Intelligent Automated Control of Life Support Systems Using Proportional Representations

Annie S. Wu and Ivan I. Garibay

***Abstract*—Effective automatic control of Advanced Life Support Systems (ALSS) is a crucial component of space exploration. An ALSS is a coupled dynamical system which can be extremely sensitive and difficult to predict. As a result, such systems can be difficult to control using deliberative and deterministic methods. We investigate the performance of two machine learning algorithms, a genetic algorithm (GA) and a stochastic hill-climber (SH), on the problem of learning how to control an ALSS, and compare the impact of two different types of problem representations on the performance of both algorithms. We perform experiments on three ALSS optimization problems using five strategies with multiple variations of a proportional representation for a total of 120 experiments. Results indicate that although a proportional representation can effectively boost GA performance, it does not necessarily have the same effect on other algorithms such as SH. Results also support previous conclusions [23] that multivector control strategies are an effective method for control of coupled dynamical systems.**

***Index Terms*—Genetic algorithm (GA), life support system control, resource allocation, proportional genetic algorithm, gene expression, proportional representation, stochastic hill-climbing (SH).**

I. Introduction

As TECHNOLOGY advances in the space industry, the opportunity and need for long term extraterrestrial missions will increase. Such missions include both missions in space and extraterrestrial planetary habitats. In all cases, a necessary element for the success and productivity of a mission is an effective Advanced Life Support System (ALSS). A closed habitat such as a planetary or space habitat is composed of multiple components, e.g., crew, air, water, plants, climate, stores, and waste, all of which interact in a complex manner. An ALSS controls the necessary processes that regenerate basic life support products such as air, water, and food within a habitat. As a result, it minimizes the need to store large amounts of consumable products and minimizes the need for frequent resupply of such products.

Energy is required to produce food, regenerate air, purify water, and process waste. A closed habitat may be limited to a pre-defined amount of stored energy or may be exposed to a reliable, fixed energy source such as the sun or regular air currents. The task of an ALSS is to distribute available energy resources among the required tasks as efficiently as possible to maximize the productivity and lifetime of a habitat. The tight coupling of the system components and inherent unpredictability of some of them, e.g., the crew and climate, make this a complex task. Nevertheless, many of the decisions and tasks of an ALSS are routine and the development of automated or semiautomated systems can significantly reduce human effort on tedious monitoring tasks.

An ALSS is an example of a class of systems called coupled dynamical systems. Such systems consist of deterministic subsystems whose behaviors are easy to predict in isolation, but the behavior of the complete system is difficult to predict and thus difficult to control. The majority of work on ALSS control to date has focused on modeling and control of individual subsystems within a life support system, e.g., the water recovery system or the air revitalization system. Control algorithms are typically reactive to changing conditions but deliberative and tailored to a specific module or set of modules [5], [11], [35]. While deliberative control systems may suffice under static conditions, some element of learning or adaptability is likely to be necessary in real world scenarios. Kortenkamp *et al.* [23] present the first study that successfully uses machine learning (ML) techniques to learn how to control an ALSS as a whole.

In this paper, we extend previous work [23] by performing a detailed comparison of genetic algorithm (GA) and stochastic hill-climbing (SH) approaches to ALSS control. Evaluation of candidate solutions is performed on a newer and more complete ALSS model than was previously available. As a result, successful results lend even stronger support to the viability of ML techniques in the development of autonomous ALSS controls. For both algorithms, problem representation determines the shape of the landscape which determines the solutions that are reachable from any particular solution. We examine the performance of these algorithms using two significantly different types of problem representations.

II. Background

A. ALSS Simulator

We have implemented an ALSS simulator that models the basic processes occurring in the Bioregenerative Planetary Life Support System Test Complex (BIO-Plex) developed by NASA Johnson Space Center [4], [23], [38]. This simulator is used to evaluate the candidate control strategies generated by our algorithm. Because of time and computational constraints, the ALSS simulator is a simplified model of the actual BIO-Plex simulator. We expect our learning algorithm to be easily linkable to the BIO-Plex simulator, if desired.

In each time step, the ALSS accepts as input the vector of control parameters $\vec{c}^{\,t}$ listed in Table I. A continuous value from

Manuscript received January 8, 2003; revised December 16, 2003. This paper was recommended by Associate Editor A. Kuba.

The authors are with the School of Computer Science, University of Central Florida, Orlando, FL 32816-2362 USA (e-mail: aswu@cs.ucf.edu; igaribay@cs.ucf.edu).

Digital Object Identifier 10.1109/TSMCB.2004.824522

TABLE I
VECTOR OF CONTROL PARAMETERS $\vec{c}^{\,t}$ OF ALSS SIMULATOR

Parameter	Description	Values	Type
$c^t_{energy.to.air}$	Energy alloc. to air proc.	$0 \leq x \leq 5$	Real
$c^t_{energy.to.water}$	Energy alloc. to water proc.	$0 \leq x \leq 5$	Real
$c^t_{energy.to.food}$	Energy alloc. to food prod.	$0 \leq x \leq 5$	Real
$c^t_{water.to.crew}$	Clean water alloc. to crew	$0 \leq x \leq 5$	Real
$c^t_{water.to.crops}$	Clean water alloc. to crops	$0 \leq x \leq 5$	Real
$c^t_{activity.level}$	Activity level of crew	$0, 1, 2, 3$	Int.
$c^t_{store.water}$	Use from water store	$0, 1$	Int.
$c^t_{store.air}$	Use from air store	$0, 1$	Int.
$c^t_{store.food}$	Use from food store	$0, 1$	Int.

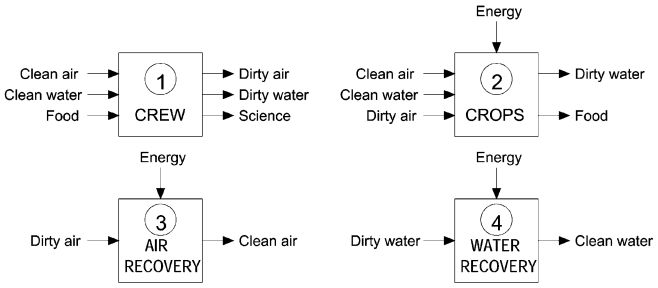

Fig. 1. Basic transitions in the ALSS simulator.

zero to five indicates energy allocation to water, air, and food processing, as well as water allocation to crew and crops. Larger values indicate more energy and water. There are four discrete activity levels for the crew: sleep (0), low activity (1), moderate activity (2), and high activity (3). A binary value each indicates whether or not water, air, and food stores will be used. At the end of a simulator run, the ALSS outputs measures of mission duration in simulation time steps or ticks (of approximately one hour) and mission productivity in units of "science" for that particular run. A simulator run ends when resources are exhausted and the environment becomes unable to support human life. The diagrams in Fig. 1 describe the basic processes and transformations that take place in the ALSS simulator in each time step.

1) The crew process consumes clean air, clean water, and food from the environment and produces dirty air, dirty water and *science*. The consumption and production rates are determined by the *activity level* input parameter. The higher the activity level, the greater the consumption of resources and production of resulting products. An internal simulation parameter, *crew status*, represents the health of the crew. A lack of resources for multiple consecutive time steps results in crew extinction and the end of the simulation.
2) The crops process consumes clean air, clean water, dirty air, and energy and produces dirty water and food. Production of food depends on the *crop status* and on the availability of the consumed products. The *crop status* represents the current crop health, and decreases if there is insufficient air, water, or energy. Lack of consumables over multiple consecutive time steps results in crop expiration and the end of food production.
3) The air recovery process consumes energy and dirty air and produces clean air. The amount of clear air produced depends on the available energy. Lack of energy for three consecutive time steps shuts down this process (*shut down period*). If energy is restored, the process requires two consecutive time steps with adequate energy supply, (*warm up period*) to restart the production of clean air.
4) The water recovery process consumes energy and dirty water and produces clean water. This process is analogous to the air recovery system, but with warm up and shut down periods of ten and five simulation time steps, respectively.

A detailed mathematical description of the simulator is given in the Appendix.

B. Related Work

Automated and semi-automated control of ALSS attempts to relieve the workload of human operators and crew members, moving them from a "vigilant monitoring" role to a "supervisory monitoring" role and allowing them more time to perform the more complex tasks of a mission [35]. Although live subject experiments have been and continue to be performed, mathematical modeling and simulation studies are also prevalent due to their significantly lower cost and high controllability. Such simulations use mathematical models to approximate and simulate the interactions that are expected to occur for any given set of input conditions [15], [16], [19], [22]. Studies of autonomous or semi-autonomous control of various individual subsystems of an ALSS include a three-tier hierarchical control of water recovery and air regeneration systems [35], a market-auction-based approach to managing power allocation and surges for a partially modeled ALSS [11], and various approaches for control of environmental management systems [15], [16], [35].

Kortenkamp *et al.* [23] are the first to evaluate ML methods for learning effective ALSS energy allocations using an early version of the BIO-Plex simulation. They find both the GA and reinforcement learning to be viable choices for maximizing either or both the mission duration and productivity of a single simulator run. Two different GA implementations are tested. In the first GA method, each GA individual specifies a single set of control parameters. The fitness of an individual is based on the duration or productivity of a run that uses that individual's parameter setting in every time step. A fitness function that optimizes for mission duration produces an optimal run of 31 ticks with 148 units of science. A fitness function that optimizes for productivity generates a maximum of 184 units of science with a mission length of 26 ticks. By comparison, reinforcement learning performs slightly better than the GA in terms of mission duration and about twice as well in terms of productivity. The second GA approach allows each GA individual to specify multiple n sets of parameter values. These n sets of values are used in n consecutive time steps in a simulation run. The fitness of an individual is based on the duration and productivity of a simulation over the n time steps it takes to use all parameter sets specified by that individual. The next individual to be evaluated executes its parameter values in the next n time steps of the same simulation. This multistep GA produces a significant improvement in performance, reaching a mission duration of 702 ticks (when optimizing for ticks) or a productivity of over 1800 units of science (when optimizing for science). Increasing

the number of sets of parameters n and the target fitness values over multiple runs appears to improve the quality of solutions generated.

Within the area of resource allocation, GAs have been successfully applied to a number of different types of problems. For example, in wireless communications, GAs are able to learn effective and adaptive frequency allocation strategies [36], [37], [39]. Military applications have used GAs to allocate resources such as aircraft [1], weapons [30], and sorties [31]. Lau and Tsang [25] apply a hybrid algorithm combining a GA with Guided Local Search to solve the Generalized Assignment Problem. Callaghan *et al.* [8] use a GA to optimize land allocation. Baglioni *et al.* [3] describe an evolutionary approach to financial asset allocation. Cousins *et al.* [10] describe a hybrid GA method for memory allocation in high-performance computing systems. Papavassiliou *et al.* [32], [33] examine the integration of agents and GAs for network management. The problem of optimizing resource allocation is closely related to scheduling and routing problems that have been widely studied in the GA field [6], [27], [29].

III. Experimental Details

A. Algorithm Descriptions

1) GA: A GA [20], [18] is a learning algorithm based on principles from natural selection and genetic reproduction. Key features that distinguish GAs from other algorithms include:

1) A *population* of *individuals* where each individual represents a potential solution to the problem to be solved;
2) A *fitness function* that evaluates the utility of each individual as a solution;
3) A *selection function* that selects individuals or "parents" for reproduction based on their fitness; and
4) Idealized *genetic operators* that create new individuals from selected parents without destroying all information from the parent individuals.

Fig. 2 shows the basic steps of a GA. The initial population may be initialized randomly or with user-defined individuals. The GA then iterates through an evaluate-select-reproduce cycle until a stopping condition is satisfied. Table II gives the GA parameter settings used in this work. Values were experimentally selected to provide good performance.

2) SH: Hill-climbing [2], [12] is a simple optimization algorithm that has proven to be a competitive alternative to GAs. If we visualize an optimization problem as a landscape on which each point corresponds to a solution and its height corresponds to the fitness of the solution, a hill-climbing algorithm searches for peaks by repeatedly moving to adjacent points of higher fitness. SH is a probabilistic variant of hill-climbing in which the search for adjacent points is stochastic and not deterministic. In the GA community, SH is a widely accepted baseline comparison of GA effectiveness for given problems [21], [28]. The SH algorithms used here adopt the *problem representation*, the *fitness function* and the *mutation operator* of their corresponding GA variant, but they use the following search procedure.

1) Choose one individual at random. Call this individual the *best-individual*.

```
procedure GA
   {
   initialize population;
   while termination condition not satisfied do
      {
      evaluate current population;
      select parents;
      apply genetic operators to
            parents to create offspring;
      set current population equal to
            the new offspring population;
      }
   }
```

Fig. 2. Basic steps of a typical GA.

TABLE II
Constant Parameter Settings for All GA Runs

Population size	100
Generations	500
Selection method	tournament
Crossover type	two-point (fixed length) homologous (variable length)
Crossover rate	1.0
Mutation rate	0.005

2) If the termination condition is satisfied, stop and return *best-individual*.
3) Mutate *best-individual*. If the mutation produces an individual with higher fitness, then set *best-individual* to the mutated individual. Go to step 2.

The termination condition is set to 50 000 fitness evaluations—equal to the number of evaluations in each GA run. Mutation rate was experimentally optimized and set to 0.05.

B. Fitness Details

1) The Optimization Problem: An ALSS simulation (Definition 1)[1] is a coupled dynamical system with a set of internal simulation states (Definition 3), a control strategy (Definition 5), and a transition function (Definition 6). The next step of a simulation (Definition 7) is calculated deterministically using the transition equations that compute the next simulation state based on the current state and the current control variables provided by the control strategy. The ML problem is to find a controller or, equivalently, to find a set of control variables to be used in each simulation step that optimizes one or more outcomes of a simulation. A simulation ends when its environment is no longer able to support human life (Definition 9).

In this paper, we study three different optimization problems.

1) Finding a control strategy that maximizes mission productivity (Definition 11).
2) Finding a control strategy that maximizes mission duration (Definition 12).
3) Finding a control strategy that optimizes both mission productivity and duration (Definition 13).

2) The Control Strategies: The *control strategy* (Definition 5) throughout this paper uses n vectors of control parameters circularly within a simulation. Therefore, the optimization problem for the various algorithms we test—GA and SH with

[1]A formal definition of this system is given in the Appendix.

both binary and proportional representations—is restricted to learning the appropriate n vectors of control parameters that, when circulated in a simulation, produce a maximized outcome of mission productivity, duration, or both. We call this control strategy Γ_n. We perform experiments with $n = 1, 3, 5, 7$, and 9 vectors of parameters and refer to these instantiations of the control strategy as $\Gamma_1, \Gamma_3, \Gamma_5, \Gamma_7, \Gamma_9$, respectively. Each individual or candidate solution is a set of n vectors of control parameters.

3) The Fitness Measures: The fitness of an individual is determined by running an ALSS simulation using an individual's n vectors and control strategy Γ_n. We test three fitness measures in our experiments.

1) *Optimizing mission productivity:* the total amount of "science" produced by the crew (Definition 14).
2) *Optimizing mission duration:* the number of time steps at which the simulation ends (Definition 15).
3) *Optimizing both:* a weighted addition of the previous two measures (Definition 16).

C. Problem Representation Details

How a problem is represented in an ML algorithm determines what can be expressed and how the solution space is connected which in turn determines the concepts that can or cannot be learned. How information is represented in a learning algorithm can affect an algorithm's expressivity, efficiency, and readability [34]. We compare the performance of a GA and SH on two drastically different problem representations: binary representation and proportional representation.

1) Binary Representation: A typical binary representation consists of a field of bits for each encoded parameter value. For our problem, each vector of control parameters is a 25-bit string. We use four bits to represent each of the five real values shown in Table I. Four bits provide a resolution of 16 possible values which are scaled to a real value between zero and five. A two-bit binary encoding indicates "activity level". A single bit encodes whether or not to use resources from the stores. With this representation, each individual is a $25 \times n$ bit string formed from the concatenation of n 25-bit strings, each representing one vector of parameters. Fig. 3 shows an example individual for $n = 1$ and Table III gives the values that it encodes. The GA uses two-point crossover and bit-flip mutation. The SH uses bit-flip mutation.

The strengths of binary representations are that they are very space-efficient and that they are logical for humans to interpret. Weaknesses include brittleness (missing, extra, or misplaced bits can severely change encoded values) and positional biases typically found in order-based encodings [14].

2) Proportional Representation: The proportional representation attempts to address the weaknesses of order-based encodings by eliminating the notion of order altogether. This representation was initially developed for a GA [41], but can also work with other algorithms such as SH. Encoded information depends solely on what does and does not exist on an individual and not on the order in which it is present. As a result, the order of the encoding is free to evolve in response to factors other than the expressed solution, for example, in response to the identification and formation of building blocks.

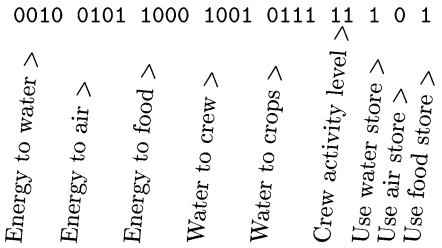

Fig. 3. A binary encoded individual and its corresponding encoded values.

TABLE III
VALUES ENCODED BY EXAMPLE INDIVIDUAL FROM FIG. 3

Parameter	Expressed Value
Energy to water	$2/15 \times 5 = 0.67$
Energy to air	$5/15 \times 5 = 1.67$
Energy to food	$8/15 \times 5 = 2.67$
Water to crew	$9/15 \times 5 = 3$
Water to crops	$7/15 \times 5 = 2.3$
Crew activity level	3
Use water store	1 (Yes)
Use air store	0 (No)
Use food store	1 (Yes)

The proportional representation uses a linear genome with a multicharacter alphabet. One or more unique characters are assigned to each parameter or component of a solution. The value of a parameter is determined from the relative proportions of the assigned characters of that parameter. Thus, characters that exist are "expressed" and, consequently, interact with other expressed characters. Characters that do not exist are "not expressed" and do not participate in the interactions of the expressed characters.

For example, in the ALSS problem, we are searching for n vectors $\vec{V}$ of $a = 9$ parameter values, $V_i, i = 0, \ldots, a-1$. The range of these values is given in Table I and will be denoted by $V_{i,\min}$ and $V_{i,\max}$. We assign a "positive" character, p_char, and a "negative" character, n_char, to each parameter, as shown in Table IV.

The number of positive and negative characters on an individual are used to calculate the proportion of positive characters, pct, as follows:

$$\mathrm{pct}(V_i) = \frac{p\_\mathrm{char}(V_i)}{p\_\mathrm{char}(V_i) + n\_\mathrm{char}(V_i)}$$

where $i = 0, \ldots, a-1$ and $0.0 \leq \mathrm{pct}(V_i) \leq 1.0$. The value of each parameter is then calculated by the equation

$$P_{\mathrm{PGA}}(V_i) = V_{i,\min} + \mathrm{pct}(V_i) \times (V_{i,\max} - V_{i,\min}).$$

The calculated value may be rounded, e.g., for integer parameters. A typical single-vector individual $(n = 1)$ of length 40 such as the one shown in Fig. 4 encodes the values shown in Table IV. As the expressed values are completely independent of the arrangement of characters, the individual in Fig. 5 also encodes the values shown in Table IV.

A fixed length GA uses two-point crossover. A variable-length GA uses homologous crossover [7] which randomly selects a crossover point on the first parent, then finds the region of highest homology (similarity) on the second parent. Crossover occurs within the region of homology. Mutation for both a GA and SH randomly switches one character

TABLE IV
PGA CHARACTER ASSIGNMENTS AND ALLOCATION OF RESOURCES AS SPECIFIED BY EXAMPLE INDIVIDUALS FROM FIGS. 4 AND 5

Parameter value V	$p\_char(V)$	$n\_char(V)$	V_{min}	V_{max}	Number of $p\_char(V)$	Number of $n\_char(V)$	$pct(V)$	$P_{PGA}(V)$	Expressed value
Energy to water	A	a	0	5	6 A's	1 a's	$6/(6+1)$	4.29	4.29
Energy to air	B	b	0	5	2 B's	3 b's	$2/(2+3)$	2.0	2.0
Energy to food	C	c	0	5	2 C's	2 c's	$2/(2+2)$	2.5	2.5
Water to crew	D	d	0	5	1 D's	0 d's	$1/(1+0)$	5.0	5.0
Water to crops	E	e	0	5	1 E's	1 e's	$1/(1+1)$	2.5	2.5
Crew activity level	F	f	1	3	3 F's	4 f's	$3/(3+4)$	1.86	2
Use water store	G	g	0	1	5 G's	2 g's	$5/(5+2)$	0.71	1
Use air store	H	h	0	1	0 H's	4 h's	$0/(0+4)$	0.0	0
Use food store	I	i	0	1	2 I's	1 i's	$2/(2+1)$	0.67	1

```
AccBfFhIiAAGgDEGGGaCFbIbABhFGgfCbhhAffeA
```

Fig. 4. An example proportional representation individual of length 40 with a single vector ($n = 1$).

```
AAAAAAaBBbbbCCccDEeFFFfffffGGGGGgghhhhIIi
```

Fig. 5. Another example proportional representation individual of length 40 with a single vector ($n = 1$). Encodes equivalent solution as individual from Fig. 4.

to another character in the alphabet. All characters are equally likely.

3) Comparing Binary and Proportional Representations: Although the proportional representation eliminates positional biases, it is clearly a less compact encoding than binary representation. Binary individuals are strings over a binary alphabet while proportional individuals are strings over a higher-arity alphabet. As a result, binary and proportional individuals of identical length will likely encode solutions spaces of different sizes. In order to fairly compare the two representations we need to find a relationship among their individual lengths and alphabet sizes to ensure that both algorithms target solution spaces of comparable complexity[2].

A binary individual of length l_{bin} over an alphabet with cardinality n_{bin} encodes $(n_{\text{bin}})^{l_{\text{bin}}}$ different binary strings which represent $(n_{\text{bin}})^{l_{\text{bin}}}$ different solutions. The size of the search space and the solution space are equal for binary representation. A proportional individual of length l_{pro} over an alphabet with cardinality n_{pro} encodes $(n_{\text{pro}})^{l_{\text{pro}}}$ different multicharacter strings. These strings are the search space of this proportional representation and map to a solution space of $\binom{n_{\text{pro}}+l_{\text{pro}}-1}{l_{\text{pro}}}$ different solutions. Thus

$$(n_{\text{bin}})^{l_{\text{bin}}} = \binom{n_{\text{pro}}+l_{\text{pro}}-1}{l_{\text{pro}}} \tag{1}$$

must hold for a binary and proportional representation to encode solution spaces of the same size. Consequently, a proportional encoding length will typically be longer than an equivalent binary length. A complete discussion of this comparison can be found in [41].

A binary-encoded individual requires a total length of 25 bits to represent a single vector of values and $25 \times n$ bits to represent n vectors. We can use (1) to calculate the corresponding proportional representation lengths for each value in a vector, then sum over all values to obtain the total equivalent proportional representation length. For example, $n_{\text{bin}} = 2$ and $l_{\text{bin}} = 4$ for the binary representation of the first parameter—energy allocated to water processing. The proportional representation uses two characters to represent any single parameter, thus $n_{\text{pro}} = 2$. Using (1), we calculate $l_{\text{pro}} = 15$ for the first parameter. We can compute the remaining equivalent parameter lengths for the proportional representation in the same way, to obtain a total equivalent proportional representation length of $l_{\text{pro}}(1) = 15+15+15+15+15+2+1+1+1 = 80$ for a single vector. The required proportional representation length for n vectors is $l_{\text{pro}}(n) = l_{\text{pro}}(1) \times n = 80 \times n$.

We compare binary representation to three proportional representations of differing lengths, Len, and consequently, differing resolutions.

1) *PSame*: Fixed-length proportional representation in which the length of an individual is the *same* as the length of a binary representation individual, $\text{Len} = l_{\text{bin}}$. This length is 25 for a single vector of parameters and $25 \times n$ for n vectors of parameters. According to (1), *PSame* is severely penalized in terms of available resolution due to its restricted genome length.
2) *PHalf*: Proportional representation with a fixed-length of *one-half* the equivalent length, $\text{Len} = l_{\text{pro}}/2$. This length is 40 for a single-vector case and $40 \times n$ for n vectors. *PHalf* is still penalized in terms of available resolution; however, less so than *PSame*.
3) *PMax*: Proportional representation with the equivalent length $\text{Len} = l_{\text{pro}}$. This length is 80 for the single-vector case and of $80 \times n$ for the n-vector case. To save space, we allow the GA to evolve variable-length individuals with a maximum length of Len. No parsimony pressure is applied. The SH uses fixed-length individuals of length Len.

The proportional representation typically requires two characters for each parameter, thus the total number of characters required to represent n vectors of nine parameters should be $2 \times 9 \times n$. As the precision of similar parameters is likely to be comparable, it is reasonable to use the same set of negative characters for all corresponding vector components. The resulting PGA requires only $9 \times (n+1)$ characters to represent n vectors of nine parameters. For example, consider the case of $n = 3$ vectors. The first parameter of each vector represents the value

[2]We define the *search space* to be the space of all encodings and the *solution space* to be the space of all solutions.

of *energy to water*. Let all three values share the same negative character: a; but have unique positive characters: A for the value in the first vector, J for the second and S for the third. Similarly, the second parameter (representing the value of *energy to air*) of all three vectors share the same negative character: b; but have unique positive characters: B, K and T, respectively. We use this sharing strategy for all the experiments with proportional representation.

IV. Experimental Results

We compare the performance of a GA and SH on finding control parameter values for the three optimization problems described in Section III-B.1 using the five control strategies described in Section III-B.2 for the ALSS simulation. In each set of experiments, we test eight algorithms: GA and SH each using four different representations, *Binary, PSame, PHalf*, and *PMax*, as described in Section III-B.3. Each experiment is run 40 times and the results averaged over all runs. Each run stops at 50 000 fitness evaluations. A fitness evaluation consists of one execution of the ALSS simulation using the control parameters to be evaluated. When the simulation terminates, a fitness value is returned as described in Section III-B.3.

Fig. 6 summarizes the results from these 120 experiments. For each experiment, we plot the fitness of the best solution found, averaged over 40 runs with 95% confidence intervals. The results are organized into three plots by optimization problem: (A) optimizing mission productivity, (B) optimizing mission duration, and (C) optimization of mission productivity and duration. The y-axes indicate the fitness of the solutions found: in units of science for (A), in simulation time steps or ticks for (B), and as a weighted combination of both for (C). The x-axis indicates the algorithms used. On this axis, from left to right, we show a group of eight algorithms working with strategy $\Gamma_1(n = 1)$, then a group of eight algorithms working with strategy $\Gamma_3(n = 3)$, and so on until $\Gamma_9(n = 9)$. Within each group, the eight algorithms are, from left to right: *Binary-GA, PSame-GA, PHalf-GA*, *PMax-GA, Binary-SH, PSame-SH, PHalf-SH*, and *PMax-SH*.

Fig. 6(A) plots the results for optimizing mission productivity. Overall, performance increases as the number of control vectors n used by the control strategy Γ_n increases. For $n = 1$, performance ranges from an average of 20 535 units of science generated by *PSame-GA* to statistically similar averages of 29 822 and 29 748 units generated by *PMax-SH* and *PMax-GA*. For $n = 3$, performance fluctuates between 30 731 and 34 931 units of science generated by *Binary-GA* and *PHalf-GA*, respectively. *PMax-GA* performance of 34 894 is statistically similar to that of *PHalf-GA*. For $n = 5$, performance fluctuates between 31 076 and 38 251 units of science generated by *PMax-SH* and *PHalf-GA*. For the higher values of $n, n = 7$ and $n = 9$, performance ranges from 29 05 to 38 118 and from 28 267 to 38 016 with the worst and best performances in both cases generated by *PMax-SH* and *PSame-GA*, respectively. The best performance for this set of experiments (A) is 38 251 units of science, achieved by *PHalf-GA* with $n = 5$. *PSame-GA* with $n = 7$ and *PSame-GA* with $n = 9$ achieve results that are statistically equivalent to the best. The best performance obtained by a SH algorithm is of 37 173 by *PSame-SH* with $n = 7$ which is statistically worse than the best GA performance. Overall, for all the values of n in (A), the best-performing algorithms are proportional GAs with only one exception: when $n = 1$, the performances of *PMax-SH* and *PMax-GA* are statistically equivalent.

Fig. 6(B) plots the results for optimizing mission duration. Again, we see an increase in performance with increasing n, a trend that appears to be independent of the optimization function used. For $n = 1$, performance ranges from an average of 6 654 time steps by *PSame-GA* to an average of 10 030 time steps by *PMax-GA*. For $n = 3$, the range is an average of 9 067 time steps by *PMax-SH* to an average of 10 207 time steps by *PSame-SH*. For $n = 5$ and $n = 7$, performance ranges from 10 189 to 14 993 and from 11 287 to 15 786, respectively, with *PMax-SH* performing the worst and *PMax-GA* performing the best in both cases. For $n = 9$, performance ranges from 12 610 by *PMax-SH* to 16 012 by *PHalf-GA*. The best performance achieved in set (B) is an average of 16 012 simulation time steps by *PHalf-GA* in the $n = 9$ case, and *PMax-GA* with $n = 5$ performance of 15 786 is statistically indistinguishable from this best. The best performance obtained by a SH algorithm is an average of 14 615 achieved by *Binary-SH* with $n = 5$ which is significantly worse than the best GA performance. Overall, for all n in (B), the best performing algorithms are again GAs using proportional representations, with an exception in the case of $n = 3$. For $n = 3$, the best performing algorithm is an SH using proportional representation.

Fig. 6(C) plots the results for optimizing both mission productivity and duration. These results exhibit similar trends as seen in the previous plots. Performance increases with the increase of control vectors n used by control strategy Γ_n. Performance ranges are:

- for $n = 1$; 40 592 by *PSame-GA* to 59 781 by *PMax-GA*;
- for $n = 3$; 63 713 by *Binary-GA* to 72 641 by *PHalf-GA*;
- for $n = 5$; 61 945 by *PMax-SH* to 77 296 by *PHalf-GA*;
- for $n = 7$; 57 312 by *PMax-SH* to 78 626 by *PSame-GA*;
- for $n = 9$; 56 062 by *PMax-SH* to 78 860 by *PSame-GA*.

The best performance obtained in (C) is 78 860 by *PSame-GA* with $n = 9$. The performance of 78 626 by *PSame-GA* with $n = 7$ is statistically equivalent to the best. The best performance obtained by a SH algorithm is 75 843 by *PSame-SH* with $n = 7$ which is significantly worst than the best GA performance. Overall, for any n in (C), the best performing algorithms are again GAs using proportional representations.

All of these experiments use the parameter settings shown in Table II; however, additional experiments have been performed with other mutation rates and population sizes to ensure that the qualitative nature of the results in this section is robust and not an artifact of a particular set of parameters.

V. Discussion

According to the NFL theorem [40], no universally superior algorithm exists; however, algorithms may be best suited for particular problem classes. Given an algorithm, the choice of representation can have a significant impact on its effectiveness.

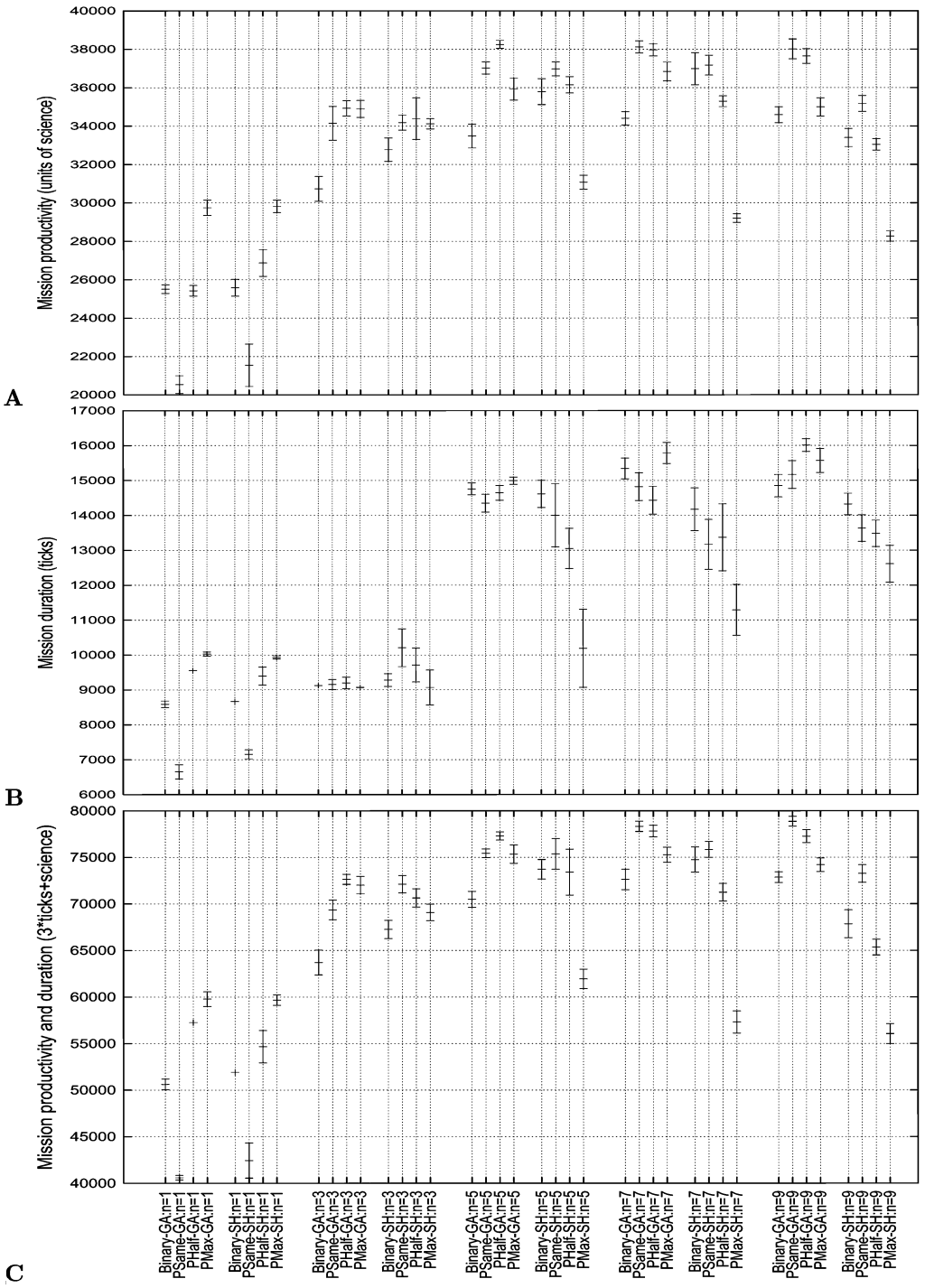

Fig. 6. GA and SH with binary, proportional-same, proportional-half, and proportional-max representations using control strategies $\Gamma_1, \Gamma_3, \Gamma_5, \Gamma_7$, and Γ_9 to optimize ALSS (A) mission productivity, (B) mission duration, and (C) both. Fitness of best solution found averaged over 40 runs and 95% confidence intervals.

Simpler representations may be inadequate in the amount of information that can be expressed [13]. Complex representations result in larger search spaces which can increase the difficulty of a problem and the computational cost of a learning algorithm [34]. Different types of representations may have biases that make them more or less easily manipulated by a learning algorithm [9], [24]. This work compares the performance of two ML algorithms using two different types of problem representations applied to ALSS control.

A. Algorithms: GA Versus SH

In comparing all of the GA and SH variants that we test, we find that GAs tend to have better overall performance than SHs for the ALSS problem. GAs generate the best performance in all groups (as defined by optimization objective and control strategy) with only once exception: optimizing mission duration with $n = 3$. GA performance consistently increases with increasing n. SH becomes less competitive with GAs

as n increases and the worst performance is consistently and noticeably found using *PMax-SH*. As problem size grows, GA performance remains relatively stable while SH shows noticeable decline. We attribute the weaknesses of SH to the simplicity of the SH search strategy—move in single steps only toward higher fitness points—which can easily become trapped at local optima. In addition, the SH algorithm we selected does not allow random walks among solutions of equivalent fitness which can further exacerbate the problem of becoming trapped (we plan to eliminate this restriction in future experiments). The more complex operators and population based approach of a GA appear to provide a better search strategy for this type of problem. Nevertheless, the simple SH search strategy is still a competitive strategy for an ALSS as it is usually only outperformed by a small, albeit statistically significant, margin.

B. Control Strategies: $N = 1, 3, 5, 7, 9$

The plots from Fig. 6 show a clear trend: performance improves as the number of control vectors n used in the optimization increases. This result remains fairly robust throughout our experiments and supports previous work [23] where this same trend is reported for a simple GA. We attribute this behavior to the fact that more control vectors allows for a more subtle control of the ALSS, leading to noticeably improved performance. There is, however, a tradeoff between the better control allowed by a high n and the consequent increase in the search space size. Increasing n increases the search space exponentially. As a result, we expect to see a performance drop at high values of n due to searchability issues. We do not observe such expected behavior for high n which we speculate is due to the fact that we work with relatively low values of n (from 1 to 9). We do observe, however, that after $n = 5$, increases of n yield diminishing performance returns.

C. Representations: Binary Versus Proportional

The proportional representation is inspired by the natural process of gene expression and the concept that, within a genome, what is most important is whether or not the necessary genes exist. Of lesser impact is the order in which existing genes are arranged. This concept led to the development of a content-based representation for GAs. Previous GA studies have shown advantages to using a proportional representation over traditional binary representation on several simple problems: number matching, resource allocation, and symbolic regression [41]. In this paper, we extend those results by showing that this advantage persists on a significantly more complex problem. The proportional GA consistently outperforms a binary GA on the problem of control of a coupled dynamical system. We suspect that advantages may be due to the proportional GA's ability to dynamically allocate genomic resources and adjust resolution in response to fitness payoffs.

We also test the proportional representation in a SH. Results indicate that, in contrast with the GA results, the performance of a proportional SH is comparable but not necessarily better than that of a traditional binary SH. We believe that a likely reason for this difference is that the SH algorithm that we selected only moves to points of greater fitness in the search space. As a result, it does not allow random walks on neutral networks of points with equal fitness. Proportional representation is known to have a high redundancy which forms neutral networks in the search space that arguably can improve the search abilities of algorithms; but for this to be beneficial, an algorithm would have to allow random walks. We plan to re-examine SH performance while allowing random walks. We should also note that, while SH uses an operator analogous to mutation to move within a search space, SH lacks an operator analogous to GA recombination.

D. Resolution: P-Same Versus P-Half Versus P-Max

In Fig. 6, the x-axis orders the tested algorithms into groups of four: the *Binary, PSame, PHalf*, and *PMax* representations. Consecutive foursomes indicate GA and SH algorithms for the five control strategies: $n = 1, 3, 5, 7, 9$.

Overall, the best performing algorithms tend to be GAs using a proportional representation. Closer examination of the best performing proportional algorithm within each foursome suggests that the value of n can affect the type of proportional representation that performs best. For low n $(n = 1)$ the best performing algorithms all use the *PMax* representation. For high n $(n = 7, 9)$ the best performing algorithms tend to use the *PSame* representation. For middle values of n $(n = 3, 5)$ the best performing algorithms tend to use *PHalf* or *PMax* for a GA and *PSame* or *PHalf* for an SH. We believe that this variation is due to the resolution-searchability tradeoff described in Section V-B: low resolution limits the quality of solutions, but reduces the search space allowing solutions to be found faster; high resolution increases solution quality at the cost of increasing the size of the space to search. Low n generates a small search space on which there is advantage to having the maximum possible resolution, *PMax*, in the representation. High n results in large search spaces, making the limited resolution and relatively smaller search space of *PSame* more attractive because of its increased searchability.

E. Comparing Binary and Proportional Representations

In order to compare the relative performance of proportional and binary representations, we calculate, for all the experiments reported in Fig. 6, the number of times that an algorithm using a proportional representation is statistically better, comparable, or worse than the same algorithm using a binary representation. Table V presents our results for the GA and Table VI for the SH.

In Table V, *PSame-GA* uses the same genomic length as *Binary-GA* which, according to (1), puts *PSame-GA* at a sharp disadvantage. Despite this limitation, *PSame-GA* performs better than *Binary-GA* in 53% of the experiments and worse in 20%. *PHalf-GA* uses a longer genome than *Binary-GA*, but is still theoretically at a disadvantage because it has only *half* of the genomic length required by (1) to ensure comparable resolutions. Despite this theoretical limitation, *PHalf-GA* performs better than *Binary-GA* in 73% of the experiments, and worse in only 7% of the experiments. *PMax-GA* uses a variable-length genome with the maximum length equal to the required genomic length given by (1). As a result, *PMax-GA* and *Binary-GA* are comparable in terms of available resolution. *PMax-GA* performs better than *Binary-GA* 73% of the time and never performs worse than *Binary-GA*. Overall, for all of the cases

TABLE V
COMPARISON OF PROPORTIONAL-GA AND BINARY-GA

	Comparing Proportional-GA and Binary-GA (number of experiments; each experiment = 40 runs)		
	Proportional-GA outperforms Binary-GA	No statistical difference	Proportinoal-GA performs worse than Binary-GA
P-Same	8(53%)	4(27%)	3(20%)
P-Half	11(73%)	3(20%)	1(7%)
P-Max	11(73%)	4(27%)	0(0%)
Total	30(67%)	11(24%)	4(9%)

TABLE VI
COMPARISON OF PROPORTIONAL-SH AND BINARY-SH

	Comparing Proportional-SH and Binary-SH (number of experiments; each experiment = 40 runs)		
	Proportional-SH outperforms Binary-SH	No statistical difference	Proportional-SH performs worse than Binary-SH
P-Same	6(40%)	6(40%)	3(20%)
P-Half	5(33%)	6(40%)	4(27%)
P-Max	5(33%)	1(7%)	9(60%)
Total	16(35.5%)	13(29%)	16(35.5%)

compared in Table V, proportional GAs perform better than binary GAs in 67% of the experiments and worse in 9% of the experiments.

Table VI shows a very different situation for the SH algorithms. The *PSame-SH* representation outperforms *Binary-SH* in 40% of the experiments but performs worse in 20%. The situation becomes worse as we increase the resolution of the proportional representation with 33% better and 27% worse for *PHalf-SH*, and 33% better and 60% worse for *PMax-SH*. There is a clear preference for compact representations over high resolutions. Overall, the proportional SHs perform better than binary SHs in 35.5% of the experiments, worse in 35.5% of experiments, and show no statistical difference in 29% of experiments. As a result, there does not appear to be a clear advantage to using either binary or proportional representation in an SH.

VI. CONCLUSION

ALSSs are a crucial component for successful space exploration. An ALSS is a coupled dynamical system whose overall behavior is difficult to predict and whose behavior is highly sensitive to initial conditions and control parameters. The unpredictability of such systems limits the effectiveness of deliberative and hand-crafted control algorithms. We investigate the performance of two well known ML techniques, GAs and SH, on the problem of learning how to control an ALSS.

The selection of representation can have a significant impact on the effectiveness and efficiency of a learning algorithm. The same problem encoded with two different representations can appear to an ML algorithm to be two entirely different problems [26]. We compare the performance of a GA and SH using two different types of representations: a traditional binary representation and a novel proportional representation. The proportional representation, originally developed for GAs, is a content-based representation derived from the concept of gene expression. Its ability to dynamically adapt the distribution of genomic resource along with a solution allows it to naturally evolve both parameters values and their appropriate resolutions. We hypothesize that this unique ability may make the proportional representation particularly suitable for the highly sensitive nature of coupled dynamical systems.

We perform experiments on three ALSS optimization problems using five control strategies. For each of these experiments, we compare the performance of a GA and SH, each using a binary representation and three variations of proportional representations: *PMax* which has an encoding resolution equal to that of the binary representation, *PHalf* which has an encoding resolution that is one half of *PMax*, and *PSame* which has a severely penalized encoding resolution as compared to the binary representation.

All algorithms show increasing performance as the number of vectors, n, in the control strategy Γ_n increases. We attribute this behavior to the fact that more control vectors are likely to allow a more precise control of the system. This result extends previous results obtained for a simple GA to the eight algorithm-representation combinations tested in this paper.

Experimental results show that the GA consistently performs as well or better than the SH. The top performance is achieved by a GA variant in all the 15 experiments, with only one exception: optimizing mission duration with strategy Γ_3. In fact, all of the best performers use the proportional representation, supporting our hypothesis that a proportional representation is competitive. For lower values of n, *PMax-GA* appears to be the strongest algorithm, and for higher values of n, *PSame-GA* appears to be the strongest algorithm. We attribute the differences to the tradeoff between the size of the search space and the resolution of the representation. For lower n, the search space is small enough to afford a full resolution of parameters, while for higher n and the corresponding larger search spaces, preference is given to limited resolution representations to favor searchability over resolution.

Finally, we compare the relative performance of the three proportional representation variants with binary representation for each algorithm. When using a GA, all of the proportional representation variants tend to be more successful than binary representation. On average, the proportional representation GAs outperform the binary GA 67% of the time and perform worse than the binary GA 9% of the time. The *PMax-GA* representation appears to be strongest, outperforming *Binary-GA* 73% of the time; exhibiting statistically equivalent performance the remaining 27% of the time. *PMax-GA* is never outperformed by *Binary-GA*. Comparison of proportional and binary representations for the SH yields a very different picture. On average the proportional SHs outperform the binary SH 35% of the time and underperform by the same percentage. As a result, while proportional representation yields a strong improvement in performance for a GA, it does not significantly affect the performance of the SH algorithms.

APPENDIX A

The simulator used in this paper is the ALSS, based on the NASA BioPlex simulator [38]. The ALSS is a coupled

dynamical system. The behaviors of its components or modules are completely deterministic and can be characterized by the state of the variables in any time step of a simulation. Transition equations determine the next simulation step for any given set of these state variables. The behavior of the entire system of coupled modules, however, is difficult to predict. The following is a mathematical description of the ALSS simulator used in this paper.

Definition 1 (ALSS): Formally, an ALSS simulation $\mathcal{A}$ is denoted by a 4-tuple

$$\mathcal{A} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}, \delta_{\mathcal{C}} \rangle$$

where

- $\mathcal{Q}$ set of simulation states;
- $\vec{q_0}$ in $\mathcal{Q}$ is the initial state of the simulation;
- $\mathcal{C}$ control strategy that defines a vector of control parameters for each simulation time step;
- $\delta_{\mathcal{C}}$ finite set of transition equations [3]

Definition 2 (Simulation Variables): Let $\mathcal{V}$ denote the set of all the variables associated with the simulation $\mathcal{A}$, then

$$\mathcal{V} \stackrel{\text{def}}{=} \{\vec{\mathbf{e}}, \vec{\mathbf{r}}, \vec{\mathbf{o}}, \vec{\mathbf{a}}, \vec{\mathbf{w}}\}$$

where

$$\vec{\mathbf{e}} \stackrel{\text{def}}{=} \langle \mathrm{e}_{\text{energy}}, \mathrm{e}_{\text{air.o2}}, \mathrm{e}_{\text{air.co2}}, \mathrm{e}_{\text{water.clean}}, \mathrm{e}_{\text{water.dirty}} \; \mathrm{e}_{\text{food}}, \mathrm{e}_{\text{science}}, \mathrm{e}_{\text{store.water}}, \mathrm{e}_{\text{store.food}}, \mathrm{e}_{\text{store.}o2}, \mathrm{e}_{\text{store.co2}} \rangle$$
vector of **environmental** variables

$$\vec{\mathbf{r}} \stackrel{\text{def}}{=} \langle \mathrm{r}_{\text{status}}, \mathrm{r}_{\text{food}}, \mathrm{r}_{\text{water}}, \mathrm{r}_{\text{air.}o2}, \mathrm{r}_{\text{starvation}}, \mathrm{r}_{\text{dehydration}} \; \mathrm{r}_{\text{asphyxiation}} \rangle,$$ vector of **crew** variables

$$\vec{\mathbf{o}} \stackrel{\text{def}}{=} \langle \mathrm{o}_{\text{status}}, \mathrm{o}_{\text{energy}}, \mathrm{o}_{\text{water}}, \mathrm{o}_{\text{low.energy}}, \mathrm{o}_{\text{low.water}}, \mathrm{o}_{\text{age}} \; \mathrm{o}_{\text{biomass}}, \mathrm{o}_{o2.\text{released}}, \mathrm{o}_{\text{co2.absorved}} \rangle$$
vector of **crop** variables

$$\vec{\mathbf{a}} \stackrel{\text{def}}{=} \langle \mathrm{a}_{\text{energy}}, \mathrm{a}_{\text{low.energy}}, \mathrm{a}_{\text{recovery.time}} \rangle$$
vector of **air recovery** variables

$$\vec{\mathbf{w}} \stackrel{\text{def}}{=} \langle \mathrm{w}_{\text{in}}, \mathrm{w}_{\text{low.energy}}, \mathrm{w}_{\text{uptime}}, \mathrm{w}_{\text{potable}}, \mathrm{w}_{\text{energy}} \rangle$$
vector of **water recovery** variables,

Definition 3 (Simulation State): A simulation state, $\vec{q}$ in $\mathcal{Q}$, is a particular assignment of values for all the simulation the variables in $\mathcal{V}$. The simulation state at time t is

$$\vec{q_t} \stackrel{\text{def}}{=} \langle \vec{\mathbf{e}}_t, \vec{\mathbf{r}}_t, \vec{\mathbf{o}}_t, \vec{\mathbf{a}}_t, \vec{\mathbf{w}}_t \rangle.$$

Definition 4 (Initial State): The initial simulation state denoted by $\vec{q_0}$ in $\mathcal{Q}$ is a defined as follows:

$$\vec{q_0} \stackrel{\text{def}}{=} \langle \vec{\mathbf{e}}_0, \vec{\mathbf{r}}_0, \vec{\mathbf{o}}_0, \vec{\mathbf{a}}_0, \vec{\mathbf{w}}_0 \rangle$$

where

$$\begin{aligned}\vec{\mathbf{e}}_0 \stackrel{\text{def}}{=} \langle & \mathrm{e}_{\text{energy}} = 10000, \mathrm{e}_{\text{air.}o2} = 3000 \times 0.2095 \\ & \mathrm{e}_{\text{air.co2}} = 3000 \times 0.0003, \mathrm{e}_{\text{water.clean}} = 10 \\ & \mathrm{e}_{\text{water.dirty}} = 0, \mathrm{e}_{\text{food}} = 4, \mathrm{e}_{\text{science}} = 0 \\ & \mathrm{e}_{\text{store.water}} = 500, \mathrm{e}_{\text{store.food}} = 500 \\ & \mathrm{e}_{\text{store.}o2} = 50000, \mathrm{e}_{\text{store.co2}} = 50\,000 \rangle\end{aligned}$$

$$\begin{aligned}\vec{\mathbf{r}}_0 \stackrel{\text{def}}{=} \langle & \mathrm{r}_{\text{status}} = 1, \mathrm{r}_{\text{food}} = 0, \mathrm{r}_{\text{water}} = 0, \mathrm{r}_{\text{air.}o2} = 0 \\ & \mathrm{r}_{\text{starvation}} = 0, \mathrm{r}_{\text{dehydration}} = 0, \mathrm{r}_{\text{asphyxiation}} = 0 \rangle\end{aligned}$$

$$\begin{aligned}\vec{\mathbf{o}}_0 \stackrel{\text{def}}{=} \langle & \mathrm{o}_{\text{status}} = 1, \mathrm{o}_{\text{energy}} = 0, \mathrm{o}_{\text{water}} = 0 \\ & \mathrm{o}_{\text{low.energy}} = 0, \mathrm{o}_{\text{low.water}} = 0, \mathrm{o}_{\text{age}} = 0 \\ & \mathrm{o}_{\text{biomass}} = 0, \mathrm{o}_{o2.\text{released}} = 0, \mathrm{o}_{\text{co2.absorved}=0} \rangle\end{aligned}$$

$$\vec{\mathbf{a}}_0 \stackrel{\text{def}}{=} \langle \mathrm{a}_{\text{energy}} = 0, \mathrm{a}_{\text{low.energy}} = 0, \mathrm{a}_{\text{recovery.time}} = 0 \rangle$$

$$\begin{aligned}\vec{\mathbf{w}}_0 \stackrel{\text{def}}{=} \langle & \mathrm{w}_{\text{in}} = 0, \mathrm{w}_{\text{low.energy}} = 0, \mathrm{w}_{\text{uptime}} = 10 \\ & \mathrm{w}_{\text{potable}} = 0, \mathrm{w}_{\text{energy}} = 0 \rangle.\end{aligned}$$

Definition 5 (Control Strategy): Let $\mathcal{C}$ denote a control strategy and be defined as follows:

$$\mathcal{C} \stackrel{\text{def}}{=} \{\vec{\mathbf{c}}^t \,|\, (0 \le t \le T_{\max}) \wedge t \in N\}$$

where

$$\begin{aligned}\vec{\mathbf{c}}^t \stackrel{\text{def}}{=} & \langle \mathrm{c}^t_{\text{energy.to.air}}, \mathrm{c}^t_{\text{energy.to.water}}, \mathrm{c}^t_{\text{energy.to.food}} \\ & \mathrm{c}^t_{\text{water.to.crew}}, \mathrm{c}^t_{\text{water.to.crops}}, \mathrm{c}^t_{\text{activity.level}} \rangle \\ & \text{is the vector of } \mathbf{control} \text{ parameters} \\ & \text{values for time } t.\end{aligned}$$

$\mathcal{C} \in \mathbf{C}$ and $\mathbf{C}$ is the set of all posible control strategies.

Definition 6 (Transition Equations): The set of transition equations denoted by $\delta_{\mathcal{C}}$ defines how to obtain the next simulation state $(t+1)$ from the current state (t) for a given control strategy $\mathcal{C}$. The simulation $\mathcal{A}$ is a coupled dynamical system with four components: crew, crops, air revitalization, and water revitalization. Therefore, we have four sets of equations

$$\delta_{\mathcal{C}} \stackrel{\text{def}}{=} \langle \delta_{\mathcal{C}_{\text{crew}}}, \delta_{\mathcal{C}_{\text{crops}}}, \delta_{\mathcal{C}_{\text{airRevitalization}}}, \delta_{\mathcal{C}_{\text{waterRevitalization}}} \rangle.$$

For definitions of $\delta_{\mathcal{C}_{\text{crew}}}, \delta_{\mathcal{C}_{\text{crops}}}, \delta_{\mathcal{C}_{\text{airRevitalization}}}$, and $\delta_{\mathcal{C}_{\text{waterRevitalization}}}$, please refer to [17].

Definition 7 (Simulation Step): Applying the transition equations $\delta_{\mathcal{C}}$ to the state $\vec{q}_t$, we obtain the next state $\vec{q}_{t+1}$ of the simulation $\mathcal{A}$

$$\vec{q}_{t+1} \stackrel{\text{def}}{=} \delta_{\mathcal{C}}(\vec{q}_t)$$

where

$$\delta_{\mathcal{C}}(\vec{q}_t) \stackrel{\text{def}}{=} \delta_{\mathcal{C}_{\text{waterRevitalization}}} \left(\delta_{\mathcal{C}_{\text{airRevitalization}}} \left(\delta_{\mathcal{C}_{\text{crops}}} \left(\delta_{\mathcal{C}_{\text{crew}}}(\vec{q}_t) \right) \right) \right).$$

Definition 8 (Simulation t Steps): The reflexive and transitive closure of $\delta_{\mathcal{C}}$ is denoted by $\widehat{\delta}_{\mathcal{C},t}$

$$\vec{q}_t \stackrel{\text{def}}{=} \widehat{\delta}_{\mathcal{C},t}(\vec{q_0}).$$

Definition 9 (Simulation End Time): Simulation $\mathcal{A}$ ends when the environment variable crew status $\mathrm{r}_{\text{status}}$ is equal to zero (the environment is not longer able to support human life) or when the limit time for the simulation $T_{\max}$ is reached. We denote this ending time as ${t_{\text{Final}}}^{\mathcal{A}}$

$${t_{\text{Final}}}^{\mathcal{A}} \stackrel{\text{def}}{=} \min_{t \le T_{\max}} [\vec{q}t = \widehat{\delta}_{\mathcal{C},t}(\vec{q_0}) \wedge (\mathrm{r}_{\text{status}})_t = 0]$$

where

$$(\mathrm{r}_{\text{status}})_t \in \vec{q}_t, \vec{q}_t \in \mathcal{Q}, \quad \text{and } \mathcal{A} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}, \delta_{\mathcal{C}} \rangle.$$

[3]Due to space constraints, we do not include the transition equations. For a complete mathematical description of the ALSS simulator used in this paper, please refer to [17].

Definition 10 (Final State): The final state of simulation $\mathcal{A}$ is denoted by $\vec{q}^{\mathcal{A}}_{t_{\text{Final}}}$ and defined as follows:

$$\vec{q}^{\mathcal{A}}_{t_{\text{Final}}} \stackrel{\text{def}}{=} \left\{ \vec{q}_t \,\middle|\, \vec{q}_t = \widehat{\delta}_{\mathcal{C},t}(\vec{q_0}) \wedge t = t^{\mathcal{A}}_{\text{Final}} \right\}$$

where

$$\vec{q}^{\mathcal{A}}_{t_{\text{Final}}}, \vec{q}_t \in \mathcal{Q}, \quad \text{and } \mathcal{A} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}, \delta_{\mathcal{C}} \rangle.$$

A. Optimization

Definition 11 (Optimization 1): Maximize mission productivity. Let $\mathcal{C}_{\text{Opt1}}$ denote the optimal control strategy for maximizing mission productivity and be defined as follows:

$$\mathcal{C}_{\text{Opt1}} \stackrel{\text{def}}{=} \left\{ \mathcal{C}_j \,\middle|\, \left(\begin{matrix} \forall \\ \mathcal{C}_i \in \mathbf{C} \end{matrix} \right) \left[(\text{e}_{\text{science}})^{\mathcal{A}_j}_{t_{\text{Final}}} \geq (\text{e}_{\text{science}})^{\mathcal{A}_i}_{t_{\text{Final}}} \right] \right\}$$

where $\mathcal{C}_{\text{Opt1}}, \mathcal{C}_i$, and $\mathcal{C}_j \in \mathbf{C}; \mathcal{A}_i = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_i, \delta_{\mathcal{C}_i} \rangle$; $\mathcal{A}_j = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_j, \delta_{\mathcal{C}_j}$; $(\text{e}_{\text{science}})^{\mathcal{A}_i}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_i}_{t_{\text{Final}}}$; $(\text{e}_{\text{science}})^{\mathcal{A}_j}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_j}_{t_{\text{Final}}}$; and $\mathbf{C}$ is the set of all posible control strategies.

Definition 12 (Optimization 2): Maximize mission duration. Let $\mathcal{C}_{\text{Opt2}}$ denote the optimal control strategy for maximizing mission duration, and be defined as follows:

$$\mathcal{C}_{\text{Opt2}} \stackrel{\text{def}}{=} \left\{ \mathcal{C}_j \,\middle|\, \left(\begin{matrix} \forall \\ \mathcal{C}_i \in \mathbf{C} \end{matrix} \right) \left[t^{\mathcal{A}_j}_{\text{Final}} \geq {t_{\text{Final}}}^{\mathcal{A}_i} \right] \right\}$$

where $\mathcal{C}_{\text{Opt2}}, \mathcal{C}_i$, and $\mathcal{C}_j \in \mathbf{C}; \mathcal{A}_{\text{i}} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_i, \delta_{\mathcal{C}_i} \rangle; \mathcal{A}_{\text{j}} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_j, \delta_{\mathcal{C}_j} \rangle$; and $\mathbf{C}$ is the set of all posible control strategies.

Definition 13 (Optimization 3): Maximize mission productivity and duration. Let $\mathcal{C}_{\text{Opt3}}$ denote the optimal control strategy for maximizing mission productivity and mission duration, and be defined as follows:

$$\mathcal{C}_{\text{Opt3}} \stackrel{\text{def}}{=} \left\{ \mathcal{C}_j \,\middle|\, \left(\begin{matrix} \forall \\ \mathcal{C}_i \in \mathbf{C} \end{matrix} \right) \left[\alpha(j) \geq \alpha(i) \right] \right\}$$

where $\alpha(k) \stackrel{\text{def}}{=} (\text{e}_{\text{science}})^{\mathcal{A}_k}_{t_{\text{Final}}} + 3 \times {t_{\text{Final}}}^{\mathcal{A}_k}; \mathcal{C}_{\text{Opt3}}, \mathcal{C}_i$, and $\mathcal{C}_j \in \mathbf{C}$; $\mathcal{A}_{\text{i}} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_i, \delta_{\mathcal{C}_i} \rangle$; $\mathcal{A}_{\text{j}} = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_j, \delta_{\mathcal{C}_j} \rangle$; $(\text{e}_{\text{science}})^{\mathcal{A}_i}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_i}_{t_{\text{Final}}}$; $(\text{e}_{\text{science}})^{\mathcal{A}_j}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_j}_{t_{\text{Final}}}$; and $\mathbf{C}$ is the set of all posible control strategies.

B. Fitness

Definition 14 (Fitness 1): Mission productivity. Let $f_1(\mathcal{C}_k)$ denote the fitness of the control strategy $\mathcal{C}_k$ while optimizing mission productivity in simulation $\mathcal{A}_k$, be defined as follows:

$$f_1(\mathcal{C}_k) = \left\{ (\text{e}_{\text{science}})^{\mathcal{A}_k}_{t_{\text{Final}}} \,\middle|\, (\text{e}_{\text{science}})^{\mathcal{A}_k}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_k}_{t_{\text{Final}}} \wedge \mathcal{A}_k = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_k, \delta_{\mathcal{C}_k} \rangle \right\}.$$

Definition 15 (Fitness 2): Mission duration. Let $f_2(\mathcal{C}_k)$ denote the fitness of the control strategy $\mathcal{C}_k$ while optimizing mission duration in simulation $\mathcal{A}_k$, be defined as follows:

$$f_2(\mathcal{C}_k) = \left\{ {t_{\text{Final}}}^{\mathcal{A}_k} \,|\mathcal{A}_k = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_k, \delta_{\mathcal{C}_k} \right\}.$$

Definition 16 (Fitness 3): Mission productivity and duration. Let $f_3(\mathcal{C}_k)$ denote the fitness of the control strategy $\mathcal{C}_k$ while optimizing mission productivity and mission duration in simulation $\mathcal{A}_k$, be defined as follows:

$$f_3(\mathcal{C}_k) = \left\{ (\text{e}_{\text{science}})^{\mathcal{A}_k}_{t_{\text{Final}}} + 3 \times {t_{\text{Final}}}^{\mathcal{A}_k} \,\middle|\, (\text{e}_{\text{science}})^{\mathcal{A}_k}_{t_{\text{Final}}} \in \vec{q}^{\mathcal{A}_k}_{t_{\text{Final}}} \wedge \mathcal{A}_k = \langle \mathcal{Q}, \vec{q_0}, \mathcal{C}_k, \delta_{\mathcal{C}_k} \rangle \right\}.$$

ACKNOWLEDGMENT

The authors would like to thank D. Kortenkamp and P. Bonasso of NASA JSC for access to the BIO-Plex simulator code. They would also like to thank the anonymous reviewers for their many helpful comments and suggestions.

REFERENCES

[1] P. Abrahams, R. Balart, J. Byrnes, D. Cochran, M. J. Larkin, W. Moran, G. Ostheimer, and A. Pollington, "MAAP: The military aircraft allocation planner," *Proc. IEEE World Congr. Computer Intelligence*, pp. 336–341, 1998.

[2] D. Ackley, *A Connectionist Machine for Genetic Hill-climbing*. Norwell, MA: Kluwer, 1987.

[3] S. Baglioni, D. Sorbello, C. da Costa Pereira, and A. G. B. Tettamanzi, "Evolutionary multiperiod asset allocation," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO)*, 2000, pp. 597–604.

[4] D. J. Barta, J. M. Castillo, and R. E. Fortson, "The biomass production system for the bioregenerative planetary life support systems test complex: Preliminary designs and considerations," in *Proc. 29th Int. Conf. Environmental Systems*, Paper 1999-01-2188, 1999.

[5] R. P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, "Experiences with an architecture for intelligent, reactive agents," *J. Explor. Theor. Artificial Intell.*, vol. 9, no. 2/3, pp. 237–256, 1997.

[6] *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., IOP/Oxford Univ. Press, Bristol/Oxford, U.K., 1997.

[7] D. S. Burke, K. A. De Jong, J. J. Grefenstette, C. L. Ramsey, and A. S. Wu, "Putting more genetics into genetic algorithms," *Evol. Comput.*, vol. 6, no. 4, pp. 387–410, 1998.

[8] A. R. Callaghan, A. R. Nair, and K. E. Lewis, "A genetic algorithm based method for optimal resource allocation: A case study of the buffalo niagara international airport expansion," in *Proc. 3rd World Congr. Structural and Multidisciplinary Optimization*, 1999.

[9] P. C. Cheng and H. A. Simon, "The right representtion for discovery: Finding the conservation of momentum," in *Proc. 9th Int. Workshop on Machine Learning*, 1992, pp. 62–71.

[10] D. Cousins, J. Loomis, F. Roeber, P. Schoeppner, and A.-E. Tobin, "The embedded genetic allocator—A system to automatically optimize the use of memory resources in high performance, scalable computing systems," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. 3, 1998, pp. 2166–2171.

[11] S. S. Crawford, C. W. Pawlowski, and C. K. Finn, "Power management in regenerative life support systems using market-based control," in *Proc. ICES*, 2000.

[12] L. Davis, "Bit-climbing, representational bias, and test suite design," in *Proc. 4th Int. Conf. Genetic Alg (ICGA)*, L. Booker and R. Belew, Eds., 1991, pp. 18–23.

[13] T. G. Dietterich and R. S. Michalski, "A comparative review of selected methods for learning from examples," in *Mach. Learn.* New York, 1983, pp. 41–81.

[14] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Biases in the crossover landscape," in *Proc. 3rd ICGA*, 1989, pp. 10–19.

[15] C. K. Finn, "Dynamic System Modeling of Regenerative Life Support Systems," NASA Ames Res. Ctr., Tech. Rep. 1999-01-2040, 1999.

[16] D. H. Fleisher and K. C. Ting, "Modeling and control of plant production for advanced life support," in *Acta Horticult.*, 2001.

[17] I. I. Garibay and A. S. Wu, "Advanced life support system simulation," Univ. Central Florida, Orlando, Tech. Rep. CS-TR-03-05, 2003.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[19] S. Goudarzi and K. C. Ting, "Top level modeling of crew component of alss," in *Proc. 29th Int. Conf. Environmental Systems*, 1999.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press, 1975.
[21] A. Juels and M. Wattenberg *et al.*, "Hillclimbing as a baseline method for the evaluation of stochastic optimization algorithms," in *Advances Neural Info. Process. Syst.*, vol. 8, D. S. Touretzky *et al.*, Eds., 1995, pp. 430–436.
[22] D. Kortenkamp and S. Bell, "BioSim: An integrated simulataion of an advanced life support system for intelligent control research," in *Proc. 7th Int. Symp. AI, Robotics, and Automation in Space*, 2003.
[23] D. Kortenkamp, R. P. Bonasso, and D. Subramanian, "Distributed, autonomous control of space habitats," in *Proc. IEEE Aerospace Conf.*, 2001.
[24] J. H. Larkin and H. A. Simon, "Why a diagram is (sometimes) worth ten thousand words," *Cogn. Sci.*, vol. 11, pp. 65–99, 1987.
[25] T. L. Lau and E. P. K. Tsang, "The guided genetic algorithm and its application to the generalized assignment problem," in *Proc. 10th IEEE Int. Conf. Tools with AI*, 1998, pp. 336–343.
[26] K. Mathias and L. D. Whitley, "Transforming the search space with gray coding," in *Proc. IEEE Conf. Evol. Comput.*, 1994, pp. 513–518.
[27] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1998, ch. 10.
[28] M. Mitchell, J. H. Holland, and S. Forrest, "When will a genetic algorithm outperform hill climbing," *Adv. Neural Info. Processing Syst.*, vol. 6, pp. 51–58, 1994.
[29] *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., IOP Pub. Ltd. & Oxford Univ. Press, 1997. V. Nissen, "Management applications and other classical optimization problems'.
[30] Y. Owechko and S. Shams, "Comparison of neural network and genetic algorithms for a resource allocation problem," *IEEE World Congr. Computational Intelligence*, vol. 7, pp. 4655–4660, 1994.
[31] S. Palaniappan, S. Zein-Sabatto, and A. Sekmen, "Dynamic multiobjective optimization of war resource allocation using adaptive genetic algorithms," in *Proc. IEEE Southeast Conf.*, 2001, pp. 160–165.
[32] S. Papavassiliou, A. Puliafito, O. Tomarchio, and J. Ye, "Integration of mobile agents and genetic algorithms for efficient dynamics network resource allocation," in *Proc. 6th IEEE Symp. Computers and Communications*, 2001, pp. 456–463.
[33] ——, "Mobile agent-based approach for efficient network management and resource allocation: Framework and applications," *IEEE J. Select. Areas Commun.*, vol. 20, pp. 858–872, Apr. 2002.
[34] C. Sammut, "Knowledge representation," in *Machine Learning, Neural and Statistical Classification*. ser. Ellis Horwood, D. Michie, D. J. Spiegelhalter, and C. C. Taylor, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1994, pp. 228–245.
[35] D. Schreckenghost, C. Thronesbery, P. Bonasso, D. Kortenkamp, and C. Martin, "Intelligent control of life support for space missions," *IEEE Intell. Syst.*, pp. 24–31, Sept./Oct. 2002.
[36] M. R. Sherif, I. W. Habib, M. Nagshineh, and P. Kermani, "A generic bandwidth allocation scheme for multimedia substreams in adaptive networks using genetic algorithms," in *Proc. Wireless Communications and Networking Conf.*, vol. 3, 1999, pp. 1243–1247.
[37] ——, "Adaptive allocation of resources and call admission control for wireless ATM using genetic algorithms," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 268–282, Feb. 2000.
[38] T. O. Tri, "Bioregenerative planetary life support systems test complex (BIO-Plex): Test mission objectives and facility development," in *Proc. 29th Int. Conf. Environmental. Systems*, Paper 1999-01-2186, 1999.
[39] M. Williams, "Making the best use of the airways: An important requirement for military communications," *Electron. Commun. Eng. J.*, pp. 75–83, 2000.
[40] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Energy Conversion*, vol. 1, pp. 67–82, Jan. 1997.
[41] A. S. Wu and I. Garibay, "The proportional genetic algorithm: Gene expression in a genetic algorithm," *Genetic Programming and Evolvable Hardware*, vol. 3, no. 2, pp. 157–192, 2002.

Annie S. Wu received the Ph.D. degree in computer science and engineering from the University of Michigan, Ann Arbor.

She is an Assistant Professor in the School of Computer Science and Director of the Evolutionary Computation Laboratory at the University of Central Florida (UCF), Orlando. Before joining UCF, she was a National Research Council Postdoctoral Research Associate at the Naval Research Laboratory.

Ivan Garibay received the M.S. degree in natural language generation from the University of Central Florida (UCF), Orlando, where he is pursuing the Ph.D. degree in the School of Computer Science.

Before joining UCF, he was Lecturer in the Electrical Engineering Department, Ricardo Palma University, Lima, Peru. His research interests include genetic algorithms, evolutionary computation, evolution of complexity, and artificial life.