The potential of soft-computing methods for mission systems: A tutorial

A.J. van der Wal
TNO-FEL Physics and Electronics Laboratory
P.O. Box 96864
NL-2509 JG The Hague
The Netherlands
e-mail: vanderwal@fel.tno.nl

1. SUMMARY

In the past decade information processing in general and signal processing in particular has undergone a revolutionary shift from pure AI-oriented methods towards a wide diversity of nonlinear, soft-computing methods that often have a paradigm in biological systems. Among the chief characteristics of these methods are their ease of application, synergy through nonlinearity, their robustness, human-friendliness, the ability to handle ambiguous (even conflicting) information, the intrinsic ability to handle vague notions and do human-like inferencing, the possibility to take into account multiple goals, their learning capability, and the separation of concerns. Although soft computing has proven to be successful in a growing number of application areas, a general theory encompassing all soft-computing methodologies together with standard linear processing methods is still lacking. In soft-computing literature it is seldom discussed *why* a particular method has been selected, or *why* a particular approach is advantageous for solving a problem. In this paper we present a concise discussion of the characteristics of the key softcomputing technologies and their possible impact on the design of mission systems. Because mission systems show a trend towards higher levels of autonomy, the complexity of these systems will increase significantly. At the same time there exists a tendency towards miniaturization, *e.g.* to avoid detection (UAVs). These two competing developments can only be reconciled by the integration of softcomputing methods into mission systems.

2. INTRODUCTION

2.1 Mission Systems

A mission system consists of an ensemble of hardware and software that is aimed at the successful completion of the mission. In this sense many systems may be considered as a mission system, depending on their level of complexity and autonomy. The concept of a mission is well-established in the aerospace community, although landbased and maritime operations face similar problems on a conceptual system level. Missions are generally initiated by man and still the majority of missions is manned to ensure that unexpected and difficult situations can be dealt with adequately. In addition we observe an increasing interest in *unmanned* missions. This is caused by changes in the world situation that complicate the execution of full-blown missions, but at the same time require mission systems to be extremely well-informed. In this context unmanned, robotic reconnaissance vehicles such as UAVs are developed.

Recent history shows that the nature of military operations changes rapidly: Although sensors are vital to the success of any military mission, it becomes at the same time much more difficult to interpret these observations. This can be illustrated by the introduction of stealth technologies (radar), by which planes become much more difficult to detect by radar, the subtleties of 'peacekeeping' missions compared to classical, full scale warfare scenarios, and finally the complexities and greater vulnerability of navy vessels operating close to shore ('littoral warfare'). Finally it should be noted that there is a genuine need to fuse sensor generated information, at least at the higher levels of command and control: the man-machine interface being the limiting factor. Although new sensors

Paper presented at the RTO SCI Symposium on "The Application of Information Technologies (Computer Science) to Mission Systems", held in Monterey, California, USA, 20-22 April 1998, and published in RTO MP-3.

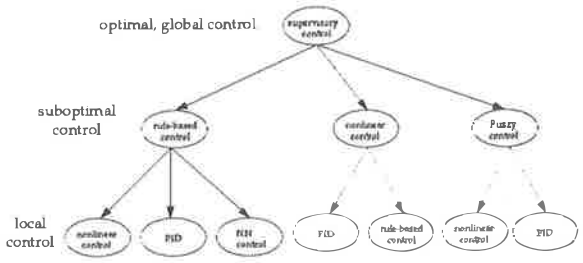

Figure 1 *Hierarchy of autonomous control processes in a mission system.*

have been developed (e.g. GPS) and accuracy and resolution in space and time of most existing sensors have greatly increased in time, the bandwidth of the man-machine interface has not. The situation of having to deal with more information than one can process in a certain time is not unsimilar to the situation where a *lack* of information exists. Both situations involve taking decisions in the presence of uncertainty and would benefit from intelligent data reduction techniques, such as softcomputing.

Other reasons for initiating unmanned missions are the safety of personnel and new tactical doctrines. Even in unmanned missions there may be on-line control by man via *teleoperation*, resulting in *telepresence*, but the mission can also be fully autonomous. Simulation may play an increasingly important part in the development and testing of a mission system prior to its deployment. Another important issue in mission systems (either manned or unmanned) is the man-machine interface (MMI), despite of (or sometimes because of) the ever-increasing speed and complexity of computer systems. Although it is not easy to give a definition of a mission system, we will in the present paper use the following working definition: *"A mission system is a system that supports the goal of a mission at a certain level of autonomy by optimizing subtasks"*.

In this recursive definition subtasks may equally well be viewed as mission systems themselves and thus a hierarchy of different mission tasks is defined at a number of levels of autonomy (Fig.1). A specialized control system has a particular task with only limited autonomy on the mission system level, whereas a software package for overall mission management will have more autonomy. Yet both systems are mission systems. Typically a mission management system will be responsible for high level goals such as the allocation of resources, the scheduling of tasks at certain phases in the mission, as well as providing adequate information to the humans responsible for the mission (Fig. 2). In fact the key issue in mission management is *optimization*. An example is the allocation of resources, *e.g.* fuel, computing power, the supply of energy, information, sensors, *etc.* But also subsystems such as controllers aim to optimize their functioning, *e.g.* by selftuning and robust performance. The goals of subsystems will often be competing with each other. The central problem to be studied in relation to mission systems therefore is that of *global* optimization *vs. local* optimization.

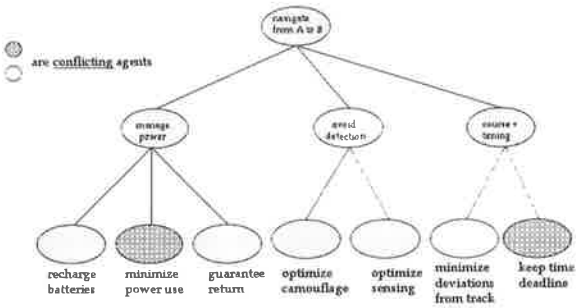

Figure 2 *The mission "Navigate from A to B" consists of many (sub)missions. Completing a mission successfully involves global optimization, which is hard because many competing processes take part in the process. The dark and white-colored processes indicate competing agents.*

2.2 Softcomputing

In the past decade information processing in general and signal processing in particular has undergone a revolutionary shift from pure AI-oriented methods towards a wide diversity of nonlinear, soft-computing methods that often have a paradigm in biological systems. Among the chief characteristics of these methods are their ease of application, synergy through nonlinearity, their robustness, human-friendliness, the ability to handle ambiguous (even conflicting) information, the intrinsic ability to handle vague notions and do human-like inferencing, the possibility to take into account multiple goals, their learning capability, and the separation of concerns. Although soft computing has proven to be successful in a growing number of application areas, a general theory encompassing all soft-computing methodologies together with standard linear processing methods is still lacking. In soft-computing literature it is seldom discussed

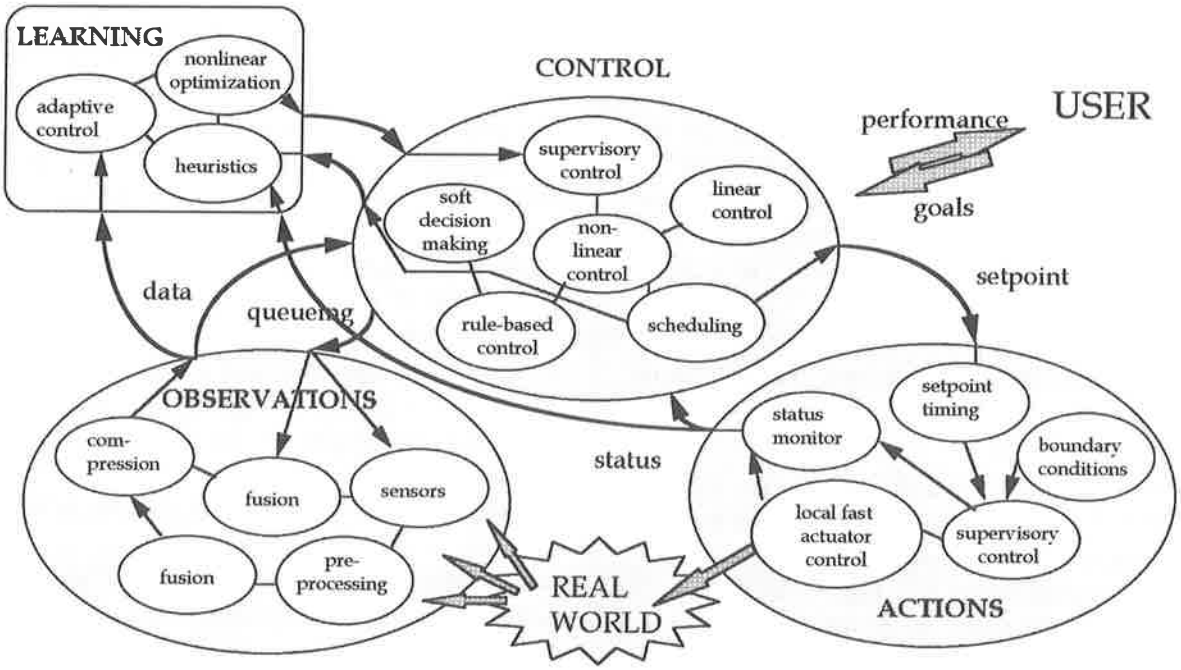

Figure 3 *The fundamental processes within a mission.*

why a particular method has been selected, or *why* a particular structure is advantageous for solving a problem.

Even more rare is a comparison between various possible approaches, *e.g.* selecting between the various possible neural architectures. Thus a novice to the field is confronted with the question how to *select* a suitable method and how to *combine* methods with each other and with more classical approaches. In this tutorial we will discuss the basic theory and properties of fuzzy logic systems (FS), neural networks (NN), genetic algorithms (GA), and the concepts of ordinal optimization (OO), and how these methods may be combined in solving problems in practical applications. A typical example of such an application is *sensor fusion*. Although in this case the uncertainty modeling is usually handled by probability theory, soft computing methods are becoming increasingly more popular. The advantages of a soft-computing approach in comparison to standard approaches have already been recognized in the field of industrial control. In military missions the modeling of uncertainty is more relevant than in industrial applications. For this reason the use of fuzzy measures in decision theory may be relevant to this field. This can be illustrated in the example of sensor fusion. By attributing to each sensor in the sensor suite a unique weight, a general fuzzy measure can be defined in a self-consistent way. With this measure it is possible to take into account non-exhaustive and non-overlapping hypotheses, and also to reduce the cardinality of the space of alternatives, thereby avoiding the combinatorial explosion that is characteristic for *e.g.* the Dempster-Shafer theory of upper and lower probabilities. All soft-computing methods have in common that they somehow carry out a *nonlinear* optimization. As is well-known from optimization theory, heuristics plays an important role, since in practical situations exhaustive optimization is not feasible because of the NP-completeness of the problem. In contrast, soft optimization generates a *soft*, *i.e.* approximate, optimal solution to such problems in only polynomial time. Depending on the type of problem, this can *e.g.* be done by *ordinal* optimization, which is in practice often sufficient. In the final part of the review we will

address the issue of merging these soft-computing methods with each other and in relation to classical, analytical methods, so as to take the best of both worlds. Finally it should be noted that although the implementation of these methods can be done on normal digital computers, softcomputing algorithms are intrinsically parallel and thus may benefit from massively parallel, and distributed processing architectures.

2.3 How mission systems can benefit from softcomputing

Mission systems essentially have the internal structure of a control process. By breaking down the overall mission goal one obtains a collection of interacting subprocesses (*cf.* Figs. 1-2). In Fig. 3 the basic functionality of a mission system is outlined: 1) observe, 2) infer (process, interpret, control) and 3) act. Through this cycle the mission system observes the world and acts on it, which action in turn gives rise to changed observations. Two extra functions are added to the scheme of Fig.3, *viz.* 1) interaction with the user and 2) learning, a process aimed to improve the underlying control loops. The learning process may itself be seen as a mission process at a higher, more abstract autonomy level. There is no opportunity in this tutorial to discuss every aspect of how a mission system may benefit from softcomputing, but as an example we will concentrate on the observation process.
In recent years, numerous research papers have been published dealing with the application of multisensor data fusion, also referred to as distributed sensing high-level fusion, especially in the domain of military observations [1-6].

Historically the idea of sensor fusion is not new: As early as the sixties multi-radar trackers have been in use by the military for air traffic control and air defense. Multisensor data fusion seeks to combine information generated by multiple sensors to achieve goals that would be very hard or impossible to achieve with single sensors. From the point of view of efficiency, scheduling, accuracy, and redundancy it seems intuitively obvious that several sensors are 'better' than a single sensor.

Nowadays data fusion is a well-accepted method for making superior inferences in the field of industrial automation (*e.g.* for controlling a power plant, an oil refinery, a cement kiln (for a review on industrial applications, see *e.g.* [7,8]), or even a nuclear reactor [9,10], and for carrying out real-time pattern recognition in industry using a variety of sensors. Especially since the advent of softcomputing methods, such as fuzzy logic, data fusion has become a widely accepted successful fusion technology in industry. We note however that the success of such methods is primarily due to their ability to model human behavior or expertise in supervisory control. Sensor fusion also endeavors to mimic cognitive processes in humans by absorbing the signals of the human observation system, our five senses, from the real world and integrate, or 'fuse', these signal streams to build a coherent picture of our environment. As such, sensor fusion is concerned with lower abstraction levels, much higher information rates, and generally requires faster response than the data fusion used in supervisory control systems. This forms also the key problem in applying soft computing methods to this field: in controlling complex industrial or organizational processes at relatively long timescales, human operators have accumulated over the years ample experience. In contrast, there is only limited insight in the way a human being builds up an environmental picture, his awareness, from continuous multisensate observations.

Although sensor fusion is important to virtually all phenomenological sciences and engineering disciplines, most work until now has been done in the field of *defense* research. This can be understood as follows. In *analytical* approaches, *e.g.* in a physics experiment, the measured quantities or interactions are often so small that the experimental setup has to be designed in such a way as to make sure that the desired quantity or effect is optimally measurable. If the measured quantities are small, the experiment is repeated many times and ergodicity and statistics are used to arrive at average values with low relative standard deviation. Especially in case one tries to prove or disprove the correctness of a theoretical model, this often is a good approach. A final point to note here is that - apart from intrinsic physical real-time aspects - such experiments generally can be repeated many times and real-time constraints are not a bottleneck.

In engineering approaches the use of sensors is more *synthetic*, as illustrated *e.g.* in the field of factory automation. Here one deals with a well-

defined problem such as the quality control of products on a manufacturing line, *e.g.* checking the soldering joints on a PCB with an automated vision system. This problem certainly has real-time aspects, but the optimization can be done off-line and the observation circumstances, like in the physics experiment, can be optimized off-line, *e.g.* by testing the best combination of sensors, the proper cameras and illumination, and parallel operation with more than one quality control station if the speed of production requires so.

In military observations we deal with a situation that is far less comfortable than the situations described above: generally speaking it is necessary to assess in real time an often complex situation, that almost certainly is outside one's complete control. Handling such observations requires the modeling of *uncertainty*. Apart from the ordinary problems such as noise and clutter, radar and electro-optical sensors operate also under adverse weather and atmospheric conditions, without any possibility to improve the circumstances of the experiment, or to repeat the experiment, under strict real time constraints, with sometimes enormous consequences of false classification and even more serious penalties for non-detection. In addition, by the nature of the military métier, most interesting targets move at high speeds, try to avoid detection actively or passively, or mislead sensors by jamming or using decoys, and they are designed in such a way as to present a minimal scattering cross section to commonly used sensors and thus to be virtually invisible ('stealth').

Under such circumstances it is clear that doing military observations invariably implies the modeling of uncertainty. Classically this is often done by applying statistical methods, notably Bayes' theorem to formulate a (multi-) hypothesis testing problem. It is however also clear that statistical uncertainty can only model part of the uncertainty. The different measures of uncertainty are now well established in classical set theory, fuzzy set theory, probability theory, possibility theory and evidence theory [11].
The breakdown distinguishes *fuzziness*, or vagueness due to a lack of definite and sharp conceptual distinctions and *ambiguity*, the situation where we are dealing with one-to-many relationships in the information obtained from sensors, yielding *non-specificity* in the case that the data leaves two or more alternatives unspecified, or even *discord*, *i.e.* disagreement in choosing from among several alternatives.

Recently methods that explicitly deal with ambiguity and partially overlapping hypotheses such as Dempster Shafer theory [12,13] and the application of belief functions instead of probability densities have become popular. Of even more recent date is the application of general fuzzy measures [14]. The difficulty inherent to making accurate observations in military applications and the lack of measurement statistics are the prime motivations to improve single sensor observations by merging (partial) inferences/conclusions from one sensor with inferences from the other one.

3. SOFTCOMPUTING

In this section we will give an overview of the principles and basic concepts of four basic softcomputing techniques, *viz.* fuzzy systems (FS), neural networks (NN), genetic algorithms (GA), and ordinal optimization (OO) from the point of view of their potential use in mission systems. Emphasis is placed on the similarities of these four techniques stressing their ability to model complex nonlinear relationships in a multidimensional world. All these softcomputing methods can be applied in universal function approximation schemes (*e.g.* pattern recognition) and in nonlinear optimization.

Both pattern recognition and optimization (*e.g.* of resources, manpower, timescheduling, priorities) are vital to the success of a mission. It is therefore extremely important to thoroughly understand the possibilities of softcomputing methods. An added bonus of the nonlinearly inherent to softcomputing methods is that these systems in addition exhibit an increased robustness compared to classical methods. In studying the various softcomputing techniques such as FS, NN, GA, and OO, it is helpful to imagine a multi-dimensional input-output space, in which we consider a hypersurface with multiple maxima and minima (Fig. 4). FSs and NNs can approximate such a nonlinear input-output relation by combining either a small number of single rules and using very simple basisfunctions (FS), or by just using one type of function (NN).

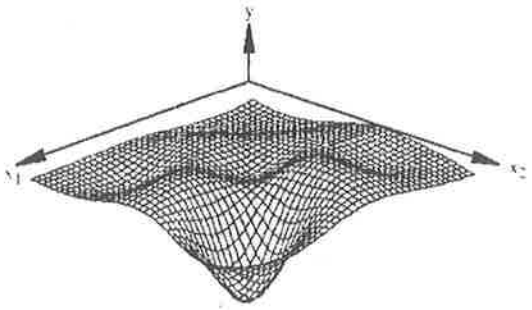

Figure 4 *A hypersurface in 3D space may represent a nonlinear input-output relation (x,y) or a complicated search space with many nearly degenerated optima.*

Basically a FS subdivides the (generally real-valued) variables of the input space in a small set of (overlapping) patches using so-called membership functions (MF), thus bringing down the number of states significantly. Next the nonlinear I/O relation is approximated by defining a rulebase for each of these input states. In a process called 'defuzzification' the fuzzy-valued output is converted to a real output value.

NNs approximate the desired relationship by using sigmoid or radial (Gaussian) basis functions that are weighted, shifted and otherwise modified by varying their synaptic weights in order to achieve the desired approximation. If we imagine the multidimensional space as *search* space, then we can view the output as a kind of performance or *"fitness"* function, measuring the error from some ideal functional behavior on *local* optimization.

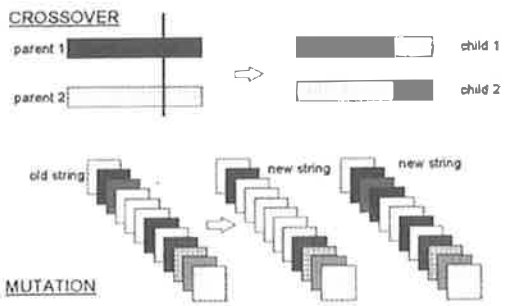

Figure 5 *The basic operations in an genetic algorithm: cross-over of two parent chromosomes forming two children (top) and mutation representing a (rare) stochastic process that randomly flips a gene on a single chromosome.*

GAs are ideally suited to search for a *global* optimum. The key concept is in a genetic optimization routine is the *representation* of the characteristics in a chromosome, a random reproduction process (*cross-over*, Fig. 5) and the *selection* of the 'best' chromosomes from the offspring to produce the next generation. GA has some distinct advantages compared to classical optimization schemes. Its prime advantage is that of fast convergence to near the global optimum. Near the global optimum the converge is however generally very slow. The global searching capabilities are not limited to smooth or simple convex structures: GAs do not require gradients to exist and just rely on a smart representation of the problem onto the chromosomes and a suitable fitness function.

A relatively young branch of softcomputing techniques is that of ordinal optimization. Like in GAs the key issue is here that one has to find a global optimum in a generally complex and large search space. There are two key differences from GAs, that gives OO a place of its own among softcomputing techniques:

1) OO aims to reduce NP-complete searches to polynomial heuristic searches; in other words OO tries to *formalize heuristics*.
2) OO explicitly allows for measurement uncertainty (stochastic observation errors) in the evaluation of the performance of a certain solution.

The basic idea behind various types of heuristic search is that of the 20/80 rule; of which many examples exist, *e.g.*: It takes 20% of time to achieve 80% performance; for the remaining 20% performance one has to spend 80% of his time. Another well-known example is the so-called "birthday paradox". The probability that 2 persons in an arbitrary group of 25 people have their birthday on the same day is > 0.5. Starting from such empirical observations Ho [15] developed the concepts of '*goal softening*' and '*ordinal optimization*'. OO can be intuitively understood by observing that it is generally much easier to determine 'order' instead of 'value' (*e.g.* it is easier to determine that A > B is true than it is to evaluate (A - B)). The concept of goal softening can be visualized by replacing the condition to be satisfied in the optimum by the much easier to fulfill condition of finding an optimum *close enough* to the true optimum.

In the following subsections we will now introduce the different softcomputing methods.

3.1 Fuzzy Logic

Fuzzy logic, fuzzy sets, and fuzzy measures are the basic concepts of FSs. Originally developed as a mathematical theory to model vague, imprecise notions, one of the first applications of FSs was in control (Fig. 6). At a first glance this may appear strange because there is nothing vague or imprecise in control engineering. In contrast one must first fuzzify the measured system inputs to be able to apply the theory of fuzzy sets. The reason for the success of FSs in the domain of control engineering is mainly the capability to absorb human (operator) experience in the form of "rules of thumb", and the capability to encapsulate all the essential knowledge to operate the fuzzy controller in a small set of fuzzy rules. An added advantage of these systems proved to be the robustness of such controllers: the nonlinear controller with its overlapping membership functions could accurately approximate the desired control surface (fuzzy controller = universal optimal approximator). The basic application of FL is through fuzzy (approximate) reasoning: fuzzy control may be viewed as an application of fuzzy approximate

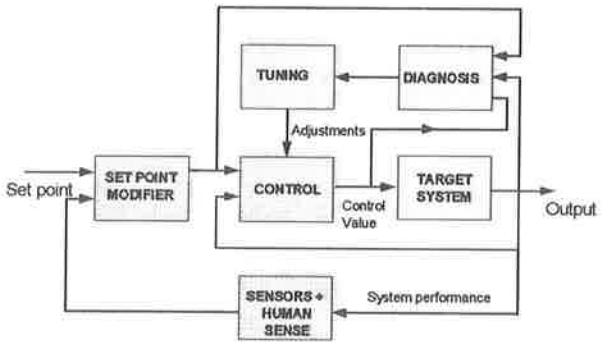

Figure 6 *Fuzzy control can be used in many different ways: Apart from the proper control process, fuzzy logic can be used to diagnose the performance of the controller and fine-tune it, to modify the setpoint and to merge human observations with sensor data.*

reasoning to control (Fig. 7). The key application areas of FSs in information science and engineering are: expert systems, control, feature extraction, and pattern recognition. Recently FS have also been developed in quite different disciplines, *e.g.* medical diagnosis, psychology, economy, management and operations research. In the following we will discuss the essential features of a FS, as illustrated in fuzzy control (FC) and see how FSs model the behavior of a skilled operator instead of modeling the system to be controlled. FC is more aimed at taking actions given certain conditions. The basic idea behind FC is that of *partitioning* the input variable space into a finite number of *overlapping* partitions and defining for each of these partitions a typical output state. The formulation of this definition is in the form of linguistic rules of the type:

"IF (x_1 is Large) AND (x_2 is Small) THEN (y is Negative_Small)"

Here x_1 and x_2 represent fuzzy input variables and y is a fuzzy output variable. Fuzzy variables take linguistic values such as "Negative Large", "Positive Small","Zero", and "Positive Medium". Each of these linguistic values is represented by a membership function μ, *i.e.* a function that is almost everywhere $\equiv$ 0 except for a finite interval, its so-called support, where the function takes positive values ≤ 1 (see Fig. 7). In order to apply this fuzzy rule base it is necessary to fuzzify the crisp (real-valued) input variables. The fuzzification process can be implemented in many ways, but basically it means that the degrees of membership (*i.e.* the values of the MFs $\mu_A(x)$ that are $\neq 0$) are associated with the rules having a rulepart of the form "IF (x is A)". In the case that more than one input needs to be considered we must determine the resulting activation of the rule from these degrees of membership. For this purpose a so-called t-norm operator must be selected. Examples of commonly used t-norms are the minimum operator MIN, used by Mamdani:

Rulestrength $w_{ij} = MIN_j(\mu_j(x_j))$ with j = 1,2.....k,

and the Product-operator $\prod$:

Rulestrength $w_{ij} = \prod_j(\mu_j(x_j)$ with j = 1,2.....k.

Finally after aggregating the inputs with the knowledge represented by the rules, the outputs can be determined from the rule strengths and the output membership functions by defuzzifying the

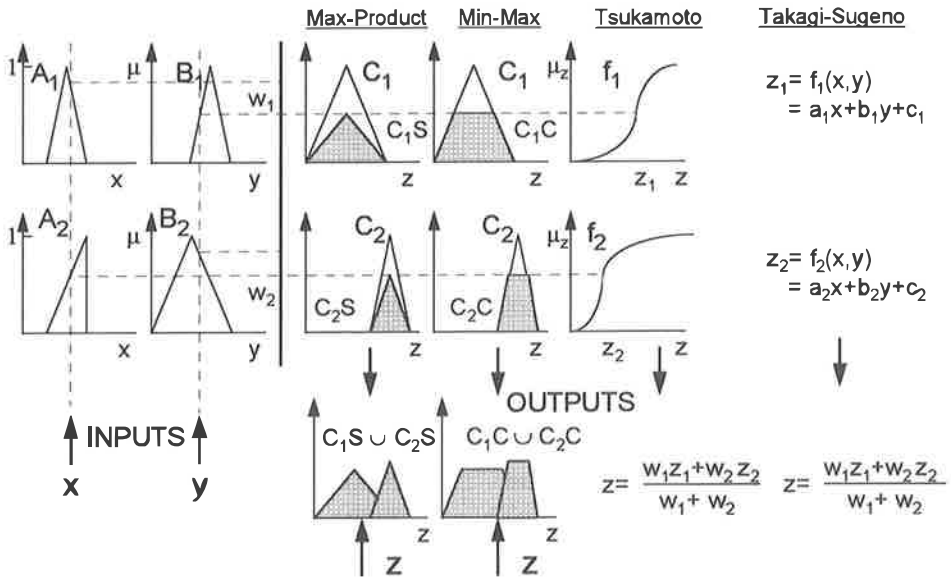

Figure 7 *Four different ways of fuzzy approximate reasoning can approximate different controllers z=f(x,y).*

outputs according to *e.g.* the center-of-gravity method as depicted in Fig.7 for the calculation of the output value z.

3.2 Neural Networks

Artificial neural networks are abstractions of the biological neural networks that constitute the brain. A biological neuron consists of dendrites, a cell body, and an axon (Fig. 8). The connections between the dendrites and the axons are called synapses. Electric pulses are generated in sensory cells (biological sensors) or in neighboring neurons and arrive on the synapses. The cell body operated on these inputs and fires a pulse, *i.e.* outputs an electrical charge on the axon, if the sum of all inputs exceeds a certain threshold. This basic mechanism is copied from the biological system to build an *artificial* neural network (henceforth abbreviated as NN), though excluding the time component: instead of a firing (repetitive) pulse, the output lasts as long as the weighted sum of the input exceeds the threshold. (Fig. 9).

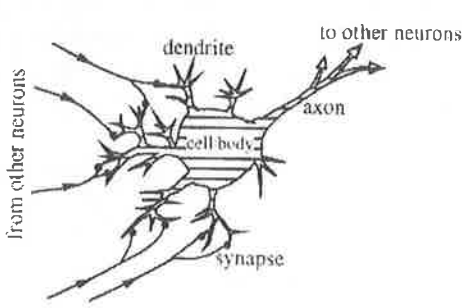

Figure 8 *A biological neuron: inputs from senses and other neurons end in the synapses. The cell body processes these signals and decides to fire, i.e. produce a series of electrical pulses. These are transferred to other neurons via the axon.*

Over time many NN architectures have been developed. In general a neural structure consists of a finite number of inputs connected to the input-layer of neurons and a finite number of output neurons. Between these lie one or more layers of so-called 'hidden' neurons. The idea is that the NN is trained by adapting the weights of the individual neurons so as to replicate the (input, output)-pairs in the training data set. This training can be achieved in two different ways *by supervised learning*, or alternatively, by *self organization*. In supervised learning a training set is available and the learning algorithm adjusts the neuron weights so as to match the desired input-output characteristics. The most frequently used learning algorithm in this

category is the backpropagation algorithm. In contrast, unsupervised learning is characterized by a mechanism that changes synaptic weights according to the input values of the network. The output characteristics are therefore determined by the network itself. Examples of self-organization are 1) Hebbian learning in which a weight w_{ij} of a neuron i and an input x_j is increased if the output y_i fires, by an amount $\Delta w_{ij} = \alpha y_i x_j$, where α represents the *learning rate* and 2) competitive learning, where all weights are modified of the unit that generates the largest output ('the winner takes all'). An example of such a self organizing competitive NN is Kohonen's self organizing feature map.

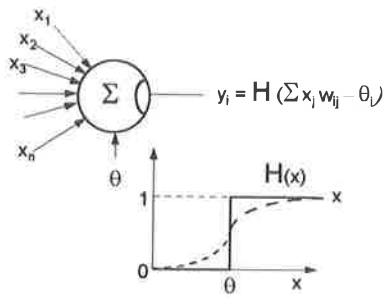

Figure 9 *An artificial neuron forms a linear combination of the inputs x and uses a nonlinear function if the input exceeds the threshold θ.*

One of the most popular learning algorithms is the *backpropagation* algorithm (BP). In the BP algorithm the difference between the desired and actual output of the neural network is backpropagated to modify the weights of all nodes involved in generating the difference. In this sense NN learning is equivalent with finding the global minimum (smallest error) of the error hyperplane in the space spanned by all the weights of the NN.

3.3 Genetic Algorithms

Genetic algorithms are searching for optimization procedures inspired by models of biological evolution. Key features of these so-called evolutionary computation are

1) representation: the coding of the problem under consideration onto chromosomes, *i.e.* strings of numbers (often bits) that code the properties;
2) the definition of an initial population of those chromosomes;
3) the application of biological-inspired random operations such as cross-over (Fig. 5) and, to an extent mutation to generate a new population from the original one;
4) the existence of a "fitness"-function that attributes to each chromosome a fitness value on the basis of which a selection is made of the 'best' chromosomes of the new generation. The 'not so fit' chromosomes are discarded from the population. This is a example of the principle of ordinal optimization to be discussed in the next section. In Fig. 10 a typical computation cycle is shown.

Genetic algorithms are relatively easy to code and converge, dependent on the 'goodness' of the chromosomes representation fairly quick in a number of cases to *near* the global optimum. The introduction of the mutation operator offers a way to escape out of local traps by the random creation of new chromosomes. The implementation of GA's can be very efficient due to the parallel nature of the algorithm: there is no preferential order in which chromosomes should be selected so cross-over can be applied on many chromosomes in parallel.

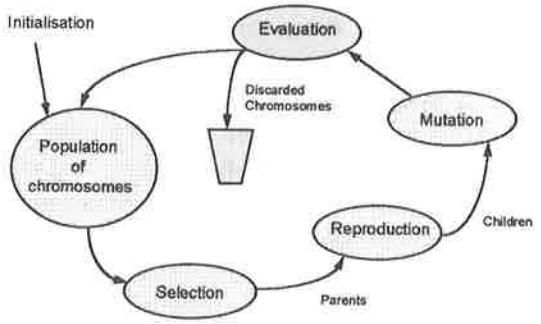

Figure 10 *A generation cycle in a GA: After evaluation the best chromosomes are added to the population and the rest discarded.*

A definite drawback of a GA is the statistical nature of the search with its inherently slow convergence $O(1/\sqrt{n})$, compared to deterministic methods. Therefore it is of great importance to include as much as possible a priori information in the representation of the chromosomes and into the fitness function. Several other implementations of GAs exist, such as evolutionary programs (EP). EPs are similar to GAs except that mutation is the *only* operator in EP to provide a new generation of

chromosomes, thus reflecting the influence of environment (boundary conditions, limitations) rather than that of the parents. In biology this difference is called *phenotype vs. genotype.*

3.4 Ordinal Optimization

Ordinal optimization plays a special role in softcomputing, because the method is not so much an alternative to other softcomputing methods but instead provides insight in the foundations of heuristics that is often used in taking decisions. At various stages of a mission decisions have to be taken in such a way as to optimize the overall goal of the mission. Without exaggeration it can be stated that it is extremely difficult for a human (and even more so for a machine) to really verify that such a decision is in fact optimal. Although the literature of optimization is huge, much of the mathematical analysis is concerned with continuous-differentiable functions, so that calculus-based methods can be used to find an optimum. In most of these methods it is necessary to be able to calculate a gradient or a derivative of a functional in order to find the optimum through a "steepest descent" method, often carried out in a successive approximation approach. This is in fact also true in the case of supervised learning in a NN. Although basically decisions in a neuron can be represented by a step- or Heaviside function of the form: if $\Sigma\, w_{ij}\, x_j \geq \theta_i$ output = +1 else 0 (see Fig. 9), where x_j= jth input synapse of neuron i and w_{ij} = weight associated with the i-th synapse, and θ_i = the threshold, in practice backpropagation requires a one-to-one correspondence between input and output and therefore no discontinuities. It is because of our limited mathematical toolbox (especially in the realm of discontinuous functions, difference equations and discrete optimization) that these calculus-based methods are desired and in act even required to find an optimum solution with analytical means. In order to force solvability, discrete problems are often 'smoothed' and it is hoped that the solution constructed in this way is a good-enough approximate solution of the real problem.

In practice a number of additional real-world difficulties are introduced by boundary conditions, complex geometries and other constraints. These can often be expressed as (in)equalities. Finally we are often confronted with insufficient data, so that some kind of uncertainty modeling has to be done. The introduction of uncertainty into the modeling presents a huge problem in practice, especially if the uncertainty is of a statistical nature (*e.g.* observation accuracy, or sensor noise) and one does not have the possibility (or time) to average over a sufficient number of observations, by which the observation errors can be reduced to an acceptable level. This is often the case in military observation systems. The key issue here is minimizing the estimation error, thus tightening the confidence level of the estimates and convergence to the 'true' optimum. There are a number of problems associated with the calculus-based optimization: 1) discrete-event dynamic systems cannot be treated this way. 2) in real systems it is often very difficult, if not impossible, to prove that the calculated (local) optimum is the desired *global* optimum.

Global minima are therefore only found by running an optimization procedure for multiple starting points (and proving that there are not too many relative optima), or by being able to show that the response surface is convex. In systems of practical interest, *e.g.* in NN this is more the exception than the rule: there we are confronted with an extremely high number of nearly degenerate minima in the energy surface, so that finding the global optimum is virtually impossible. Other examples are the identification of two adjacent frequencies in spectral analysis, combinatorial problems such as finding the shortest way connecting N sites (Traveling Salesman Problem) or scheduling problems such as minimizing production delays, or discrete parameter design. All these problems are difficult because of their enormous search space and the only way to find approximations of the optimum is by running many simulations. In addition they are NP-complete, *i.e.* the time needed to find a solution increases exponentially with the size of the problem. In order to find approximate solutions to such programs one is forced to use heuristics, rules of thumb, and ad hoc methods to achieve some kind of global optimization. Both the difficulty to take into account *all* local detail and the necessity to arrive at a solution in real time have induced a novel softcomputing approach. The justification of the ideas presented by Ho [15,16] is that humans manage reasonably well in making real-world decisions despite the NP-completeness of these problems and the insufficient knowledge.The following example is taken from Ref. [16] and illustrates the basic steps:

Consider, for example, that we have 200 ordered alternatives to evaluate. We blindly pick 12

alternatives out of these 200 and ask "what is the probability that among the 12 picked alternatives there is actually at least one alternative that is in the top-12?" The surprising answer is 0.5 ! If the number 12 is changed to 35, then the probability of finding a "good" alternative is close to one in the above statement. The implication of this is that even in the absence of any knowledge, one can dramatically reduce the number of alternatives one has to evaluate to narrow the search for "good" choices.

The central idea behind the previous statement is that of *ordinal optimization*: the idea that the relative order (instead of the cardinal value) of the performance of various alternatives in a general decision problem is quite robust with respect to estimation noise. The number of true top-r alternatives in the set of estimated top-r alternatives can be quite substantial even in the face of very large estimation errors in the performance value of the alternatives. In the above example or randomly picking alternatives, the equivalent estimation noise has infinite variance. If, on the other hand, the variance is not infinite, *i.e.*, there is some bias in favor of the actual good alternatives (however slight), then we can only improve the odds and help to narrow down the search. This is the core of the probabilistic justification of using heuristics in complex decision problems.

3.5 How to combine softcomputing with classical methods

All softcomputing methods have their proper application areas and it is impossible to even approximately describe how they can generally be combined. It is also clear that in contrast to the original introduction of these systems as competing and independent developments, one observes nowadays a convergence towards intelligent, hybrid systems [17], where NNs are combined with FS to achieve a certain level of adaptability. Also in more complex systems it is customary to distinguish a hierarchy of levels of autonomy and depending on the fuzziness of the goals and the uncertainty and ambiguity present in the observations, the resulting system is implemented as a mixture of classical and softcomputing approaches. It is therefore important to have a general idea of how methods could be combined in a meaningful way.

TABLE 1: A hierarchy of modeling techniques. It should be noted that all these methods can in principle be combined with each other, but that a higher one in the hierarchy (*i.e.* more analytical) is preferred over a lower one.

Model	Theory	Properties	Uncertainty	Speed
Analytical	Calculus	high precision continuous,global, numeric	no	off-line, fast
Rule-based	Fuzzy Logic	discrete, finite precision, local, structural, symbolic	yes, incomplete and ambiguous	on-line fast
Lookup table	Neural Network	learning, numeric, local, black box, unstructured	yes, noise can improve training	learning: off-line slow
Global optimization	Genetic algorithms	numeric, global, evaluation function	simulated annealing	slow
Ordinal Optimization	Ordinal Optimization	probabilistic, global and local	goal softening	fast

For the novice in this field it can be very bewildering to see how different authors solve the same type of problem with very different methods, each with its own merits, but it is almost impossible to compare the performance of these methods without a deep understanding of the underlying processes and actually re-doing the experiment. Although still many publications are devoted to one of the softcomputing techniques discussed here, without even giving a rationale why the selected method is preferred in this case, it should be pointed out that the so-called 'intelligent hybrid systems' gain rapidly in importance and aim at an integral approach of all techniques available. This is the line of thought that we also adhere. Without trying to review all combinations of NN, FS, GA, and OO that have been published we will try to give a guideline according to which the various softcomputing techniques might be applied: (see Table I).

Following the ordering of Table I, we start with conventional, often linearized modeling of the problem at hand. The advantage of such methods is that there is a well-developed body of mathematical methods available for this type of problems and even in the nonlinear case analytical expressions, conservation laws, and other relations can readily be derived. These methods have the great advantage that they are fast to evaluate (because analysis is essentially off-line), provide very accurate data and also provide insight in the underlying mechanisms: They allow us to parametrize environmental variables and allow us to explore their effect on the solution. In the absence of continuity and high precision data, or if the underlying problems are too complicated to model, solve analytically, or calculate numerically, it pays to approximate the 'exact' truth by trading in precision for speed of calculation. It has become only recently clear that precision can be very costly and that it may be much more efficient to use underlying structural knowledge of the type "IF X increases a bit THEN Y decreases strongly". It is in this context that fuzzy rule bases become important. At one hand they allow us to deal with 'difficult' details of classical analytical systems, at the other they provide us with a means to 'fuse' human operator experience with physical observations and mathematical models based on differential equations. The accuracy (*i.e.* input-resolution) diminishes in such systems, but the overall approximation of the observed system behavior increases. If structure is completely absent (at least the underlying structure cannot be recognized), but a sufficiently large 'training data set' *i.e.* (input, output) pairs is available, it is worthwhile to model the system at hand as a black box. This method has some drawbacks: the concept of a blackbox is not appealing to the scientist because one is never sure that a training set is of the correct size. Nevertheless NNs can provide a powerful method in extracting patterns in *e.g.* image recognition. Once the NNs are trained sufficiently (which is a slow process due to the statistical $O(1/\sqrt{n})$ performance), a NN provides a fast on-line lookup-table for connecting inputs and outputs. In addition it is theoretically possible to approximate *any* mapping $\Re^n \rightarrow \Re^m$ with arbitrary precision, provided enough training data are available. In the absence of numeric training data, it may be possible to find an optimal approximation of a good representation and a suitable evaluation function. It should be noted that good representation is always an essential step in finding a solution to a problem. The advantage of a GA approach is that discrete optimization is possible. Unlike in the case of *e.g.* NNs there is no need for gradients to exist and optimization is only on the basis of a suitable evaluation function. Complex boundary conditions may be taken into account and convergence to near the global optimum is generally fast. Finally if the search space for the optimal solution becomes prohibitively large (as is case of large NP complex problems), ordinal optimization (OO) may provide a solution path. Ordinal optimization tries to formalize common heuristics by softening the goal and finding the optimum within acceptable accuracy and probability limits through the use of even the slightest global information on the topology and structure of the search space, without suffering from the slow performance imposed by the $O(1/\sqrt{n})$ limit.

Each method has its particular strengths and weaknesses. Depending on the availability of training data and a priori structured knowledge a model may be developed that approximates the observed behavior sufficiently accurate. The penalty for not using the available a priori information is that the system has to discover these structures itself at the price of slow convergence (law of large numbers). It is well-worth considering to decrease the required accuracy because this

generally results in a substantial reduction of computation time. Finally it should be noted that, like in all modeling problems, representation often is the key to solving a problem: without the proper algebra it is often virtually impossible to find a solution, whereas with a suitable representation the solution may even look trivial.

4. CONCLUSION

In this review we have discussed a number of basic softcomputing techniques and have indicated how they may be combined to form intelligent hybrid systems suitable for optimizing missions. Key aspects of these systems such as their capability to handle uncertainty, their adaptability, and their robustness make such systems of great interest for future mission systems. The latter are expected to evolve from human decision support systems towards unmanned, highly autonomous robotics systems that should be capable of reaching their goal, even in adverse circumstances and with scarce resources. The increasing complexity of these tasks and the limited capacity to include more computing power into the mission platform (ignoring the relevance of including more computing power because of the NP-completeness), it is of great importance that mission systems benefit from modern developments in softcomputing, thereby increasing their robustness to cope with new and unexpected situations. Although progress in many other technologies will certainly be needed to achieve the ambitious goals set for future (un)manned missions, we are convinced that softcomputing technology is one of the key technologies to achieve this aim.

5. ACKNOWLEDGMENT

The author wishes to thank Ms. Ellen J.M. Tieland for her help in preparing the manuscript.

6. REFERENCES

[1] J. Llinas and E. Waltz, Multisensor Data Fusion, ArtechHouse, Norwood, MA.,1990.

[2] D. Hall and R. Linn, "A taxonomy of multisensor data fusion techniques", Proc. 1990 Joint data fusion symposium, vol 1, 593-610, 1990.

[3] C.B. Weaver, Ed., "Sensor Fusion", Proc. of SPIE, vol 931, Orlando, FL, 1988.

[4] P.S.Schenker, Ed., "Sensor Fusion, Spatial reasoning and scene interpretation", Proc. of SPIE, vol 1003, 1988.

[5] C.B. Weaver, "Sensor Fusion II", Proc. of SPIE, vol 1100, Orlando, FL, 1989.

[6] P.S.Schenker, "Sensor Fusion II, human and machine strategies", Proc. of SPIE, vol 1198, Philadelphia, PA, 1989.

[7] A.J. van der Wal, "Application of fuzzy logic control in industry", Fuzzy Sets Syst 74, 33-41, 1995.

[8] A.J. van der Wal, "Fuzzy Logic: Foundations and Industrial Applications", ed. D. Ruan, Kluwer, Chapter 14, 275-311, 1996.

[9] D.Ruan, Z. Liu, L. Van den Durpel, P.D'hondt, and A.J. van der Wal, "Progress of Fuzzy Logic control applications for the Belgian Nuclear reactor BR1", Proceedings EUFIT '96, Aachen vol2, 1237-1241, 1996.

[10] A.J. van der Wal and D. Ruan, Proc. JCIS 97, Research Triangle Park 136, 1997.

[11] G. J. Klir and B. Yuan, Fuzzy sets and fuzzy logic, Prentice Hall, New Jersey, 1995.

[12] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping", Ann. Math. Statistics, 38, 325-339,1967.

[13] G. Shafer, A mathematical theory of evidence, Princeton University Press, Princeton, 1976.

[14] K. Leszczynski, P. Penczek, and W. Grochulskki, "Sugeno's fuzzy measures and fuzzy clustering", Fuzzy Sets Syst., vol 15 147-158, 1985.

[15] Y.-C. Ho and M.E. Larson, "Ordinal optimization approach to rare event probability problems", J. DEDS 5, 281-301, 1995.

[16] Y.-C. Ho, "Heuristics, rules of thumb, and the 80/20 proposition", IEEE Trans. on Automatic Control 39 (5), 1025-1027, 1994.

[17] For an up to date review: "Intelligent hybrid systems: fuzzy logic, neural networks, and genetic algorithms", Ed. D. Ruan, Kluwer Academic, Boston, 1997.