

Evolutionary Computing for the Design Search and Optimization of Space Vehicle Power Subsystems

Abstract

Evolutionary computing has proven to be a straightforward and robust approach for optimizing a wide range of difficult analysis and design problems. This paper discusses the application of these techniques to an existing space vehicle power subsystem resource and performance analysis simulation in a parallel processing environment. Our preliminary results demonstrate that this approach has the potential to improve the space system trade study process by allowing engineers to statistically weight subsystem goals of mass, cost and performance then automatically size power elements based on anticipated performance of the subsystem rather than on worst-case estimates.

Mark Kordon, Gerhard Klimeck, David Hanks
Oct. 7, 2003

Evolutionary Computing for the Design Search and Optimization of Space Vehicle Power Subsystems

Mark Kordon
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91001
818-393-0476
Mark.Kordon@jpl.nasa.gov

Gerhard Klimeck
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91001
818-354-2180
gekco@jpl.nasa.gov

David Hanks
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91001
818-393-7210
David.Hanks@jpl.nasa.gov

Hook Hua
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91001
818 393-7182
Hook.Hua@jpl.nasa.gov

Abstract— Evolutionary computing has proven to be a straightforward and robust approach for optimizing a wide range of difficult analysis and design problems. This paper discusses the application of these techniques to an existing space vehicle power subsystem resource and performance analysis simulation in a parallel processing environment. Our preliminary results demonstrate that this approach has the potential to improve the space system trade study process by allowing engineers to statistically weight subsystem goals of mass, cost and performance then automatically size power elements based on anticipated performance of the subsystem rather than on worst-case estimates.

The process starts with Pre-Phase A where the goals and objectives of the mission are defined and several plausible mission concepts are created. These early mission concepts will trade off various elements in the design so that project managers can choose between different alternatives for mass, cost, performance and risk. This cycle of goal definition, mission concept creation and design trade study is repeated many times with each pass refining and improving the resolution of the design. The product of this process is a mission architecture characterized such that its effectiveness in achieving mission objectives can be properly evaluated. One important aspect of the mission architecture is the flight system.

TABLE OF CONTENTS

| | |
|--|---|
| | |
| 1. INTRODUCTION | 1 |
| 2. MMPAT OVERVIEW | 2 |
| 3. APPLYING EVOLUTIONARY COMPUTING..... | 2 |
| 4. EVOLUTIONARY COMPUTING ENVIRONMENT... | 4 |
| 5. RESULTS..... | 4 |
| 6. CONCLUSIONS..... | 8 |
| ACKNOWLEDGEMENTS..... | 8 |
| REFERENCES..... | 8 |

1. INTRODUCTION

At the Jet Propulsion Laboratory (JPL) the life cycle of a deep space mission normally goes through six phases, each culminating with a review by project management and its funding agencies [1]:

- **Pre-Phase A:** Advanced Studies
- **Phase A:** Mission & System Definition
- **Phase B:** Preliminary Design
- **Phase C:** Design & Build
- **Phase D:** Assembly Test & Launch Ops
- **Phase E:** Operations

The purpose of the space vehicle flight system is to transport the payload safely to its destination. Typically the flight system is composed of several subsystems [1]:

- Power Subsystem
- Command & Data Handling Subsystem
- Telecommunications Subsystem
- Propulsion Subsystem
- Mechanical Subsystem
- Thermal Subsystem
- Guidance Navigation and Control Subsystem
- Spacecraft Flight Software

Each subsystem is responsible for a particular function, such as electrical power distribution, and has design characteristics like solar array size, solar cell technology, secondary battery size and battery cell technology. Designing these subsystems to meet payload, trajectory, communication and activity requirements within the mass, cost and performance constraints of the project is vital for mission success and if this process can be automated, we can ensure consistent design quality while at the same time allow experts to spend less time on routine tasks and more time evaluating various design options.

¹ 0-7803-8155-6/04/\$17.00 © 2004 IEEE

² IEEEAC paper #1525, Updated October 7, 2003

This paper discusses how we applied evolutionary computing to a power subsystem simulation to automate the design search and optimize space vehicle subsystem elements for given set of project requirements and constraints. It will also show how this allows engineers to use anticipated performance of the subsystem rather than the usual worst-case estimates. The paper begins with a brief overview of the simulation used on this effort, the Multi-Mission Power Analysis Tool (MMPAT). It then discusses evolutionary computing and how it was applied to the MMPAT simulation. It goes on to describe features of the evolutionary computing framework, PERSON, and concludes with the results of this effort and outlines areas for future work.

2. MMPAT OVERVIEW

In order to develop a system that would allow engineers to statistically weight subsystem goals of mass, cost and performance then automatically size elements based on anticipated performance of the subsystem, we required a simulation that could seamlessly handle multiple mission types and phases, and that could be integrated with an optimizer in a parallel processing environment. More specifically, the simulation needed to be a multiplatform library deployment with all of its design characteristics and state variables parameterized, and accessible through an Application Programming Interface (API). The API would also need to allow the user to enter an activity plan and trajectory. Moreover, the simulation would need to use actual flight project data to quickly predict the resources and performance of the subsystem over the mission timeline, and would need to run in a closed loop manner with environment models that were, preferably, already integrated. Lastly, while not specifically required for this task, we wanted the simulation to be able to respond dynamically to inputs from other subsystems for compatibility with future research efforts. Given these requirements we choose to use the Multi-Mission Power Analysis Tool (MMPAT).

MMPAT is one tool in a suite of Multi-Mission Subsystem Analysis Tools at JPL [2]. It is a multiplatform software simulator currently used in Mars Exploration Rover (MER) operations to predict the performance and resources of space vehicle electrical power subsystems before a sequence of activities is uploaded. The simulation is variable fidelity and produces dynamic time and sequence dependent results rather than static point solutions. As such, it models the behavior of power sources and energy storage devices as they interact with the spacecraft loads and the environment over a mission timeline at a level of detail appropriate to each stage of the project lifecycle, which in MER's case is operations. The models in MMPAT include:

- Solar Array Model
- Solar Array Thermal Model
- Orbital Mechanics

- Astrodynamics Model
- Pointing Model
- Atmospheric Model
- Secondary Battery Model
- Secondary Battery/Thermostatically Controlled Heater Thermal Model
- Power Bus Model
- RTG Model
- Power Equipment List Model

All of the models were developed by power subsystem experts or adapted from validated heritage models. The tool itself comes with models for many of the most commonly used power sources, storage devices and power bus control methods used on space vehicles today. All of these models have been validated on previous or current missions, such as Pathfinder and MER, and give an accurate prediction of the system performance and resources.

The simulation is controlled by model parameters and was designed to be data-driven, modular and multiplatform. This means the models can be expanded to include additional hardware types. It also means that the application can be deployed stand-alone or as a library in another application, which in our case means integrated with an optimizer in a parallel processing environment. Moreover, the parameterized interface on MMPAT can also be used to change the mission type and analyze different mission phases since the tool supports the analysis of planetary landers, planetary orbiters, heliocentric orbiters and rovers as well as cruise, landed and orbiting phases and special events like flyby, TCM and EDL.

3. APPLYING EVOLUTIONARY COMPUTING

Evolutionary computing seeks an optimal solution for a given system by using a computer program to simulate the biological processes of natural selection [3,4]. This means that by using a process of random variation and selection through competition in an environment, the quality of solutions will iteratively improve. Simply put, the process involves generating a population of candidate solutions, evaluating how well they satisfy the requirements and constraints, then randomly mating the solutions to create children for the next generation. The selection of mates is weighted toward the better solutions so that they will have a reproductive advantage. Implicit in this process is the notion of a particulate mechanism of inheritance.

In biology, organisms have a genetic coding referred to as a genotype, and morphology, physiology and behavior known as a phenotype [5]. They are related to each other in that an organism's genotype describes, influences and controls its phenotype. This means that changing an organism's genes will change its function, structure or behavior, and will oftentimes affect several characteristics at once since genes are typically pleiotropic. So in our application the design

parameters are the genotype of the system, which succinctly describe and influence the structure and behavior of the subsystem or phenotype. Reproducing in this instance means contributing some design parameters from each parent to the child thus creating a combination of both of them that is hopefully better. This evolutionary process continues until some number of iterations has occurred or until the solution converges.

Given that we had an analysis tool that modeled the relationship of design parameters to the structure and behavior of the subsystem, and that it could be deployed as a library that we could send design parameters to, execute, and retrieve results from, our task consisted of:

1. Defining the parameters to operate on
2. Defining the objective function
3. Utilizing an evolutionary computing framework

For this initial prototype we selected three parameters to vary. They consisted of:

- Maximum Capacity of Secondary Battery (amp-hrs)
- Number of Cells Per String in the Solar Array
- Number of Strings Per Segment in the Solar Array

Where the capacity of secondary battery is a real number greater than zero and the number of cells and strings are integer values greater than zero. All other design parameters, such as Ni-H2 battery chemistry and triple junction solar cell technology, were set at initialization and remained fixed throughout the optimization.

The objective function is created to assign a non-negative figure of merit for a system, so that we can evaluate which solutions should have a reproductive advantage. For our application we developed this as a minimization function, decomposed as a superposition of simpler functions for cost, mass and performance. The battery and solar array cost functions attempt to minimize the cost of the system and get as close to free as possible. These equations are shown below:

$$\text{BattCostFit} = W_{bc}(\text{BatteryCapacity} * \text{CostPerAmpHr})^2 \quad (1)$$

$$\text{SACostFit} = W_{sc}(\text{NumberSolarCells} * \text{CostSolarCell})^2 \quad (2)$$

Where W_{bc} and W_{sc} are real numbers greater than zero that represents a statistical weight that can be applied to the cost fitness to adjust how steep the parabolic curve is, thereby adjusting the importance of this aspect of fitness. A similar set of equations were developed for the mass fitness except that here we were attempting to minimize the mass of the system to get as light as possible. These equations are shown below:

$$\text{BattMassFit} = W_{bm}(\text{BatteryCapacity} * \text{MassPerAmpHr})^2 \quad (3)$$

$$\text{SAMassFit} = W_{sm}(\text{NumberSolarCells} * \text{MassSolarCell})^2 \quad (4)$$

Where W_{bm} and W_{sm} are real numbers greater than zero that represents a statistical weight that can be applied to the mass fitness.

To determine how well the system performed we needed to run the MMPAT simulation by initializing it with design parameters, giving it an activity plan and trajectory, and then execute it while storing the battery state of charge (SOC) at each time step. Power subsystem engineers want to use as much of the battery as possible yet have the SOC remain above a certain minimum. They also do not want to charge the battery to full capacity since it is not energy efficient. To take these considerations into account we defined two performance functions, one that penalizes not using the battery effectively and one that rewards going toward a desired SOC. For the battery utilization function we simply stored the lowest SOC achieved over the simulation run and used this value in the performance equation shown below:

$$\text{PerformanceFit}_{\min} = W_{\min}(\text{SOC}_{\min} - \text{SOC}_{\min\text{allowed}})^2 \quad (5)$$

Where W_{\min} is a real number greater than zero that represents a statistical weight that can be applied to this minimum aspect of performance fitness. SOC_{\min} is a non-negative real in amp-hrs that is the lowest SOC achieved during the simulation run and $\text{SOC}_{\min\text{allowed}}$ is a user-defined non-negative real in amp-hrs representing the lowest desirable SOC. To reward the battery for approaching the target SOC we needed to take into account the fact that the maximum battery capacity may be different for any architecture. So we calculated the average fraction SOC over the entire run with the following equation:

$$\text{AverageSOC} = (1/N) \sum_{i=0}^N (\text{SOC}_i / \text{BatteryCapacity}) \quad (6)$$

Where N is the number of time steps, SOC_i is the battery state of charge in amp-hrs at time step i , and BatteryCapacity is the maximum battery state of charge possible in amp-hrs. Since we did not want to penalize a solution that went over the desired percent SOC, we made this function conditional as shown below:

if $\text{AverageSOC} > \text{FractionSOC}$

$$\text{PerformanceFit}_{\text{tgt}} = 0.0 \quad (7)$$

else

$\text{PerformanceFit}_{\text{tgt}} = W_{\text{tgt}}(\text{AverageSOC} - \text{FractionSOC})^2 \quad (8)$
 Where W_{tgt} is a real number greater than zero that represents a statistical weight that can be applied to target aspect of performance fitness and FractionSOC is user-defined real number between 0.0 and 1.0 that represents the target percentage that the user desires.

So given these simple functions, the objective function for the system is:

$$\text{ObjectiveFunction} = \text{BattCostFit} + \text{SACostFit} + \text{BattMassFit} + \text{SAMassFit} + \text{PerformanceFit}_{\min} + \text{PerformanceFit}_{\text{tgt}} \quad (9)$$

Now having defined the parameters to operate on and the objective function, we were ready to integrate MMPAT and the objective function into the PERSON evolutionary computing framework.

4. EVOLUTIONARY COMPUTING ENVIRONMENT

PERSON is an acronym for Parallel Evolvable and Revolutionary Synthesis and Optimization Environment. It provides a software framework that is suitable for the large class of design optimization problems. At its core, PERSON is an optimization package that follows a sequence of procedures common among all genetic algorithms. This sequence is outlined below and diagrammed in figure 1.

1. Model parameterization and gene encoding
2. Initialization of population
3. Evaluation of fitness function for population
4. Selection of subset of population
5. Reproduction through crossover and mutation
6. Evaluation of fitness function and convergence check

At the integration layer, PERSON uses the Python scripting language to facilitate the integration of new applications into the framework and to allow rapid development and tuning of the fitness functions without recompilation of the whole source code. PERSON also provides several encoding schemes that are alongside typical binary encodings as well as explicit real, integer, and exponential number encodings with mutation and crossover operations that can be customized to the physical problem. These encodings are handled by the I/O interface and can be changed on the fly.

Because the calculations of the fitness function involve computations that can be quite intensive, executing the evolutionary computing algorithm on massively parallel computers is essential for high-fidelity models. These points are encapsulated in PGAPack, a parallel genetic algorithm library [7]. This package consists of a set of library routines supplying the user multiple levels of control over the optimization process. The levels vary from default encodings, with simple initialization of parameters and single statement execution, to the ability to modify, at a low-level, all relevant parameters in the optimization process. User written routines for evaluation or crossover and mutation can also be inserted if necessary. The package is written using the Message Passing Interface (MPI) for parallel execution on a number of processors.

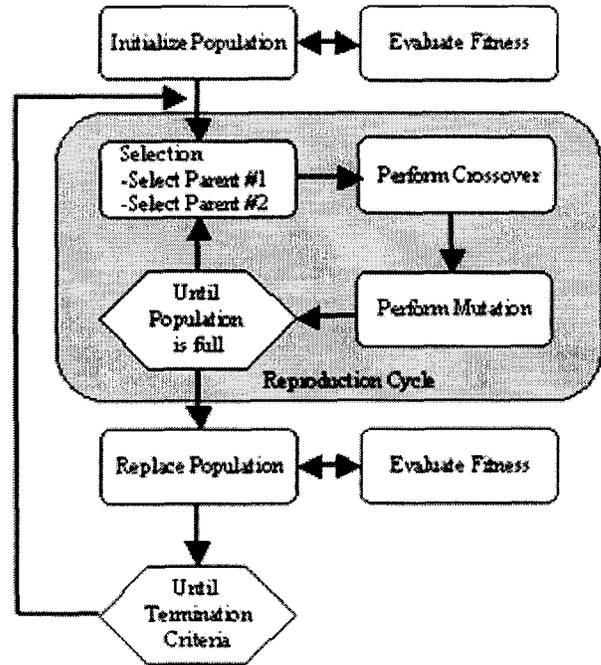


Figure 1 –Evolutionary Computing Flow Diagram [6]

5. RESULTS

We selected two power subsystem configurations to optimize: the cruise phase of Deep Impact, a mission set to launch in late 2003 or early 2004 and the surface operations phase of one of the Mars Exploration Rovers (MER), a mission set to land in the early part of 2004. To setup the analysis we had to give the PERSON framework some initial design parameters and a valid range of values as well as define the cost and mass constants, weight the objective function equations, create an activity plan and select a trajectory.

For Deep Impact, we varied the number of cells per string and number of strings per segment for two solar array segments as well as the battery capacity. The PERSON framework chose the initial population based upon a random draw over a uniform distribution for each of the variable power subsystem design parameters before invoking MMPAT. As a starting point, the framework was instructed to use an anticipated Deep Impact solar array size and battery capacity as shown below:

- Cells per string in segment one: 22
- Strings per segment in segment one: 44
- Cells per string in segment two: 16
- Strings per segment in segment two: 112
- Battery capacity in amp-hrs: 16.0

For the Deep Impact optimization the following intervals were used for each of the variable design parameters:

- Cells per string in segment one: [1, 50]
- Strings per segment in segment one: [1, 100]
- Cells per string in segment two: [1, 40]
- Strings per segment in segment two: [1, 200]
- Battery capacity in amp-hrs: [8.0, 40.0]

The following cost and mass values were used and remained fixed throughout the analysis:

- Solar cell cost: \$0.832k
- Solar cell mass: 0.01 kg per cell
- Battery cost:
- Battery mass: 0.01667 kg/watt-hr

For the trajectory, we decided to have the spacecraft start from Earth or 1.0 Astronomical Units (AU) and travel an ellipse to 1.5 AU, which is about the distance to where the spacecraft will encounter Comet Tempel 1. The orbital elements were set up in such a way that it would take approximately 8.3 months to traverse the distance. For the activity plan, we chose a simple constant load of 400 watts on the spacecraft and had the solar panels tilted at 23 degrees off normal simulating the anticipated cruise

configuration. We also simulated five trajectory correction maneuvers by having the MMPAT turn the solar array edge-on to the Sun for about three and a half minutes every 50 days. This had the effect of forcing the battery to be the sole source of power to the spacecraft during this time. The PERSON framework was instructed to run with this trajectory and activity plan for 2000 generations with a population size of 200. The framework was also instructed to use the anticipated power subsystem configuration as the starting point. Therefore, for each generation there were 200 invocations of MMPAT. Since it is possible to have parameterizations that did not crossover or mutate then approximately 70 percent, or 140, of the parameterizations were unique each generation. We ran two optimizations, one weighted more towards cost and mass, and the other weighted more towards the performance of the power subsystem. In the first cost and mass case, the following weightings were used and remained fixed throughout the analysis:

- Solar array cost: 1.0
- Solar array mass: 1.0
- Battery cost:
- Battery mass: 1.0
- Battery SOC min.: 1000.0
- Battery avg. SOC: 1000.0

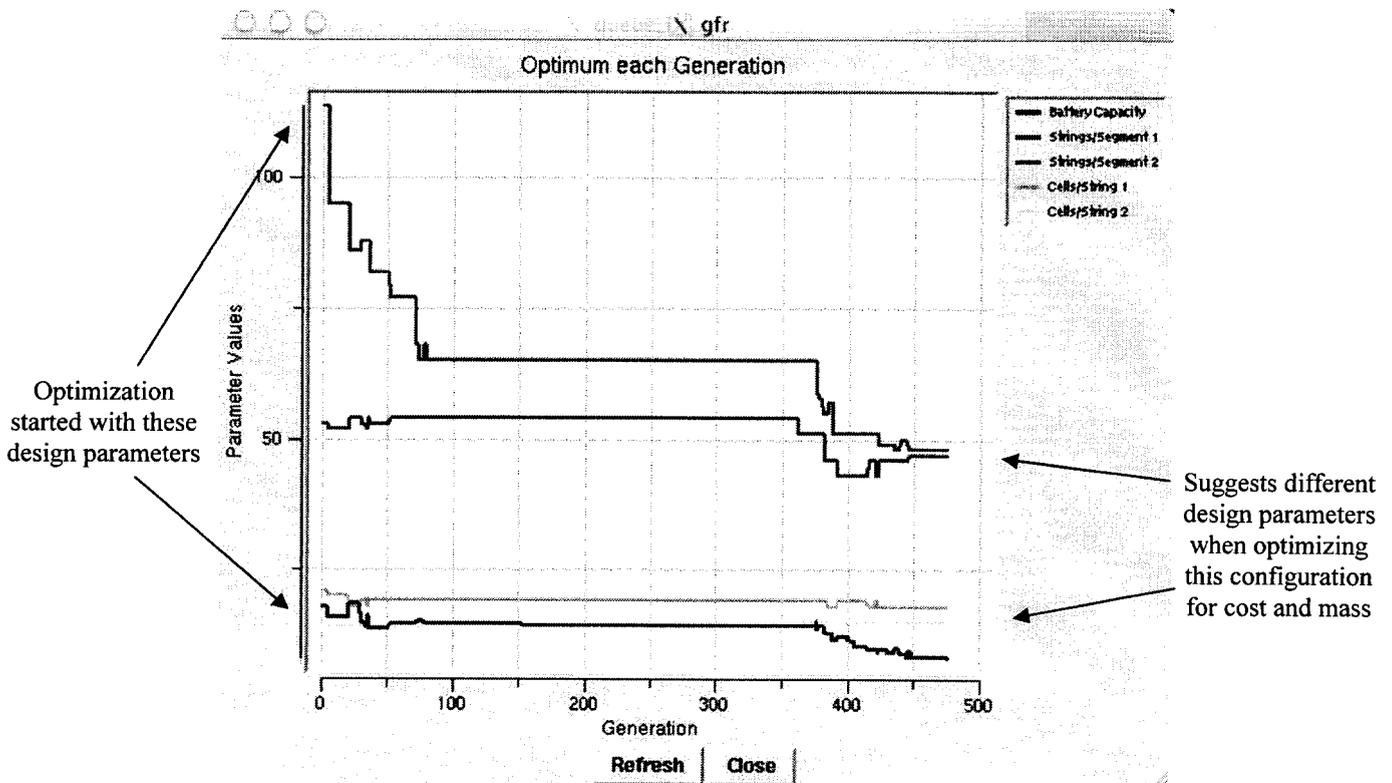


Figure 2 – Deep Impact Case 1 – Statistically Weighted for Cost and Mass

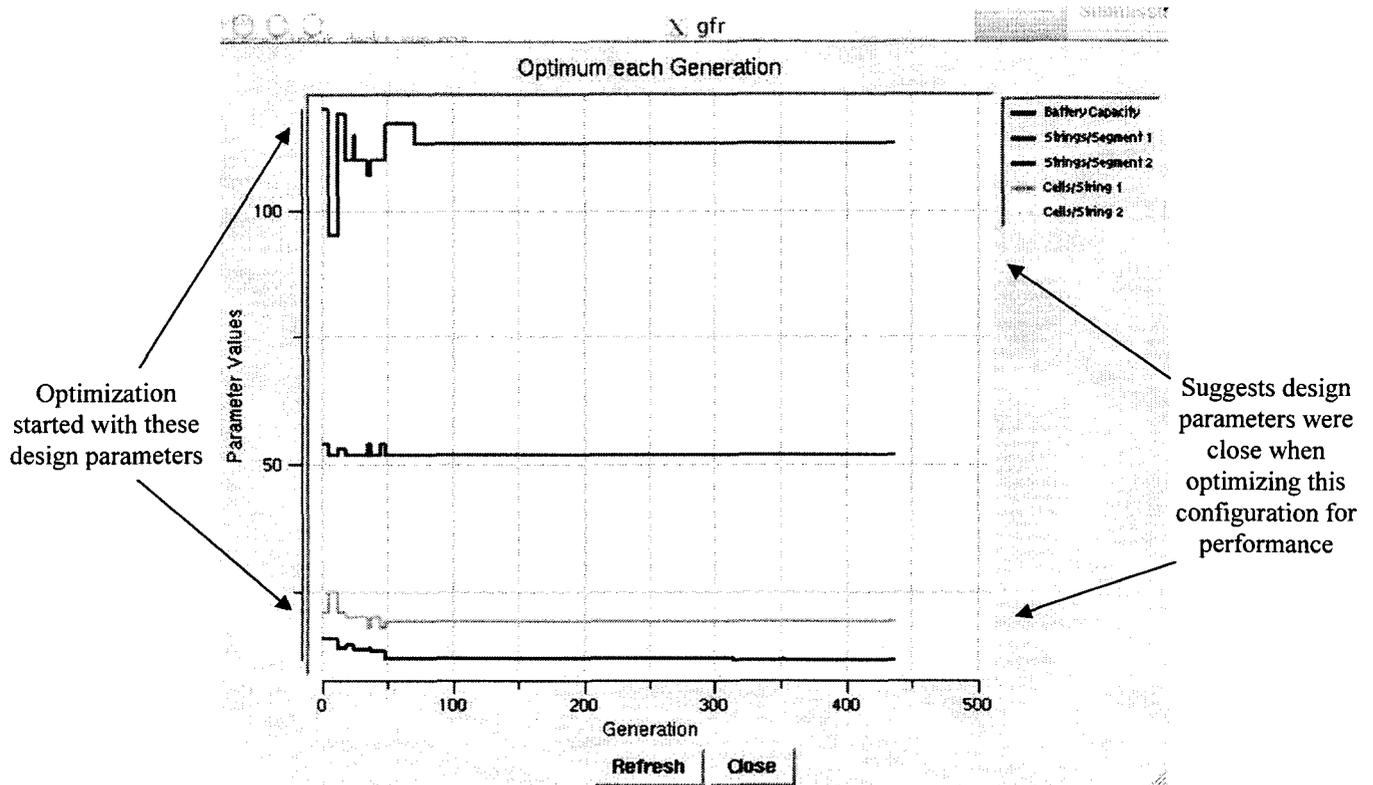


Figure 3 – Deep Impact Case 2 – Statistically Weighted for Performance

The optimization weighted more towards cost and weight resulted in both the solar array and battery size being reduced from the starting point. In figure 2, it can be seen how the configuration of the power subsystem changed as a function of generation. It is interesting to note that even though there is a period of about 250 generations where the configuration did not change, the evolution of the subsystem had not converged. This is indicative of a mutation breaking through a local optimum.

In the second analysis weighted more towards performance, the following weightings were used and remained fixed throughout the analysis:

- Solar array cost: 0.000001
- Solar array mass: 0.000001
- Battery cost:
- Battery mass: 0.1
- Battery SOC min.: 1000000.0
- Battery avg. SOC: 1000000.0

The size of the components did not change as much as in the analysis weighted more towards cost and weight. As can be seen from figure 3, it also converged much earlier. Both runs took approximately 8 hours each.

For the MER mission optimization we varied the number of cells per string and the number of strings per segment for the

six solar array segments as well as the battery capacity. As in the Deep Impact analysis, the PERSON framework chose the initial population based upon a random draw over a uniform distribution for each of variable power subsystem design parameters before invoking MMPAT. As a starting point, the framework was instructed to use an anticipated MER rover solar array size and battery capacity as shown below:

- Cells per string in all segments: 16
- Strings per segment in segment one: 4
- Strings per segment in segments two and three: 5
- Strings per segment in segment four: 6
- Strings per segment in segments five and six: 5
- Battery capacity in amp-hrs: 8.0

For the MER optimization the following intervals were used for each of the variable design parameters:

- Cells per string in all segments: [1, 40]
- Strings per segment in all segments: [1, 20]
- Battery capacity in amp-hrs: [4.0, 16.0]

The rover was placed at 7 degrees south latitude, which is one of the possible landing sites for the mission and given an activity plan that lasted 90 days, the planned length of surface operations for the rover. The activity plan consisted of applying a 15-watt load to the spacecraft during local

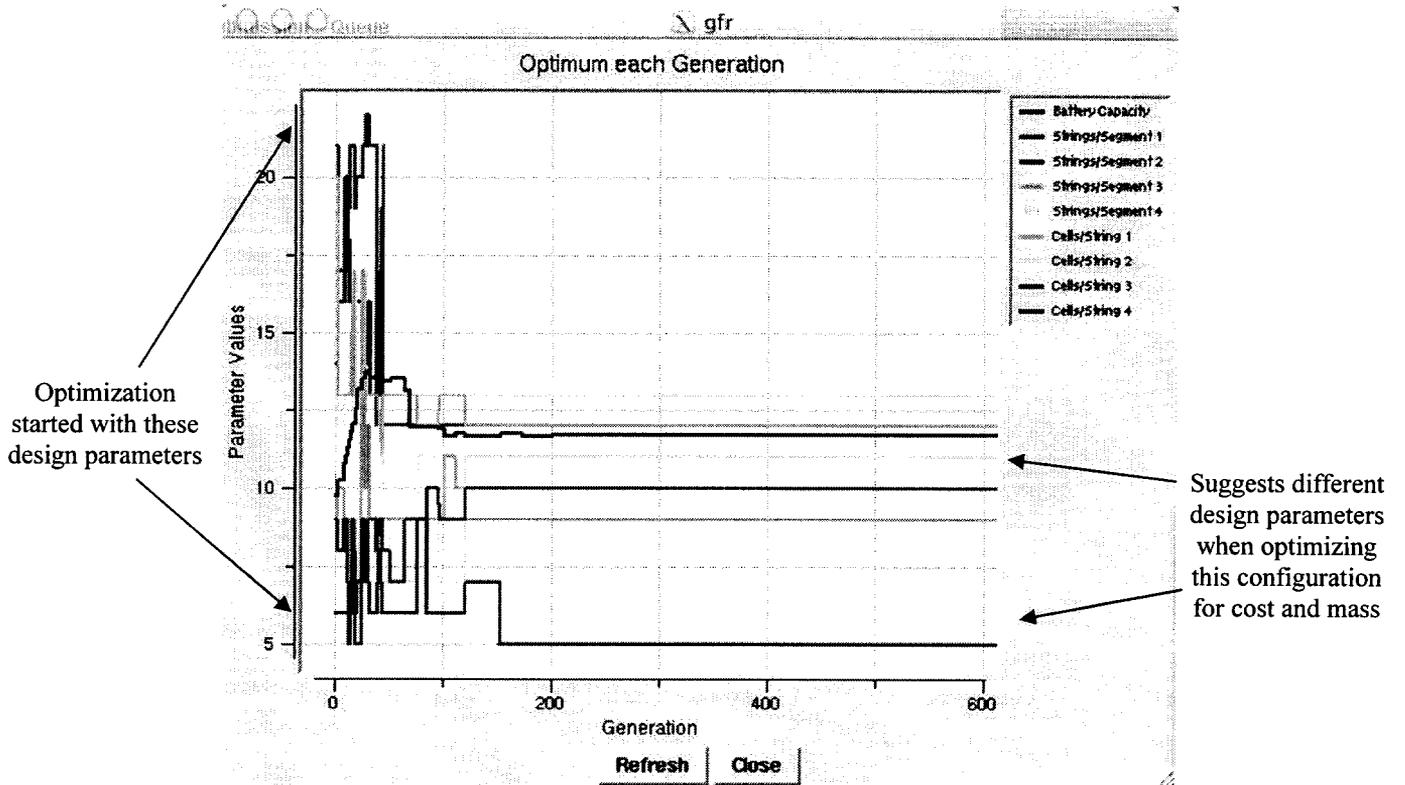


Figure 4 – MER Case 1 – Statistically Weighted for Cost and Mass

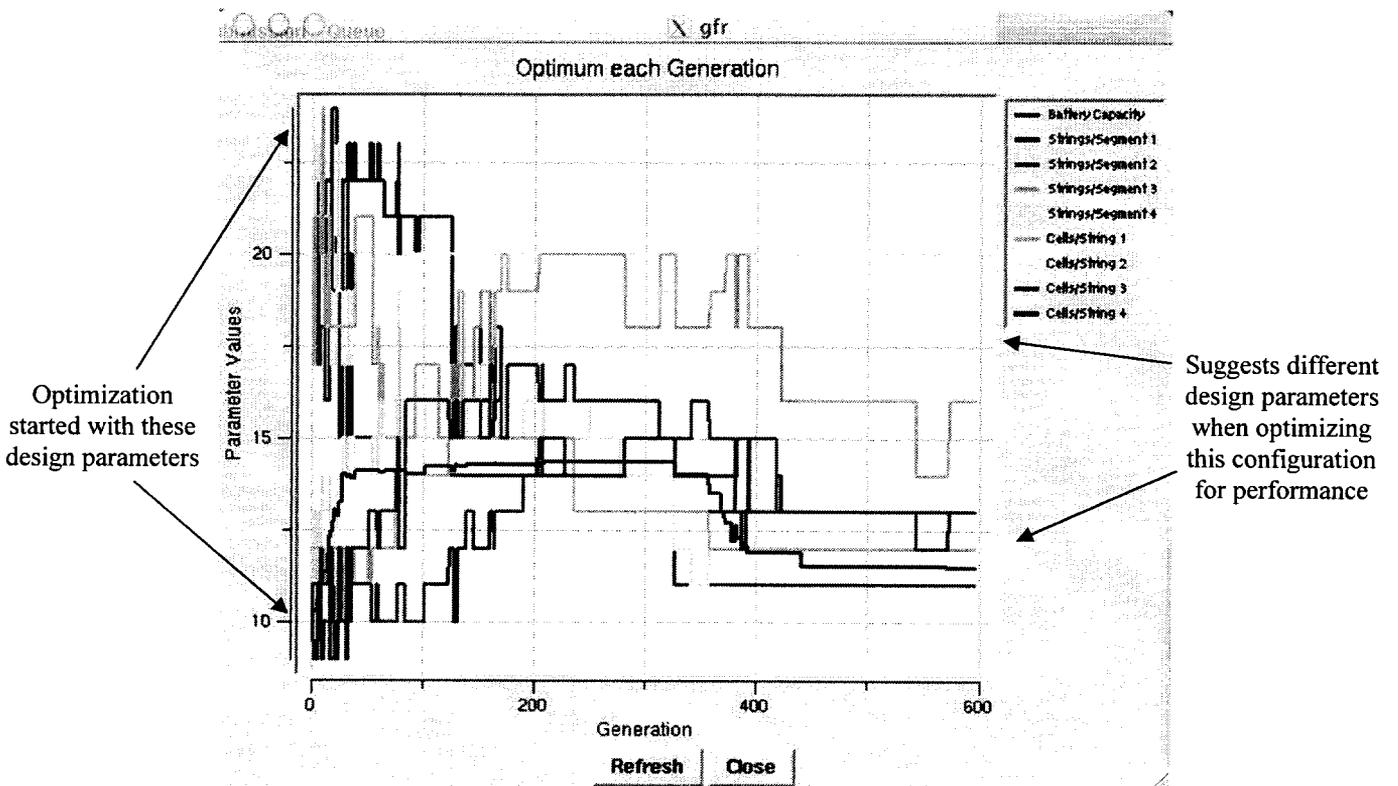


Figure 5 – MER Case 2 – Statistically Weighted for Performance

daytime for four hours a day and heater power for the remaining time. This simulated the load on the rover while it performed its duties during the day, and let the rover conserve battery power for the heaters at night. The same number of generations and population size were used as in the Deep Impact analysis. Once again, a possible power subsystem configuration was used as the starting point and two optimizations were performed, one weighted more towards cost and mass, and the other towards performance as show in figures 4 and 5, respectively. Each of these runs took approximately 16 hours to complete.

6. CONCLUSIONS

This work shows that evolutionary computing is a viable approach for the design search and optimization of space vehicle power subsystems and offers suggestions on how it may be introduced into the formulation phase of a flight project to improve the design quality and quite possibly lower the mission cost. It also illustrates that this methodology is practical in that it can be integrated with existing simulations and is relatively easier to implement than other methods like expert systems. Nevertheless, as promising as this field is, there is still much work to be done.

On the implementation side, this prototype needs to include more variable power subsystem design parameters such as battery and solar cell technology. Then the software needs to be infused into the formulation phase of a flight project and metrics on process improvements need to be collected. Next, the evolutionary methodology needs to be applied to other subsystems. Finally, all of the subsystems need to be integrated together to provide a complete solution.

On the theoretical side, the objective functions need to be scaled so that all of the equations use a similar weighting system. Also, an objective function for technical and mission risk will need to be developed and added. In addition, better objective functions for non-linear equations, like battery cost as a function of amp-hrs, will need to be integrated into the system. Finally, we will need to prove, theoretically and empirically, that the solutions we are producing are in fact optimal.

ACKNOWLEDGEMENTS

The work described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

REFERENCES

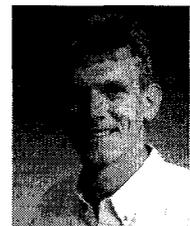
- [1] Robert Shishko, Robert G. Chamberlain, *NASA Systems Engineering Handbook*, NASA Publication SP-6105, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, second printing April 1996.
- [2] Kordon, M., and E. Wood, "Multi-Mission Space Vehicle Subsystem Analysis Tools". Proceedings IEEE Aerospace Conference, 2003.
- [3] Holland, J., *Adaption in Natural and Artificial Systems*. 1975, Ann Arbor: The University of Michigan Press.
- [4] Goldberg, D.E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley Longman, Inc., 1989
- [5] Daly, M., and M. Wilson, *Sex, Evolution and Behavior*, Second Edition. Wadsworth Publishing Company, Inc., 1978
- [6] Johnson, J. and Y. Rahmat-Samii, *Genetic Algorithms in Engineering Electromagnetics*. IEEE Antennas and Propagation Magazine, 1997. 939(4): p. 7-21.
- [7] Levine, D., *PGAPack Parallel Genetic Algorithm Library*. Techn. Rep. Argonne Natl. Lab, ANL-95/18, 1995.

Mark Kordon is the Technical Group Supervisor for the Modeling and Simulation Technologies Group, and Task Manager for Multi-Mission Analysis Tools at the Jet Propulsion Laboratory. His research interests include modeling and simulation techniques, evolutionary computing, multi-agent systems and space systems.



Mark conceived and coordinated the work described in this paper. He received his Bachelor of Science in Computer and Systems Engineering from Rensselaer Polytechnic Institute.

Gerhard Klimeck is the Technical Group Supervisor for the Applied Cluster Computing Technologies Group and a Principal Member at the Jet Propulsion Laboratory. His research interest is in the modeling of nanoelectronic devices, parallel cluster computing, genetic algorithms, and parallel image processing.



Gerhard developed the PERSON framework described in this publication. He also developed the 3-D Nanoelectronic Modeling tool (NEMO 3-D) for multimillion atom simulations and continues to expand NEMO 1-D. Previously he was a member of technical staff at the Central

Research Lab of Texas Instruments where he served as manager and principal architect of the Nanoelectronic Modeling (NEMO 1-D) program. Dr. Klimeck received his Ph.D. in 1994 from Purdue University and his German electrical engineering degree in 1990 from Ruhr-University Bochum. Dr. Klimeck's work is documented in over 90 peer reviewed publications and over 140 conference presentations. He is a member of IEEE, APS, HKN and TBP. More information about his work can be found at <http://hpc.jpl.nasa.gov/PEP/gekco>

David Hanks is a member of the technical staff in the Modeling and Simulation Technologies Group at the Jet Propulsion Laboratory. David was responsible for developing the software that allowed MMPAT to be integrated into PERSON. David also was responsible for running the analyses mentioned in this paper. He received his Bachelor of Arts in Mathematics and Bachelor of Science in Physics from the California State University, Fullerton.



Hook Hua is a software developer in the Applied Cluster Computing Technologies Group at the Jet Propulsion Laboratory. His contributions include the development of the C/C++ embedded Python datastore used in the GA architecture for dynamic scripting of science applications as well as flexible tuning of fitness functions. He also contributed to recent enhancements to the 3-D Nanoelectronic Modeling tool (NEMO 3-D). He received his degrees of Bachelor of Science in Computer Science and Bachelor of Science in Applied Mathematics from University of California, Los Angeles.

