Simulating Lunar Habitats and Activities to Derive System Requirements

David Kortenkamp [*]

NASA Johnson Space Center/Metrica Inc., Houston, Texas, 77058, USA

Scott Bell[†]

NASA Johnson Space Center/SKT Inc., Houston, Texas, 77058, USA

Luis Rodriguez [‡]

NASA Johnson Space Center/USRA, Houston, Texas, 77058, USA

Simulations allow for the testing of different system configurations before hardware is built. It is important that such simulations are dynamic and that they integrate all major subsystems and activities. This paper describes a specific habitat simulation being built at NASA Johnson Space Center. It is a discrete-event simulation that is dynamic and stochastic. It simulates all major subsystem of a Lunar or Martian habitat, including the crew (with variable ages, weights and genders), biomass production (with scalable plantings of nine different crops), water recovery, air revitalization, food processing, solid waste recycling and energy production. Crew activities are modeled including: exercise, sleep and work. Extravehicular activities (EVAs) are simulated by having crew members leave the habitat taking small amounts of consumable resources with them and then returning later with fewer consumables. Habitat pressure is simulated including leak rates and losses due to airlocks and EVA. Malfunctions can be injected at any time into any subsystem. The simulation also models sensors and actuators to provide for control opportunities. The simulation is written in an object-oriented paradigm that makes it portable, extensible and reconfigurable. In addition to introducing the simulation, this paper also presents results from using the simulation to analyze five different Lunar habitat configurations.

I. Introduction

Direct experimentation with physical systems is slow, expensive, dangerous and not possible until the physical system is built. Simulations allow for testing of ideas without having to build hardware. There are many types of simulations. For example, dynamic simulations versus static simulations; continuous simulations versus discrete simulations; and deterministic simulations versus stochastic simulations to name a few. This paper describes a particular habitat simulation that is dynamic, discrete and stochastic. It models most of the components of a physical habitat including the crew, crops, water and air recovery systems, EVAs and power. This simulation can be used to test various habitat configurations and components before building an actual habitat. In the work described in this paper we use an automated search tool called a genetic algorithm to find an optimal habitat configuration for a ninety day mission to the moon.

II. Simulating a lunar habitat

Our habitat simulation is based on an existing simulation called BioSim.[1] BioSim has been developed at NASA JSC over the past three years and is still under active development. It is a generic habitat simulation so our first task was to create a specific instance of BioSim for a lunar habitat. Because instances of BioSim

[*]Senior Scientist, Metrica Inc, NASA JSC Mail Stop ER2, Houston TX 77058

[†]Scientist, SKT Inc., NASA JSC Mail Stop ER2, Houston TX 77058

[‡]Research Scientist, Universities Space Research Association, NASA JSC Mail Stop EC2, Houston TX 77058

are stored in an eXtensible Markup Language (XML) file, creating a new instance of BioSim does not require changing any computer code. The XML file is read by the BioSim simulation at start-up and the appropriate configuration is instantiated. This makes creating different kinds of habitats very easy even for those with no programming experience. In addition, we are working on a graphical user interface (GUI) to further simplify the process.

A. Simulation overview

A typical habitat system consists of multiple interacting modules.[2] We have modeled most of these modules using the best available information. The models are process models in that they take in certain resources and produce other resources. They are not component models, that is, they do not model physical objects such as valves, pumps, etc. Figure 1 shows the modules in our simulation. We now describe each of them in detail.

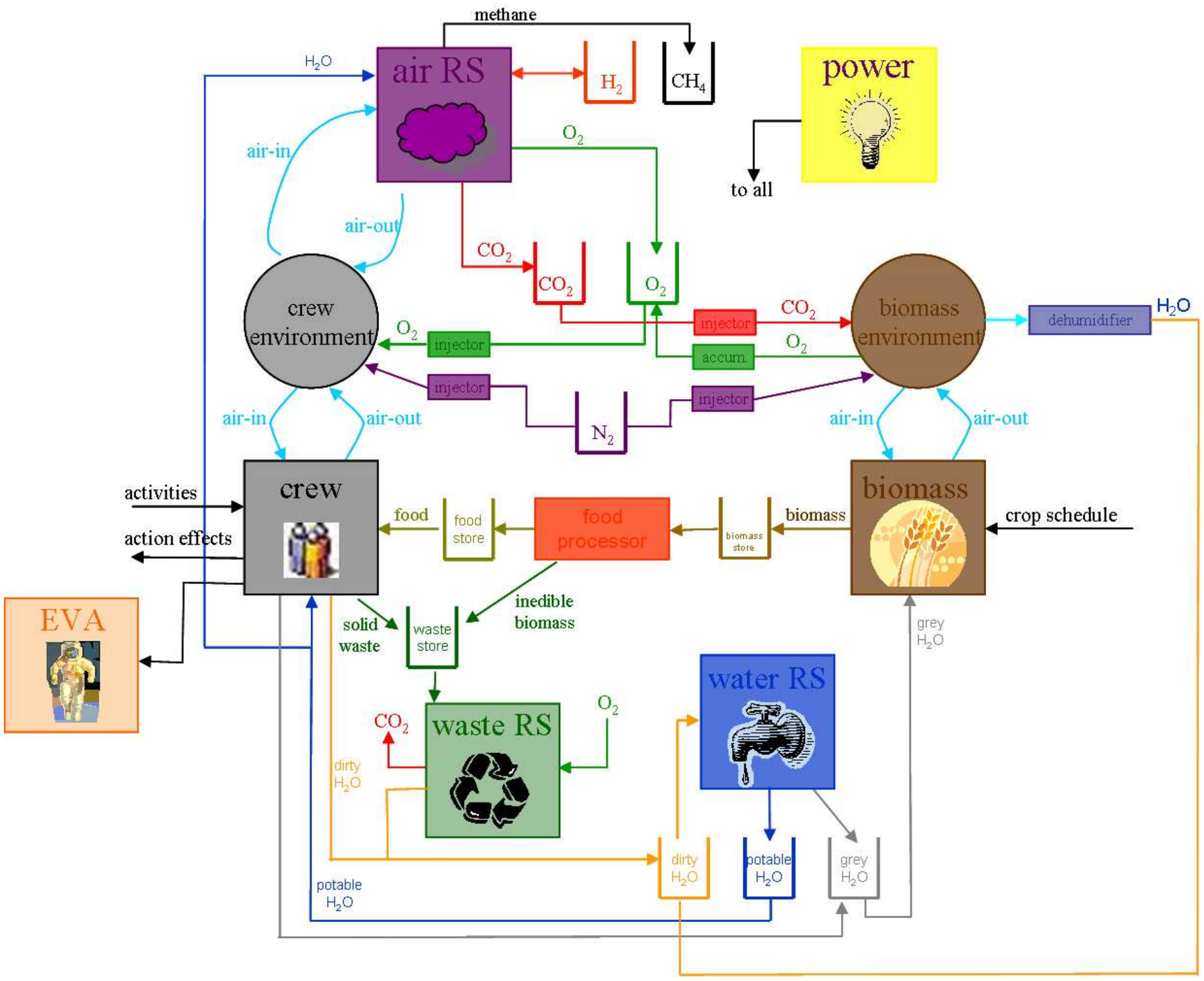


Figure 1. The various modules that comprise a habitat life support system

1. *Crew*

The crew module is implemented using models described by Goudarzi and Ting.[3] The number, gender, age and weight of the crew are settable as input parameters. The crew cycles through a set of activities (sleep, maintenance, recreation, etc.). As they do so they consume O_2, food and water and produce CO_2, dirty water and solid waste. The amount of resources consumed and produced varies according to crew member attributes and their activities. The crew's activities can be adjusted by passing a new crew schedule to the crew module. A default schedule can also be used. The crew module is connected to a crew environment that contains an atmosphere that they breathe. The initial size and gas composition (percentages of O_2, CO_2, H_2O and inert gases) are input parameters. As the simulation progresses the composition of gases in the atmosphere changes.

2. Biomass

The biomass module models crops that produce food and oxygen in the simulation. Currently, nine crops are modeled – wheat, rice, soybean, tomato, lettuce, dry bean, peanut, sweet potato, and white potato. The growing area of the biomass module is fixed at the start of the simulation. Within this growing area the amounts of the different crops can be adjusted during a simulation run either through the crew interface or directly through the biomass interface. As crops grow they consume CO_2, potable or grey water and light. At the same time, they produce O_2 and transpire H_2O. The crops also produce biomass when they are harvested. The production and consumption of resources are modeled according to Barta, et al.,[4] Jones and Cavazzoni,[5] and the Baseline Values and Assumptions Document (BVAD) published by NASA's Advanced Life Support (ALS) Program.[6] The biomass module has its own environment that contains an atmosphere for the plants. The default configuration has separate biomass and crew environments because the ideal gas composition for growing plants is not ideal for humans. Because the simulation is reconfigurable it can be initialized with a single environment for crew and crops or with multiple environments for different crops. Harvesting and planting of crops is currently done automatically with the default that the same type of crop is planted as was harvested. However, a different crop schedule can be passed to the biomass module. Crops are planted and harvested in shelves that contain 8.2 square meters of growing area and have their own artificial lighting. The lighting and water available to the crops is adjustable.

3. Food processing

The food processing module takes in harvested biomass and produces edible and inedible mass. The inedible mass is further divided into inedible dry mass, which goes to the solid waste store and inedible water mass, which goes to the dirty water store. Edible biomass is converted into a food object type that has three properties: calories, water, and mass. Each of the nine crops are converted into nine different food types. The food processing module adds these food types to the food store, which may contain other user defined food not produced by the biomass production system. When the crew module needs food it asks the food store for a specific number of calories. The food store creates a mix of food types to provide those calories and passes them to the crew module while deleting that food from the food store. The crew module uses the water content of food to reduce the amount of potable water consumed by the crew. In the future, we would like to add the ability to request specific menus and to balance other nutrients besides just calories.

4. Air revitalization

The air revitalization module consists of several subsystems that provide breathable air to the crew environment. The Variable Configuration CO_2 Removal System (VCCR) takes air from the crew atmosphere, removes CO_2, and returns the remaining air mixture. The removed CO_2 is collected in a CO_2 store. The CO_2 Reduction System (CRS) takes CO_2 from the store and reacts it with H_2, making H_2O and CH_4. The methane (CH_4) is vented (the simulation tracks how much is vented) and the H_2O is passed to the third system, the O_2 Generation System (OGS), which breaks down the H_2O into H_2 and O_2. The O_2 is saved in an O_2 store and the H_2 is returned to the CRS. The air revitalization module may take H_2O from the potable water store for use with the OGS, or it may put H_2O into the potable water store. In addition, an O_2 accumulator extracts O_2 from the biomass atmosphere and places it in an O_2 store. Injectors are available to take gases from the stores and inject them into the atmospheres. A control challenge is to maintain an optimal gas mixture in the crew and biomass environments while minimizing both energy use by the air revitalization module and minimizing store sizes. The capacities of the stores can be assigned at initialization. All modules require power to although the OGS requires substantially more power than the other subsystems.

5. Water recovery

The water recovery module consumes dirty water and produces potable and grey water (i.e., wastewater other than that including human excrement, such as bath water). The water recovery module consists of four subsystems that process the water. The biological water processing (BWP) subsystem removes organic compounds. Then the water passes to a reverse osmosis (RO) subsystem, which recovers 85% of the water. The 15% of the water not recovered from the RO (called brine) is passed to the air evaporation subsystem (AES), which recovers the rest. These two streams of grey water (from the RO and the AES) are passed through a post-processing subsystem (PPS) to be purified and made potable water. An external controller can turn on or off various subsystems. For example, all water can pass through the AES at a higher energy cost. We based our water recovery module on a recently completed test at NASA Johnson Space Center.[7]

6. Extravehicular activities

Extravehicular activities (EVA) involve crew members leaving the habitat for variable periods of time. While outside the habitat the crew members no longer place a load on the habitat life support systems, but instead need alternate life support systems. We implement this in our simulation by spawning a "mini" habitat with parameters settable by the user. This mini-habitat exists for the duration of the EVA. The mini-habitat has its own environment, stores, recycling systems, etc. As an example, the mini-habitat could be a spacesuit with a (small) air volume, small water stores, a very small air revitalization system and a small power supply. This mini-habitat would be suitable for one crew member. During the mini-habitat's existence the crew member would breath her air, drink her water, etc. and would not be drawing from the original, larger habitat. When the EVA is over, the mini-habitat is merged back with the original habitat. Because all parameters of this mini-habitat are settable, it could represent any EVA hardware, from space suits to pressurized vehicles capable of holding multiple crew members for extended periods. For set-up and control purposes the mini-habitat is no different from any other habitat simulation and uses the same underlying models. They can be spawned dynamically at any time and for any duration. Multiple spawned habitats can exist at the same time.

7. Solid waste

The current solid waste module models an incineration process in order to recover carbon. The module takes dry waste from the crew and food processing modules and oxygen to produce carbon dioxide. We plan to add a lyophilizer to the solid waste module in the future.

8. Power

The power production module supplies electricity to all of the other modules. There are two models for this module. One simulates a nuclear-style power system that supplies a continuous, fixed amount of power. A second simulates a solar-style power system that supplies a varying amount of power. An external control program can set the amount of power going to each module up to the total amount of power available.

B. Habitat parameters

The previous section (Section A) described a generic habitat simulation. For the experiments in this paper we implemented a specific instance of the simulation to reflect a lunar habitat. The instance was designed with information from an internal JSC memo describing a lunar reference mission.[8]

The reference mission assumes a four person crew with equal numbers of men and women. A mission is 90 days with the habitat initiated and operating nominally upon crew arrival. The landing site is the lunar south pole with the sun above the horizon 80% of the time and surface temperatures between 210K and 230K during the day. The habitat atmosphere is composed of 29% oxygen at an overall pressure of 65.5 kPa and a leakage rate of 0.00224 kg/day. Food is shipped in most circumstances (although we looked at the addition of small salad crops) and is 0.257 kg/crewmember-day moist food and 0.665 kg/crewmember-day of dry food. Air, water, and waste recovery systems are part of the habitat.

Parameters such as the size of the habitat (specifically air volume), the size of the recovery systems, the amount of salad crops and the size of the power subsystem were not fixed and were determined through analysis of simulation results as described in Section IV.

1. *Extravehicular activities*

As described in Section 6 we can spawn EVAs at any time. Following the lunar reference mission our simulation spawns one crew member on a four hour EVA each day of the mission. An EVA takes place through an airlock that is 3.7 m^3 in size and 10% of the airlock atmosphere is lost each time the airlock is used.

2. *Malfunctions*

Our simulation environment allows malfunctions to be inserted into any subsystem at any time either by the user or randomly by the system. These malfunctions can be permanent or temporary. For all of the experiments described in this paper two malfunctions were introduced into the habitat. First, there is an O_2 storage malfunction at 200 hours into the mission. This results in both a loss of 50% of the store capacity and 50% of the oxygen in the store at that time. The capacity reduction is permanent. Second, there is a power production malfunction at 502 hours into the mission that results in a permanent loss of 50% of the power production capability.

C. Lunar habitat scenarios

The goal of the experiments described in this paper was to determine appropriate system-sizing requirements for the habitat life support systems and to compare different habitat configurations. We examined five different habitat configurations:

1. No biomass production facility in the habitat – all food is from stored food brought along for the mission and all air and water recovery is performed by physicochemical subsystems.
2. Small biomass production facility in the habitat that is integrated with the crew cabin air and water recovery subsystems.
3. Small biomass production facility in the habitat that is integrated with the crew cabin air recovery subsystem, but which had its own water recovery subsystem.
4. Small biomass production facility in the habitat that is integrated with the crew water recovery subsystem, but which had its own air recovery subsystem.
5. Small biomass production facility in the habitat that has its own water and air recovery subsystems separate from the crew subsystems.

The biomass production facility of these experiments included only lettuce and tomatoes.

D. Interfacing to the simulation

BioSim has an Application Programmers Interface (API) that allows external programs to access BioSim functionality. The API is written using the Common Object Request Broker Architecture (CORBA), which standardizes network access (see http://www.omg.org). Thus, a program can be running on one computer and interfacing with BioSim running on another computer. The use of CORBA also means that external programs can be written in any language that supports CORBA, including C, C++, Java, LISP and many others. The API can be used to configure BioSim, to control any of its modules, to inject failures and to step BioSim through discrete time intervals. The API can also be used to read data from BioSim, including module status, store levels, crew activities, etc.

In the experiments described in this paper, the external program is a genetic algorithm that searches for an optimal habitat configuration (see next section). In this particular case, the genetic algorithm is written in Java and connects to the simulation via CORBA. It configures the simulation according to its gene and then uses an API call to start the simulation running. When the mission ends (due to insufficient resources or reaching the 90 day limit) an API call informs the genetic algorithm program, which evaluates the result.

The external program could also be a graphical representation of the underlying simulation. As an example we have implemented a three-dimensional animation of BioSim that receives data from a running simulation. This animation is written using the Unreal Tournament Engine (see http://www.unrealtournament.com) and scripting language and communicates using the same CORBA API. Figure 2 shows a screen shot of the

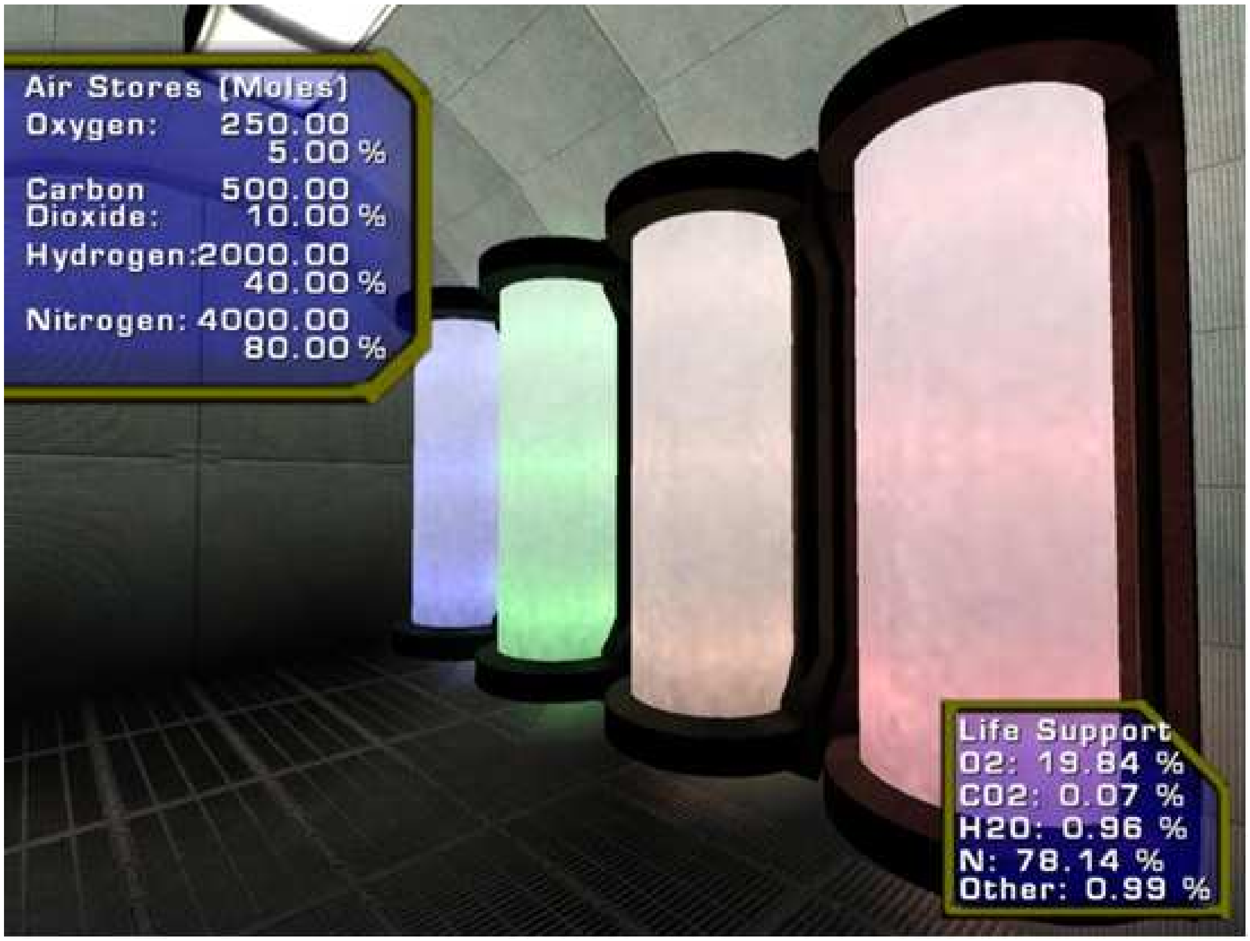


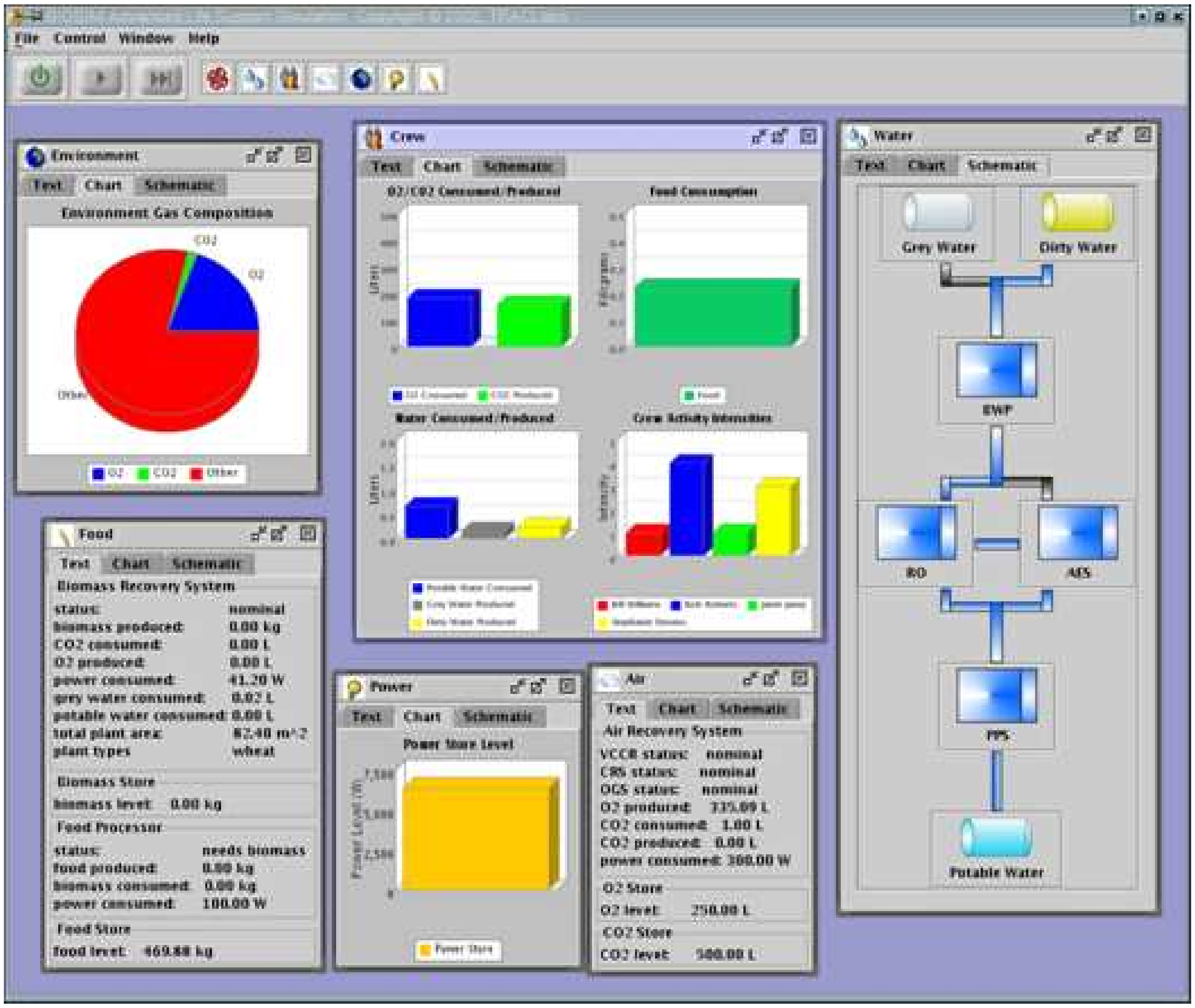
Figure 2. A three-dimensional animation of the habitat simulation.

animation. Figure 3 shows a screen shot of a more traditional 2D graphical representation of the simulation that is written in Java.

III. Genetic algorithms

A genetic algorithm[9] encodes the control inputs as a "gene" (usually a binary string). There is a fitness function that evaluates the gene. The process starts with a random population of genes with each gene evaluated. The best genes are saved while the worst genes are eliminated. The best genes are then mutated or crossed (i.e., parts of two genes are swapped with each other) creating a new population. The genes in this new population are then evaluated with the fitness function and the process continues until the population is no longer improving with respect to the fitness function.

We implemented a genetic algorithm in which the gene was a description of the initial configuration of the simulation (e.g., crop size, storage capacities, etc.). The genetic algorithm program takes this initial configuration and sets up the simulation accordingly. Our simulation is run until consumable resources become too low and the mission is ended. The fitness function is a combination of the length of the mission and the Equivalent System Mass (ESM) of the configuration. ESM is a measure of the configuration mass – a lower mass being preferred for exploration missions[10] (see next section). Thus, in our experiments "good" genes are those configurations that resulted in the longest running time with the least mass. We fixed a maximum mission length at 90 days, so after 90 days genes were only distinguished by their ESM. Good genes (i.e., those with the longest running time and least mass) are crossed, mutated or inverted. Bad genes are replaced by genes that had a higher fitness creating a new population. The process is repeated until the algorithm no longer finds genes that improve the fitness function. Simulations can run in parallel on one or



```
Food
Biomass Recovery System
status:                   nominal
biomass produced:         0.00 kg
CO2 consumed:             0.00 L
O2 produced:              0.00 L
power consumed:           41.20 W
grey water consumed:      0.02 L
potable water consumed:   0.00 L
total plant area:         82.40 m^2
plant types               wheat
Biomass Store
biomass level:   0.00 kg
Food Processor
status:              needs biomass
food produced:       0.00 kg
biomass consumed:    0.00 kg
power consumed:      100.00 W
Food Store
food level:    469.88 kg
```

```
Air
Air Recovery System
VCCR status:      nominal
CRS status:       nominal
OGS status:       nominal
O2 produced:      335.09 L
CO2 consumed:     1.00 L
CO2 produced:     0.00 L
power consumed:   300.00 W
O2 Store
O2 level:      250.00 L
CO2 Store
CO2 level:     500.00 L
```

Figure 3. A 2-dimensional graphical representation of the habitat simulation.

several machines, rapidly increasing search speed.

A. Genes

A gene in our genetic algorithm represents a life support configuration. Depending upon the scenario (see Section C) a gene will have between 12 and 20 attributes, each with an integer value. At first these integer values are randomly assigned to each attribute. Over time, the genetic algorithm adjusts each attribute's value to search for an optimal solution. The attributes and the range of values they can take are:

- All Scenarios:
  - O_2 Injector flowrate (0-30 mol/hr)
  - OGS power consumption (0 - 2500 watts)
  - VCCR power consumption (0 - 100000 watts)
  - PowerPS power production (0 - 700000 watts)
  - CrewEnvironment volume (0 - 2000000 L)
  - O_2 Store capacity/level (0 - 2000 mol)
  - PowerStore capacity/level (0 - 500000 watts)

  - FoodStore capacity/level (0 - 2000 kg)

- Scenario 1:
  - WaterRS power consumption (0 - 3000 watts)
  - PotableWaterStore capacity/level (0 - 8000 L)
  - GreyWaterStore capacity/level (0 - 4000 L)
  - DirtyWaterStore capacity/level (0 - 5000 L)
- Scenario 2:
  - Tomato Shelf Size (0 - 20 m^2)
  - Lettuce Shelf Size (0 - 20 m^2)
  - WaterRS power consumption (0 - 3000 watts)
  - PotableWaterStore capacity/level (0 - 8000 L)
  - GreyWaterStore capacity/level (0 - 4000 L)
  - DirtyWaterStore capacity/level (0 - 5000 L)
- Scenario 3:
  - Tomato Shelf Size (0 - 20 m^2)
  - Lettuce Shelf Size (0 - 20 m^2)
  - Crew WaterRS power consumption (0 - 3000 watts)
  - Crew PotableWaterStore capacity/level (0 - 8000 L)
  - Crew GreyWaterStore capacity/level (0 - 4000 L)
  - Crew DirtyWaterStore capacity/level (0 - 5000 L)
  - Plant WaterRS power consumption (0 - 3000 watts)
  - Plant PotableWaterStore capacity/level (0 - 8000 L)
  - Plant GreyWaterStore capacity/level (0 - 4000 L)
  - Plant DirtyWaterStore capacity/level (0 - 5000 L)
- Scenario 4:
  - PlantEnvironment volume (0 - 2000000 L)
  - Tomato Shelf Size (0 - 20 m^2)
  - Lettuce Shelf Size (0 - 20 m^2)
  - WaterRS power consumption (0 - 3000 watts)
  - PotableWaterStore capacity/level (0 - 8000 L)
  - GreyWaterStore capacity/level (0 - 4000 L)
  - DirtyWaterStore capacity/level (0 - 5000 L)
- Scenario 5:
  - PlantEnvironment volume (0 - 2000000 L)
  - Tomato Shelf Size (0 - 20 m^2)
  - Lettuce Shelf Size (0 - 20 m^2)
  - Crew WaterRS power consumption (0 - 3000 watts)
  - Crew PotableWaterStore capacity/level (0 - 8000 L)
  - Crew GreyWaterStore capacity/level (0 - 4000 L)
  - Crew DirtyWaterStore capacity/level (0 - 5000 L)
  - Plant WaterRS power consumption (0 - 3000 watts)
  - Plant PotableWaterStore capacity/level (0 - 8000 L)
  - Plant GreyWaterStore capacity/level (0 - 4000 L)
  - Plant DirtyWaterStore capacity/level (0 - 5000 L)

B. Fitness function

The intention of this fitness function is to identify the optimal design for the 90 day lunar mission described in Section B. An optimal design will consume minimal resources while meeting mission objectives. The fitness function utilized is:

$$f = w_t t + \frac{\sum_{\forall i}(w_i a_{i,max} - w_i a_i)}{\sum_{\forall i}(w_i a_{i,max})}$$

where f is the fitness of a configuration, t is the length of the mission in hours, w_i is the weight of attribute i, a_i is the value of attribute i and $a_{i,max}$ is the maximum value of attribute i (the list of attributes can be found in Section A). The function f is a unitless measure of fitness. Thus, the weight w_t has unit 1/hr and takes a value equal to one.

The first term, $w_t t$, measures the length of time that the simulation runs without failure in integer values, as BioSim is a discrete event simulation. The second term will never be greater than one, which allows the first term to dominate. That is, a missions lasting 2160 hours will always be rated as more fit than any mission lasting fewer than 2159 hours, no matter the amount of resources consumed. In this analysis, we are most concerned with 90 day Lunar missions (2160 hours), so we have instructed BioSim to terminate any simulations that complete 90 days. This renders the first term equal to 2160 for all 90-day missions, whether or not a configuration may have been capable of surviving for longer than that time. Since all 90-day missions will be more fit than any mission lasting less than 90 days, the second term in the fitness function then differentiates configurations selecting those that minimize resources. The second term considers the sizing of the various subsystem attributes, their relative weights, and rewards configurations with components sized smaller than their maximum values. The attributes utilized and their weights are shown in Table 1.

Attribute	Attribute Unit	Weight	Weight Unit
O_2 Injector	mol/hr	2.1060E-04	kg-hr/mol
VCCR power req	W	0.0000E+00	kg/W
OGS power req	W	0.0000E+00	kg/W
Power production	W	6.2000E-02	kg/W
Crew/crop volume	L	1.3310E-01	kg/L
O_2 storage	mol	3.3088E-02	kg/mol
Power storage	W	6.8700E-01	kg/W
Food storage	kg	2.3600E+00	kg/kg
Tomato shelf size	m^2	3.7010E+01	kg/m^2
Lettuce shelf size	m^2	3.7010E+01	kg/m^2
WaterRS power req	W	0.0000E+00	kg/W
Potable water storage	L	1.0684E+00	kg/L
Grey water storage	L	1.0684E+00	kg/L
Dirty water storage	L	1.0684E+00	kg/L

Table 1. Each configurable attribute and its contribution to the utility function.

The weights utilized in this analysis have been roughly derived based on ESM and as such allow comparison of system resources on a mass basis. When sizing components within the system the GA will vary the attributes listed in Section A. For example, the power production system is sized based upon the maximum power output necessary to sustain the system. In this analysis, this suggests that the value chosen will be the power output at Lunar noon. This aspect of the system is weighted using the cost equivalency specified in the BVAD, 0.062 kg/W.[6] Power storage, crew and crop volumes, food storage are similarly weighted using cost equivalencies.[6]

Specific cost equivalencies are not available for the remaining attribute weights so these have been derived on a case specific basis. For this purpose, it is critical to understand the manner that BioSim simulations handle power distribution during a simulation. During each hour increment within the simulation, the power system distributes power to the various systems that are scheduled to work either directly from the power production system or power storage. If at this time the available power is not enough to power all systems, some will not operate. The order which components within the system are allocated power is randomly generated each tick. Thus, there is within BioSim an impetus to size components to fit within the available power supplied by the power systems. If not, components will fail to operate due to lack of power which may lead to a system failure.

Similarly, as components are sized they must fit within the envelope defined by the crew and crop volume attributes. Currently, there is no oversight procedure within BioSim to ensure that components sized by the GA fit within the volume, leaving enough living space for the crew. This will be addressed in future versions of BioSim and at this time it is assumed that component sizing is not limited by volume.

Based on the above power and volume conventions within BioSim, the remaining attribute weights are generally sized strictly on a mass basis. The assumption is that both power and volume issues are addressed by BioSim. Cooling issues are generally proportional to power issues and are currently not incorporated in the fitness function, although it is anticipated that this will change in the future. Crew time issues are not currently addressed, but will be if sufficient data becomes available. The only remaining exception is for the the VCCR, OGS, and Water recovery systems. These systems are sized based on how much power is consumed to process their daily design loads. At the time of this analysis, a satisfactory weighting relation for power consumption to processor mass had not been identified, so a zero weight has been utilized. This implicit assumption is that the actual mass of these processors is negligible relative to their power, volume, and cooling requirements.

IV. Analysis

We used the genetic algorithm described in the previous section on the five scenarios described in Section C. This section explains our experimental methodology and our preliminary results.

A. Experimental setup

We ran a significant number of experiments using our simulation and the genetic algorithm. The experimental setup included a genetic algorithm handler program that managed the genes and the fitness function. The handler had access to fourteen instances of BioSim running on eight different computers. The first step in the experimental process was for the handler to randomly create as many genes as there were instances of BioSim – for the purposes of this explanation, let's say there are twelve instances of BioSim. The handler then sends each instance of BioSim one of the twelve randomly created genes. Each BioSim executes its gene – meaning it configures the BioSim systems according to the gene (see Section A) and then runs BioSim until either the mission ends due to lack of resources or the ninety day mission length is reached and sends its fitness back to the handler. The handler keeps genes that score in the top third (four genes in our example). Then, amongst these four genes some are crossed, some are mutated and some are inverted. Crossing happens 10% of the time and genes are crossed by taking half of one gene and half of the other. Mutating happens 80% of the time and only one parameter in the gene is mutated. Inverting happens 10% of the time and means that parameters are shuffled within the same gene. All genes are then sent back to a BioSim instance for simulation and evaluation. The process happens asynchronously as simulations may run for a long or a short period of time. Crossing, mutating and inverting happen constantly and simulations happen constantly with the top third of the genes being saved. This process continues until 1000 simulations have been run (that is, 1000 genes have been evaluated). After 1000 simulations we start the process over again with a new set of randomly chosen genes. This happens four times for each of the five configurations described in Section C. Thus, the preliminary results described in the next section are based on 20,000 simulation runs. Configurations are numbered sequentially (1-5) as they are listed in Section C. The four replications of each experiment are numbered alphabetically (A-D).

B. Preliminary results

Several interesting results have been observed. Some level of constraint verification and modified weights needs to be developed for BiosSim and the GA. Thus, results in this section should be considered preliminary. Plans have been prepared to complete this analysis in the future. At this time, several configurations have

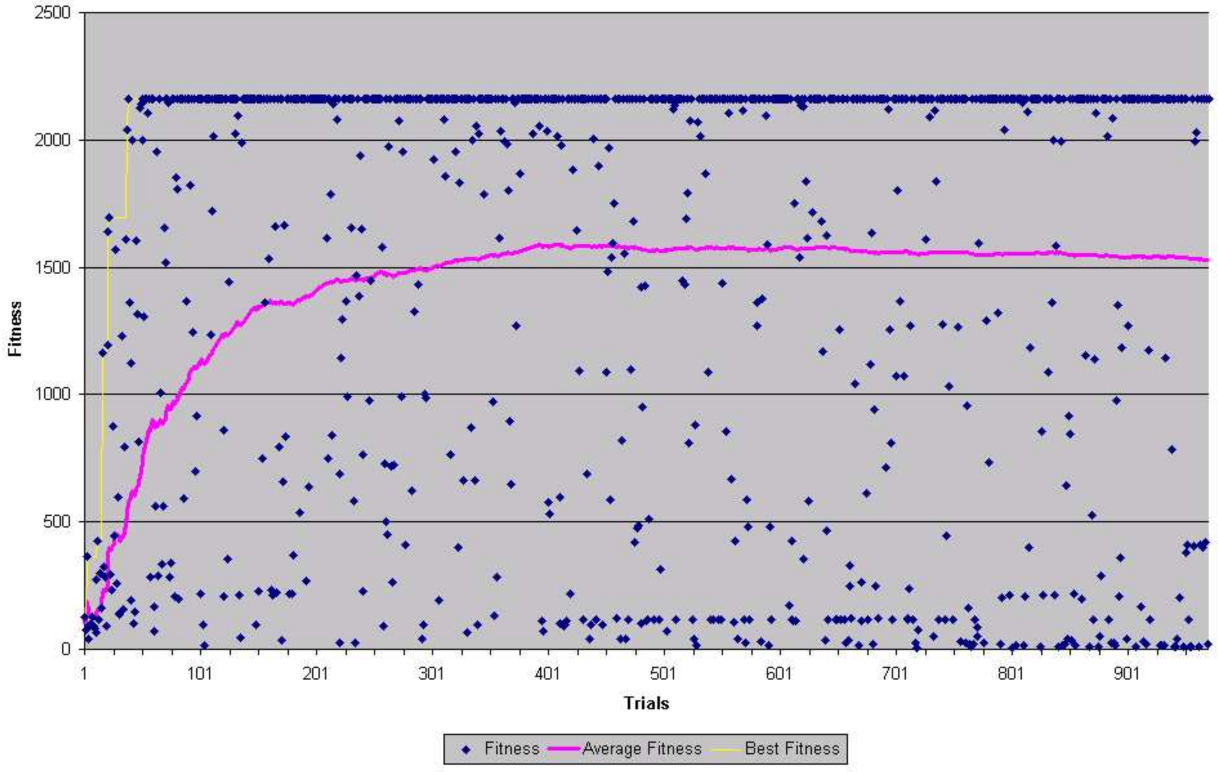


Figure 4. System evolution is depicted as the genetic algorithm attempts more trials in configuration 1 run D.

been identified that will survive the desired 90-day mission. Several components within the system have been sized to some extent, although some reduction of the error bounds would be desirable.

Figure 4 demonstrates what a prototypical GA output may look like. Charts such as Figure 4 assist in determining if the GA approached a useful solution. Diamonds mark the fitness of a individual simulation run of BioSim representing the performance of a gene chosen by the GA. Along the horizontal axis are the number of BioSim trials executed by the GA. For our experiments it is typical to have on the order of 1000 trials. The best fitness observed thus far is charted by the thin line. It can be observed in Figure 4 that the fittest configuration increases in quality as the GA completes more trials. Eventually the GA reaches the 90 day (2160 hr) threshold we are interested in. At this scale, it is not possible to observe further increases in the fitness function, although they do occur, as the thin line is hidden behind the diamond markers.

The thick line represents the average fitness observed during trials up until that trial. As the fittest configuration increases in quality it can be observed that the average fitness also increases. This indicates that the GA is in fact performing as desired, increasing the fitness of configurations it chooses, thereby increasing the average fitness.

Figure 5 shows an example of a parameter that was not conclusively sized by the GA, although this configuration did approach the desired 90-day mission. In this figure, the diamonds indicate the sizing chosen by the GA in a simulation as related to the resultant fitness. The failure of the GA to size the VCCR is most apparent at the extreme right of the figure where a distinct column of diamonds appear indicating that multiple solutions of various sizes all have equal fitness. The fitness at the extreme right of the chart corresponds to that of 2160, or a 90 day mission. This indicates that precise VCCR sizing is not critical for successful execution of a 90 day mission in this configuration.

Alternative output is demonstrated in Figure 6, where food storage is related to system fitness. In this case, it is apparent that mission viability is closely related to the available food in the system for the crew. For example, no amount of food less than approximately 700 kg was able to sustain a crew beyond 1500 hours. It is clearly shown that the minimum amount of food required for a viable mission is proportional to the mission length. Excess food, however, is not a limiting factor.

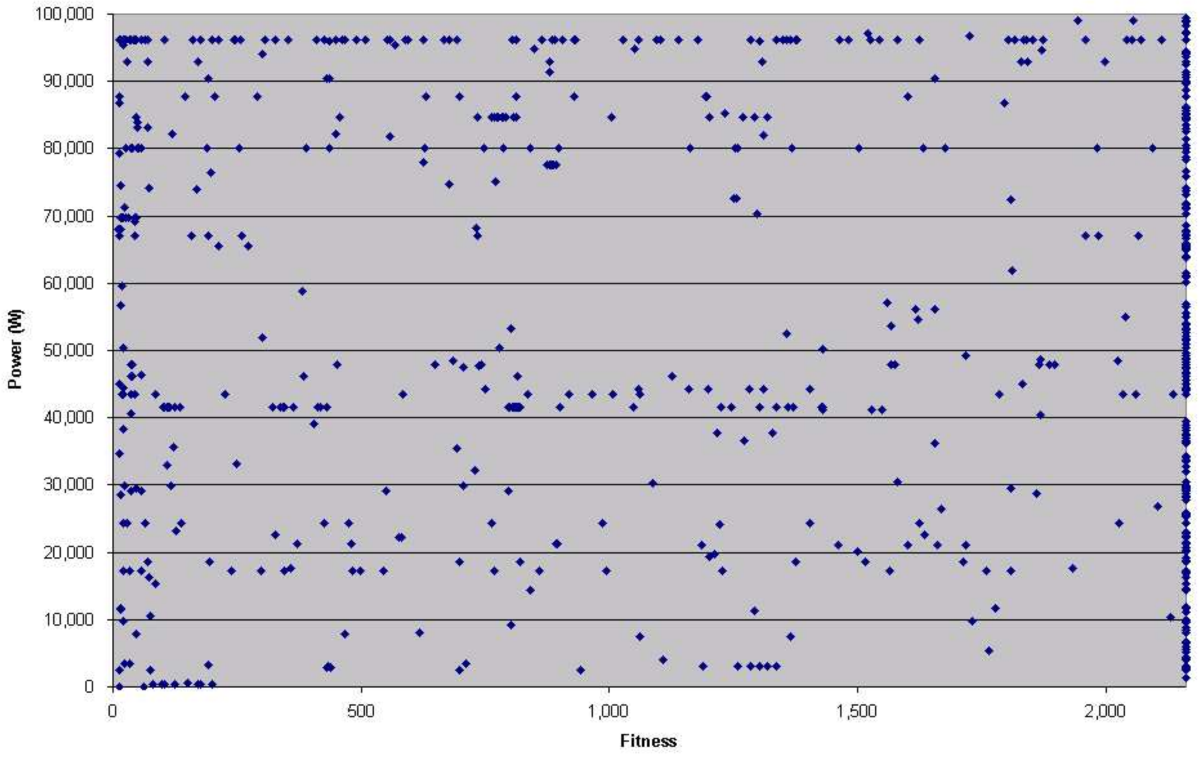


Figure 5. The relationship between fitness and VCCR power consumption in configuration 1 run B. No concrete conclusion can be made regarding VCCR sizing from this output, as is the case with all replications of the GA with configuration 1.

Similarly, maximima can be identified by the GA. This is show in Figure 7. As a general theme, power consumption has been identified by the GA as a critical issue. Maximum power consumption values, such as that pictured in Figure 7, has been identified in each configuration for many components. One notable exception is the the VCCR component in configuration 1 (see Figure 5).

In some cases, minima and maxima can be identified for an attribute varied by the GA. Figure 8 shows that a minimum amount of oxygen generation is required for this configuration to sustain a proper O_2 partial pressure, however excessive oxygen generation overtaxes the power system.

It has been observed that it can be worthwhile to consider the range of the fitness function that minimizes the consumption of resources. For example, Figure 9 demonstrates the manner that reductions in power storage were capable of increasing fitness when comparing systems that achieve the 90-day goal. This is likely due to the relative weight of power storage.

Interestingly, the VCCR can be sized very precisely in configurations 2-5. Figure 10 shows that systems that survive a complete 90 Day mission must have a VCCR sized to consume slightly less than 3 kW.

By considering the 4 replications of each GA analysis of each configuration it is possible to preliminarily size several components within the system. Referring to Figures 11 we observe that some components are sized more precisely than others, indicated by tight error bounds. It is theorized that tighter error bounds suggest a component has a more profound effect upon fitness. Further, Figure 12 demonstrates how the VCCR was not precisely sized in configuration 1, while it was sized quite precisely in configurations 2-5. It has been theorized that since configurations 2-5 all include a crop subsystem, this may have prevented the GA from oversizing the VCCR, which would have removed the CO_2 necessary for crop growth.

The term $\sum_{\forall i}(w_i a_i)$ from the numerator of the second term of the fitness function can be utilized as a proxy for ESM, provided that the weights are correctly representative of the metric. As is the case for ESM, small values for the proxy are also indicative of preferred configurations. Averaged values from the replicates of each GA run for each configuration are displayed in Figure 13. As expected, there is a distinct cost for including crops within the system. Interestingly, the divorce of both the water and air revitalization

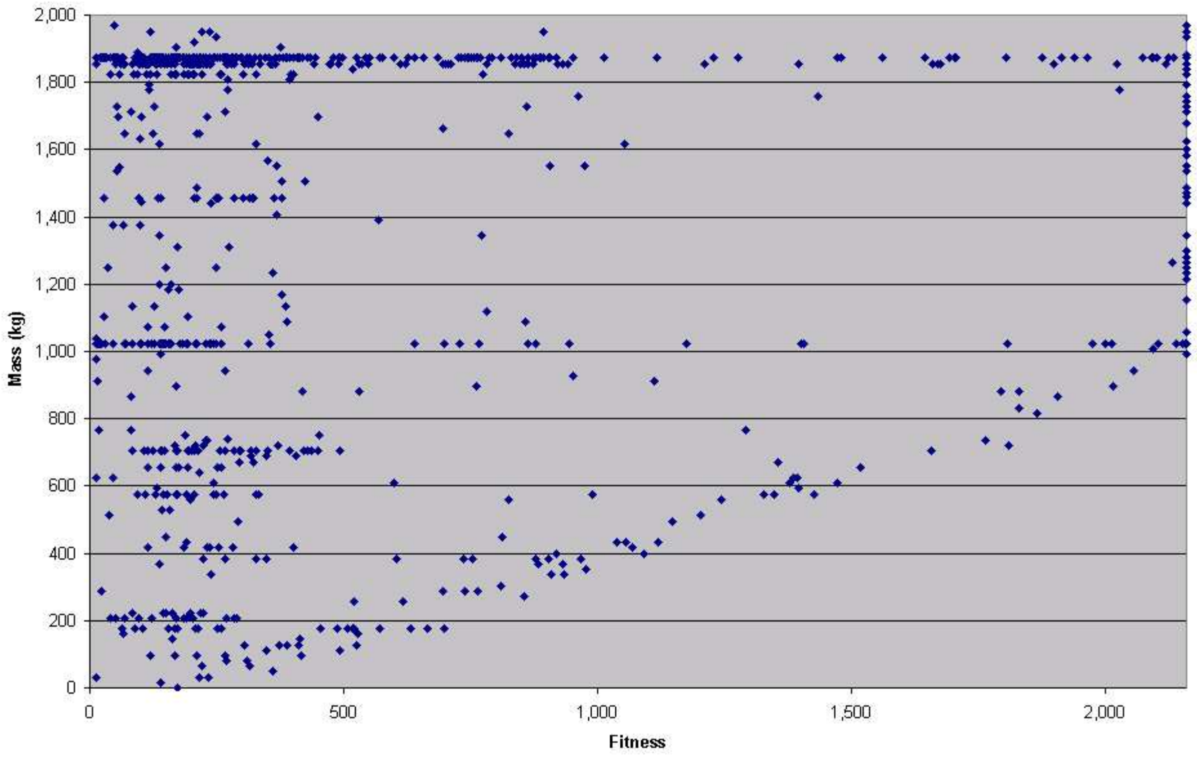


Figure 6. Adequate food is critical for mission success as is shown here in configuration 2 run A, where a minimum food masses are related to mission length, and thus fitness.

systems for the crew and crop subsystems greatly increases the value of the ESM proxy by over an order of magnitude. It is not likely that such a configuration will be preferred from an ESM perspective. It is theorized that this increase is due to the lack of CO_2 additions to the crop environment, which would otherwise be provided by crew respiration within an integrated atmosphere. This causes the GA to select a costly large crop environment volume to provide the necessary CO_2 and buffering capacity.

V. Conclusion

This paper demonstrates the use of dynamic simulations and automated search tools in configuring and sizing lunar habitats. Dynamic simulations allow for inserting run-time failures into habitat systems and taking those failures into account during the design process. Automated search tools can explore vast search spaces, honing in on specific configurations that produce optimal designs. In our case we searched through 20,000 configurations within a much, much larger search space (greater than 2^{24}). Our conclusions with respect to an optimal habitat configuration are preliminary. However, our primary conclusion is that dynamic simulations and automated search tools can help a human designer or design team to home in on optimal configurations and search for interesting combinations that might never have been tried. The ability to repeatedly and quickly simulate habitat configurations and to change those configurations in mid-stream allow a designer much greater flexibility. The simulation, automated search tool and utility function can all be replaced with different algorithms and we showed just one possible combination. The use of fast turn-around design tools, especially early in mission design, can provide significant cost savings both in design and in the mission itself.

References

[1]Kortenkamp, D. and Bell, S., "Simulating Advanced Life Support Systems for Integrated Controls Research," *Proceedings International Conference on Environmental Systems*, 2003.

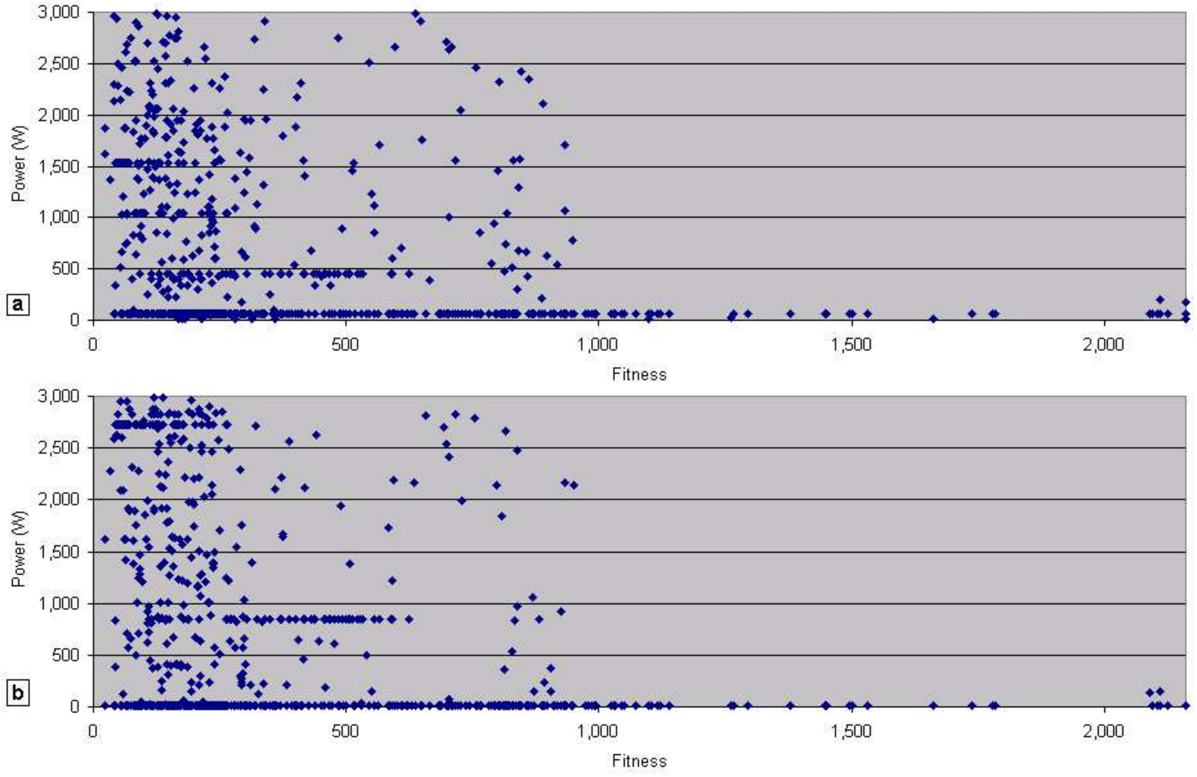


Figure 7. Excessive power consumption is shown to be a limiting factor in enabling missions of extended duration. Diagram 'a' shows the relation of the crew water recovery system in configuration 3 run D. Diagram 'b' shows the same relation for the crop water recovery system in the same simulation.

[2]NASA JSC, "JSC Advanced Life Support Requirements Document," Tech. Rep. JSC-38571C/CTSD-ADV-245C, NASA JSC (available at: http://advlifesupport.jsc.nasa.gov), 2003.

[3]Goudarzi, S. and Ting, K., "Top-Level Modeling of Crew Component of ALSS," *Proceedings International Conference on Environmental Systems*, 1999.

[4]Barta, D. J., Castillo, J. M., and Fortson, R. E., "The Biomass Production System for Bioregenerative Planetary Life Support Systems Test Complex: Preliminary Designs and Considerations," *Proceedings International Conference on Environmental Systems, SAE paper 1999-01-2188*, 1999.

[5]Jones, H. and Cavazzoni, J., "Top-Level Crop Models for Advanced Life Support Analysis," *Proceedings International Conference on Environmental Systems, SAE paper 2000-01-2261*, 2000.

[6]NASA JSC, "Advanced Life Support Baseline Values and Assumptions Document," Tech. Rep. JSC-47804/CTSD-ADV-484A, NASA JSC (available at: http://advlifesupport.jsc.nasa.gov), 2004.

[7]Bonasso, R. P., Kortenkamp, D., and Thronesbery, C., "Intelligent Control of a Water Recovery System," *AI Magazine*, Vol. 24, No. 1, 2003.

[8]Hanford, A., "Transient Modeling Challenge: A Lunar Reference Mission for a 90-Day Habitat," 2004.

[9]Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor MI, 1975.

[10]Levri, J. A., Drysdale, A. E., Esert, M. K., Fisher, J. W., Hanford, A. J., Hogan, J. A., Jones, H. W., Joshi, J. A., and Vaccari, D. A., "Advanced Life Support Equivalent System Mass Guidelines Document," Tech. Rep. NASA/TM-2003-212278 (available from http://advlifesupport.jsc.nasa.gov), NASA Center for AeroSpace Information, 2003.

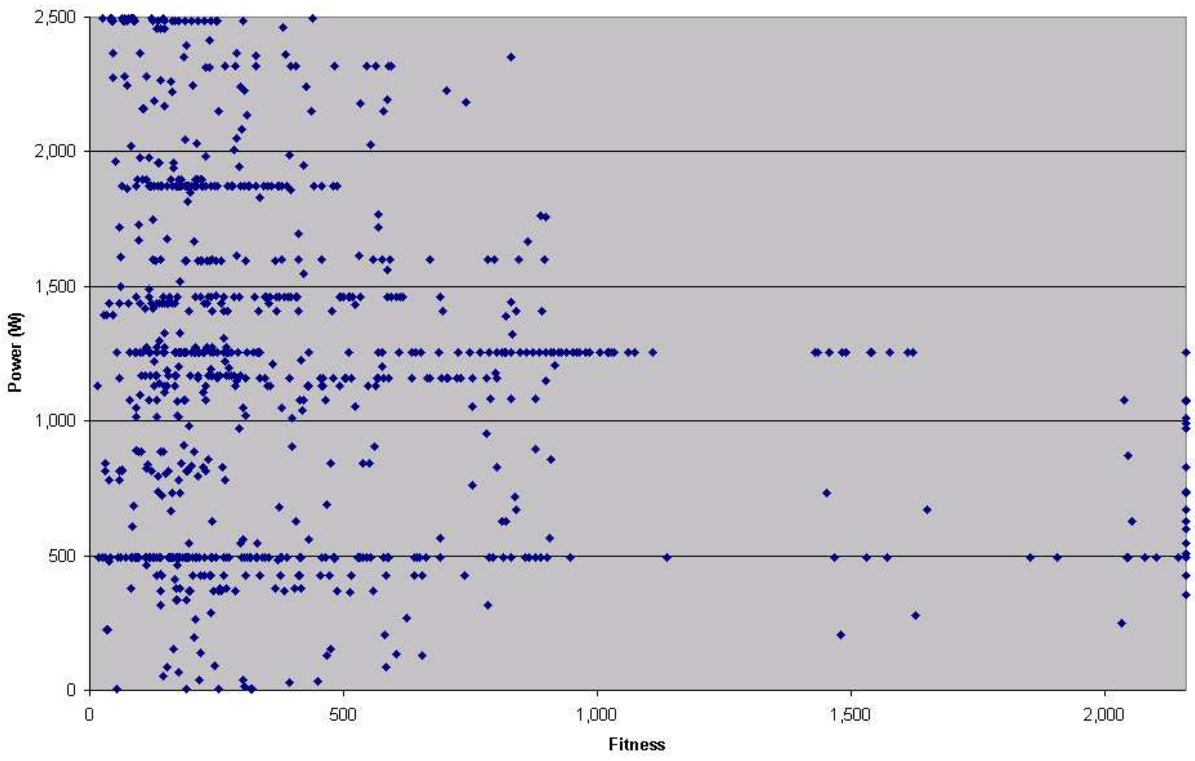


Figure 8. Although power issues are critical, it is possible to identify minimum power expenditures to ensure a viable system. However, a maximum threshold for power consumption also exists in this case. Configuration 5 run D.

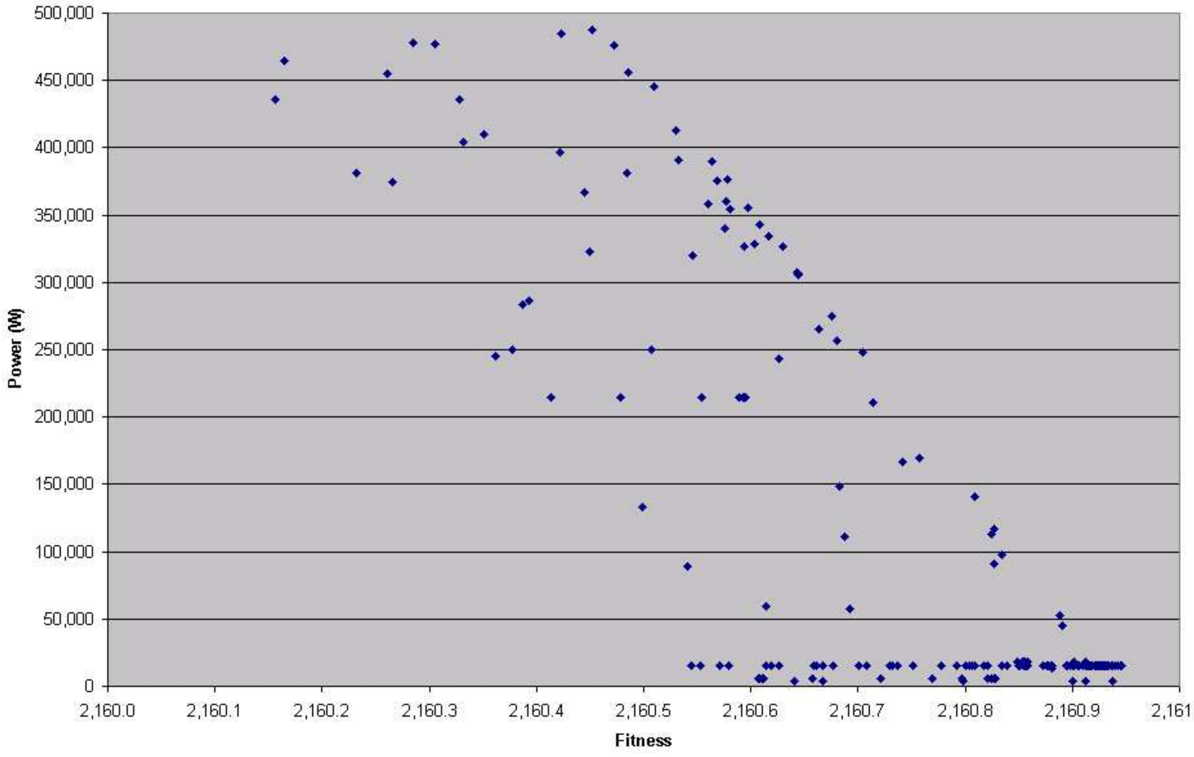


Figure 9. The fitness function is capable of identifying configurations which consume fewer resources. All configurations pictured complete the requisite 90-day mission. Evidently those systems with minimal power storage have higher fitness in configuration 2 run B.

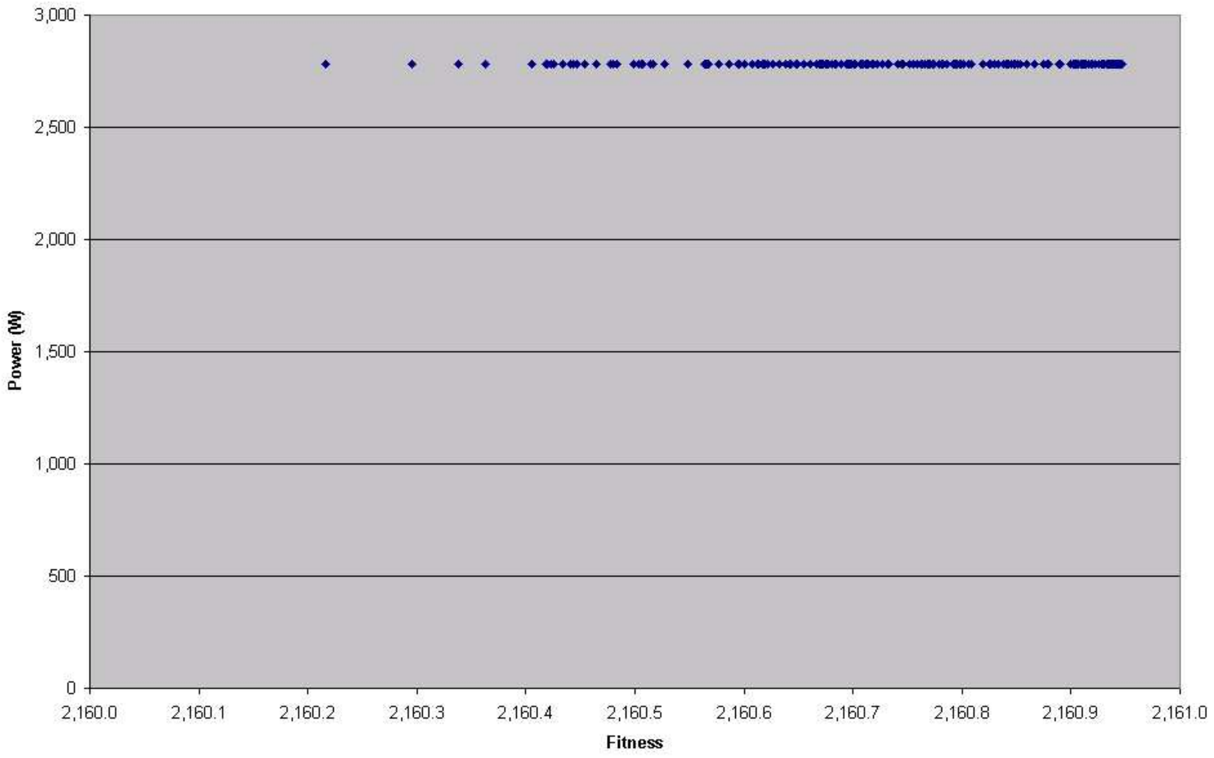
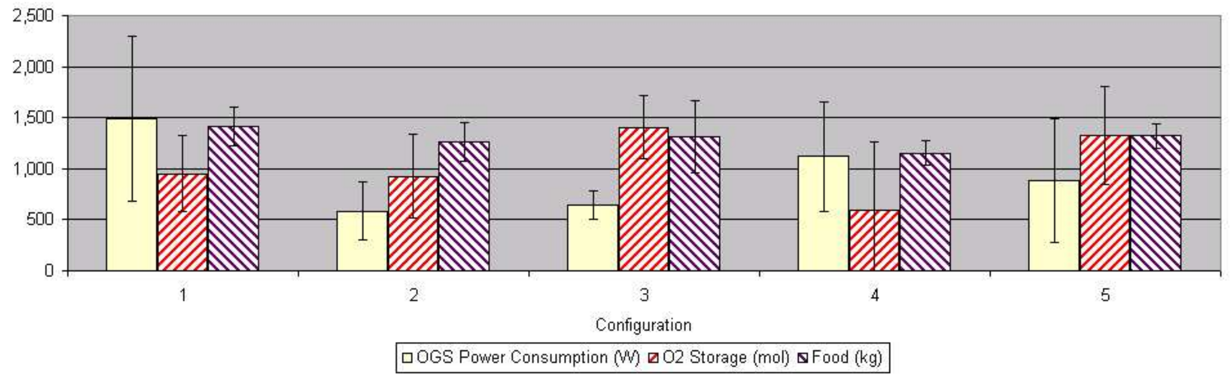


Figure 10. In configuration 4 run C, it can be seen that the VCCR can be sized very precisely. This is true for configurations 2-5.



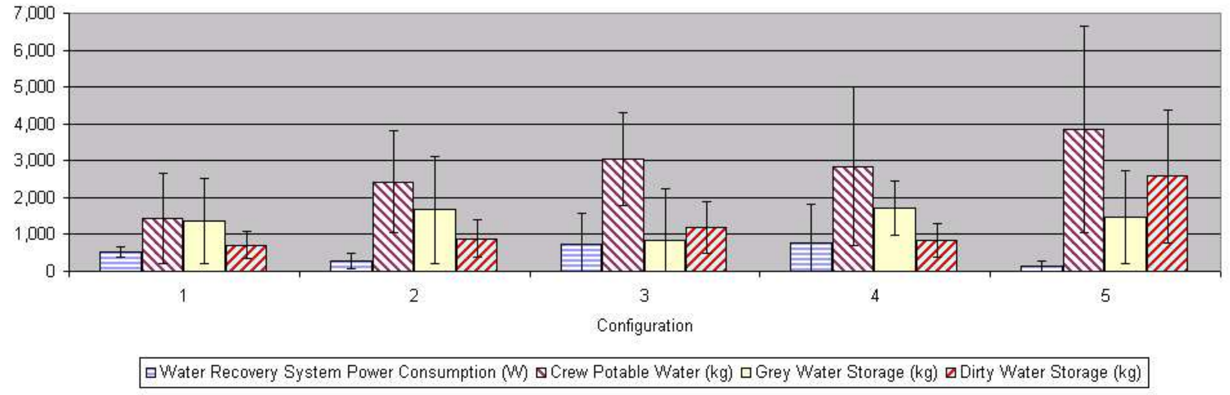


Figure 11. Sizing of various system components in each configuration is shown with 95% confidence intervals.

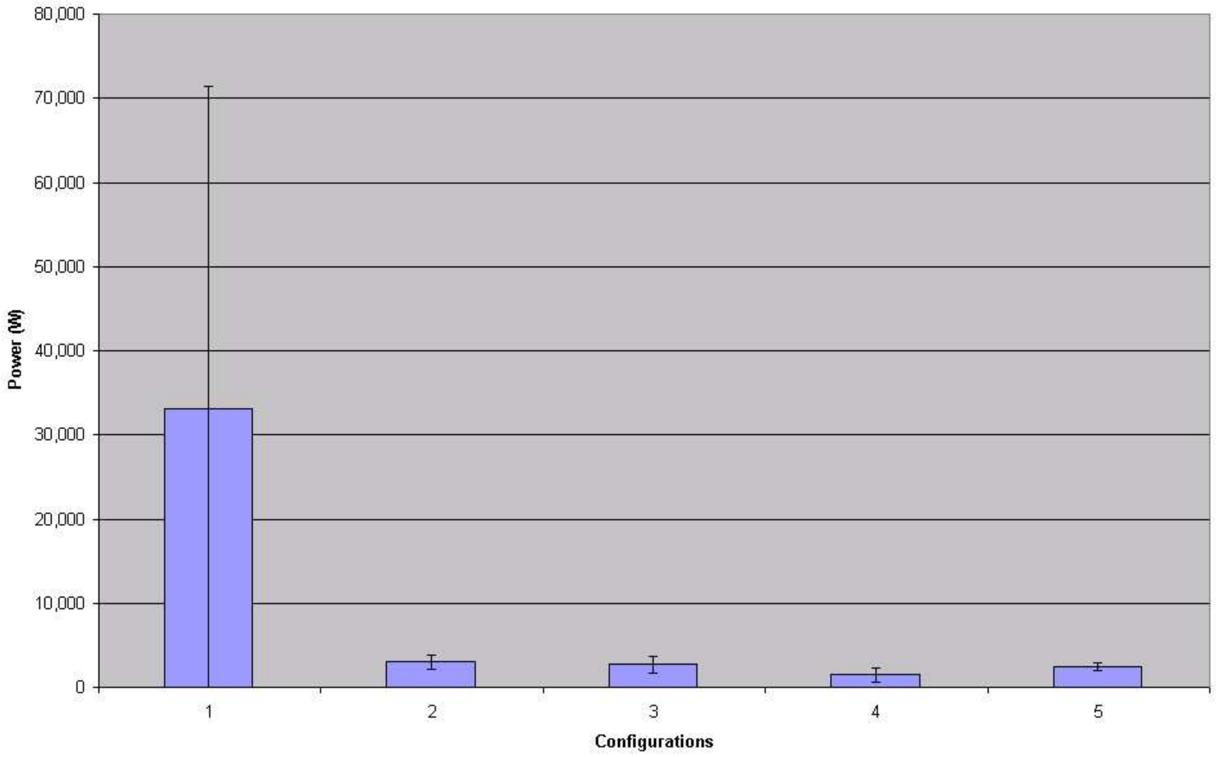


Figure 12. Average sizing and error bounds of the VCCR component in each configuration. The VCCR component was not precisely sized in configuration 1, but was very precisely sized in configurations 2-5.

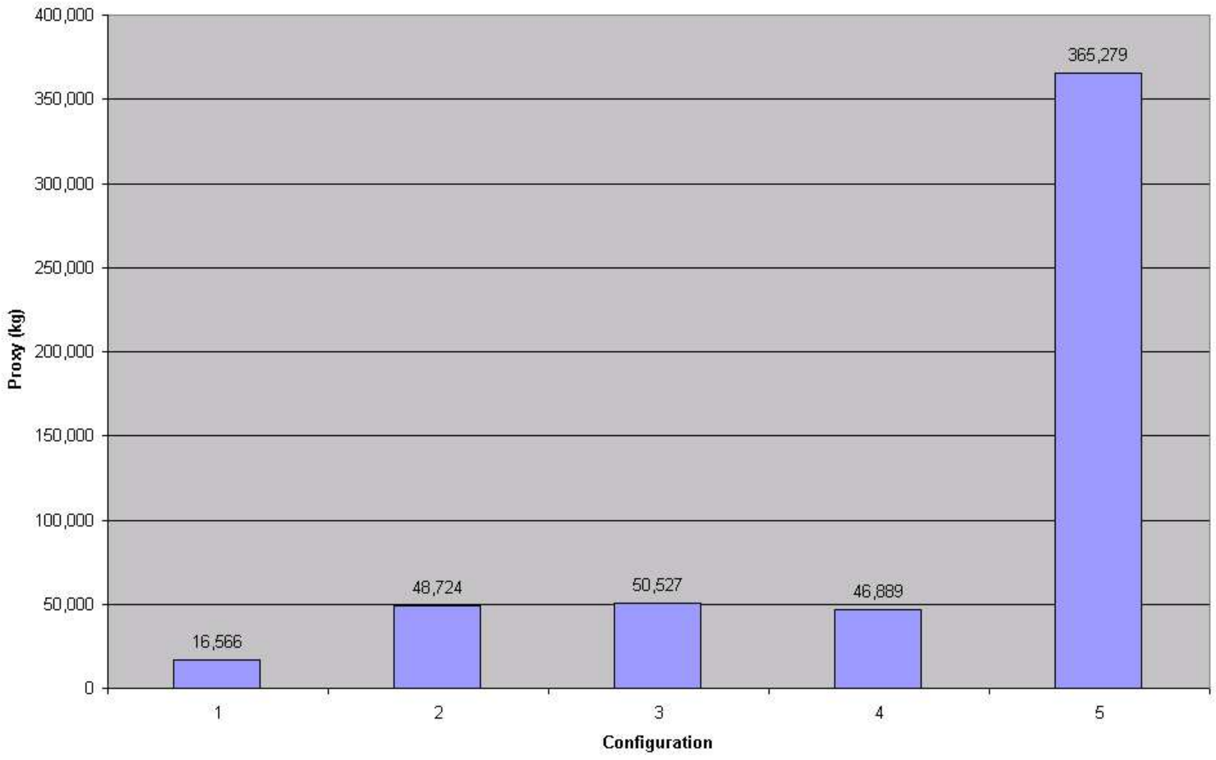


Figure 13. Fitness, being designed based upon ESM, can be utilized as a proxy for ESM. Each configuration is contrasted based upon this proxy.