1		0.3	
ϵ	N_{min}	R	N_{min}	R
0.1	73	0.8733	655	0.8933
0.2	18	0.8333	164	0.8533

Table 1: Simulation results showing values of reliability and minimum number of averages (cost) for a desired reliability of 90 %.

σ_b	0.1		0.3	
R_d	N_{min}	R	N_{min}	R
90 %	8	0.9267	73	0.8733
95 %	11	0.9400	103	0.9467

Table 2: Simulation results showing values of reliability and minimum number of averages (cost) for a desired $\epsilon = 0.3$.

The simulations were made in PRO-MATLAB Version 3.5i, running on a Sun SparkStation. The results

N_{min}	R_{ol}/σ_b	R_{cl}/σ_b
4	98%/0.2	99%/2.0
9	97%/0.3	98%/3.0

Table 3: Simulation results showing values of reliability for the same minimum number of averages (cost) and different noise variances, using the open loop (ol) and closed loop (cl) algorithms. $\epsilon = 1$.

are presented in tables 1 and 2, showing for each setup the reliability obtained after 150 runs with a number of averages greater than the minimum theoretically required (also shown).

In general, the outcomes agree quite well with the expected results. In some cases, reliability is slightly lower than expected. This may be explained by the fact that MATLAB's random number generator does not assure complete independence of the output values, hence additional terms would be present in equation (7), raising the lower bound on the number of averages.

The closed loop algorithm was simulated under the same setup. By trial and error, we determined the standard deviation of the superimposed noise needed to obtain reliabilities close to those of the open loop case with the same number of averages. Table 3 shows these results. The same N_{min} is sufficient to obtain accurate estimates with the same reliability, but with a noise standard deviation approximately 10 times larger.

This means that, given a desired reliability, we may use the formalism to measure the cost of the alternative algorithms. On the other hand, given a N we can determine the maximum possible reliability and minimum possible cost of an algorithm.

5 Conclusions and future work

A first approach to the study of computational complexity of Intelligent Machines was described in this paper. The formalism introduced is based on Information-Based Theory of Complexity and formulates the problem of maximizing the performance of the Intelligent Machine as a Discrete Optimization problem, where the cost of a task is to be minimized, and the constraints are imposed by a required Reliability.

Future work will apply the introduced formalism to other kinds of problems present in the execution level of the Intelligent Machine (control, path planning) and extend it to the total hierarchy of the Machine. Another research topic of interest is how to articulate reliability and complexity at all levels of the IM, and to use *feedback* to learn the reliability of the algorithms along time.

Acknowledgments

The first author was supported by JNICT (portuguese government institution for research support) under Grant #BD/1357/91-IA, and the second author by the NASA Center for Intelligent Robotic Systems for Space Exploration (CIRSSE) under Grant #NAGW-1333, where the first author also works as PhD student.

References

[1] Don R. Lefebvre and George N. Saridis. A computer architecture for Intelligent Machines. In *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, may 1992.

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

[3] Michael C. Moed and George N. Saridis. A Boltzmann machine for the organization of intelligent machines. *IEEE Transactions on Systems, Man and Cybernetics*, 4(4):323–334, sep 1990.

[4] George N. Saridis. Toward the realization of intelligent controls. *IEEE Proceedings*, 67(8), 1979.

[5] George N. Saridis. Entropy formulation for optimal and adaptive control. *IEEE Transactions on Automatic Control*, 33(8):713–721, 1988.

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

[7] Kimon P. Valavanis and George N. Saridis. *Intelligent Robotic Systems*. Kluwier Publishers, 1992.

[8] Fei-Yue Wang and George N. Saridis. A coordination theory for Intelligent Machines. *Automatica*, 26(5):833–844, 1990.

[9] G. Zames. On the metric complexity of causal linear systems, ϵ-entropy and ϵ-dimension for continuous time. *IEEE Transactions on Automatic Control*, AC-24(4):220–230, 1979.