JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 41, No. 1, January–February 2004

Multi-Objective, Multidisciplinary Design Optimization Methodology for Distributed Satellite Systems

Cyrus D. Jilla* and David W. Miller†
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

A multi-objective, multidisciplinary design optimization methodology for mathematically modeling the distributed satellite system conceptual design problem as an optimization problem has been developed. The tradespace for distributed satellite systems can be enormous, too large to enumerate, analyze, and compare all possible architectures. The seven-step methodology enables an efficient search of the tradespace for the best families of architectures during the conceptual design phase. Four classes of optimization techniques are investigated, Taguchi, heuristic, gradient, and univariate methods. The heuristic simulated annealing algorithm found the best distributed satellite system architectures with the greatest consistency due to its ability to escape local optima within a nonconvex tradespace. The conceptual design problem scope is then broadened by expanding from single-objective to multi-objective optimization problems, and two variant multi-objective simulated annealing algorithms are developed. Finally, several methods are explored for approximating the true global Pareto boundary with only a limited knowledge of the full design tradespace. In this manner, the methodology serves as a powerful, versatile systems engineering and architecting tool for the conceptual design of distributed satellite systems.

Nomenclature

A_v = system availability
C = capability
E = system energy
L = mission duration
N = number of dimensions in the objective function
P = candidate Pareto optimal set
P^* = true Pareto optimal set
p = state probability
T = system temperature
$\tilde{T}$ = arithmetic mean
t = time
U = utility
y = mission year
$\mathbf{\Gamma}$ = design vector
$\mathbf{\Gamma}_b$ = current baseline design vector
$\mathbf{\Gamma}_i$ = initial design vector
γ = design vector variable
Δ = objective function value difference
θ_r = angular resolution, marcsec
Φ = life-cycle cost, $
χ = random number between 0 and 1
Ψ = system performance
Ω = null depth

Introduction

Motivation

OPTIMIZATION is defined as the process of achieving the most favorable system condition on the basis of a metric or set of metrics. Within the past 50 years, different optimization techniques have been applied to numerous complex problems, ranging from the design of airline flight networks that maximize revenues[1] under scheduling constraints[2] to the allocation of assets in financial portfolios[3] under capital, regulatory, and risk constraints. This paper explores the potential of, and develops a framework for, the application of optimization techniques to the multidisciplinary conceptual design of distributed satellite systems.

The conceptual design of space systems currently tends to be unstructured, with designers often pursuing a single concept or modifying an existing idea rather than generating new alternatives. With the traditional aerospace point design approach, there is no guarantee that a system-level focus will be taken, and often the final design architecture chosen achieves only feasibility instead of optimality.[4] System-level trades are delayed until after a point design has been selected because of the perceived time and effort required to conduct a credible analysis.[5] When the system tradespace is not properly explored and an optimal or even efficient solution is not converged on during the conceptual design phase, the life-cycle cost of the system can greatly increase because modifications are required to integrate and operate properly the system during the latter stages of the design process, when design changes become much more expensive to implement.[6]

Figure 1 shows the importance of the decisions made during the conceptual design phase, also known as phase A, for a space system.[6,7] The International Council on Systems Engineering estimates that up to 75% of the development cost of a large system is predetermined after only 10–20% of the development time has been completed.[8] Thus, better systems engineering tools are needed to enable a more intelligent, thorough search of the tradespace during the conceptual design phase of a program. The methodology developed in this paper provides just such a tool to help systems engineers make better up-front design choices to improve the system performance and reduce the life-cycle cost of future distributed satellite systems.

The pitfalls in not following a structured process during conceptual design hold especially true for distributed satellite systems, which tend to be among the most complex and expensive space systems. A distributed satellite system (DSS) is defined as a system of multiple satellites designed to work together in a coordinated fashion to perform a mission.[9] Examples include the global positioning system for navigation, recently deployed low-Earth-orbit global mobile communications constellations, and proposed separated spacecraft interferometers for astronomy.

Distributed satellite systems are among the most challenging systems to design because a large number of highly coupled variables are involved. Take the example of a formation-flying separated

Received 16 January 2003; revision received 12 May 2003; accepted for publication 12 June 2003.

*Graduate Research Assistant, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue; cjilla@alum.mit.edu. Member AIAA.

†Associate Professor, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue; millerd@mit.edu. Member AIAA.

spacecraft interferometer, one of the architectures being considered for NASA's terrestrial planet finder (TPF) mission, intended to image extrasolar planets.[10] Both the total number of spacecraft in the array and the orbit of the interferometer drive the selection of the launch vehicle, often a dominant contributor to system cost. The number of spacecraft also directly determines the operations complexity and indirectly determines the imaging rate. The orbit also affects the imaging rate by determining the amount of local zodiacal dust the interferometer must peer through, and so forth. As in all DSSs, countless trades exist between system performance, system cost, and each of the design parameters. Typically, only a handful of point designs are derived from existing, previous generation designs. As a result, the final design is probably inefficient, leaving room for significant improvements in performance and reductions in life-cycle cost. Thus, a method is needed to enable a greater search of the tradespace and explore design options that might not otherwise be considered during the conceptual design phase.

Optimization is one such method. In its pure definition, optimization refers to finding the absolute best solution to a problem. This definition will not be used here, however. Rather, the engineering interpretation of optimization will be referred to as the process of finding good solutions with the intention of finding the best solutions to the conceptual design problem. Because DSS design problems tend to be combinatorial in nature with discrete variables in nonlinear relationships, linear optimization techniques that require continuously differentiable functions, such as the simplex method, cannot be used. Furthermore, DSS design problems are typically too large, with tradespaces approaching over a million architectures, for mixed integer programming methods. Rather, algorithms that can handle discrete variables in nonlinear problems with multiple criteria objective functions for multidisciplinary design optimization problems[11] are required. Different optimization techniques have been investigated for application to the design of interplanetary spacecraft,[12] small satellites,[5] and the Earth Observing System[13] with varying degrees of success.

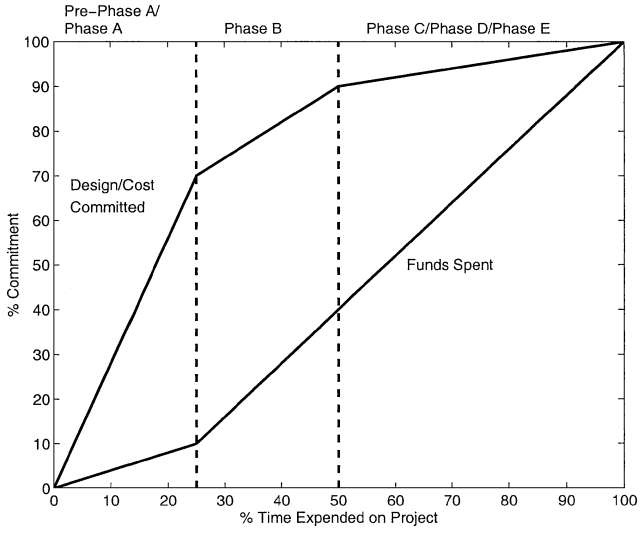


Fig. 1 Time expended vs funds/cost committed for a typical space project.[6,7]

The research presented in this paper develops and applies a methodology, the multiobjective, multidisciplinary design optimization systems architecting methodology (MMDOSA), for the conceptual design of distributed satellite systems. MMDOSA provides space systems engineers with a new supporting tool for the conceptual design of DSSs.

Objectives

The goal of this work is to demonstrate that DSS design problems can be formulated mathematically as optimization problems and solved to balance the system objectives and constraints in the conceptual design phase. The goal in applying optimization techniques to the design of DSSs is not to automate fully the design process and remove humans from the design loop, but rather to facilitate the conceptual design process.

Specific objectives of this work include first to develop a framework and methodology for mathematically formulating DSS conceptual design problems as optimization problems. The methodology will use optimization techniques to search efficiently the system tradespace to find regions likely to contain the best solutions based on the metric(s) of choice. The solutions to the optimization problem will then guide the systems engineer as to where to focus efforts during the next phase of work. The second objective is to advance the state of the art and develop specialized multi-objective optimization algorithms tailored to search the nonlinear, nonconvex tradespaces of DSSs.

The remainder of this paper develops and demonstrates the MMDOSA methodology for the conceptual design of DSSs. First, the operations research foundation of the work is defined, and the DSS conceptual design problem is formulated as an optimization problem. Second, the MMDOSA methodology is detailed step by step. Finally, the results and contributions this work has made to the field of space systems engineering and architecting are summarized.

Problem Formulation

Figure 2 maps the components of a standard optimization problem onto the formulation of the DSS conceptual design problem as a mathematical optimization problem. The decision variables in an optimization problem are the parameters whose values may be varied over a given range to minimize or maximize the desired function. Similarly, the design vector Γ in a DSS conceptual design problem contains the key independent design parameters the systems engineer has control over in developing a system architecture to accomplish the mission. Sample parameters γ_i that may be placed in the design vector include the number of satellites in the system, the orbital altitude of the satellites, and so forth. A unique set of values

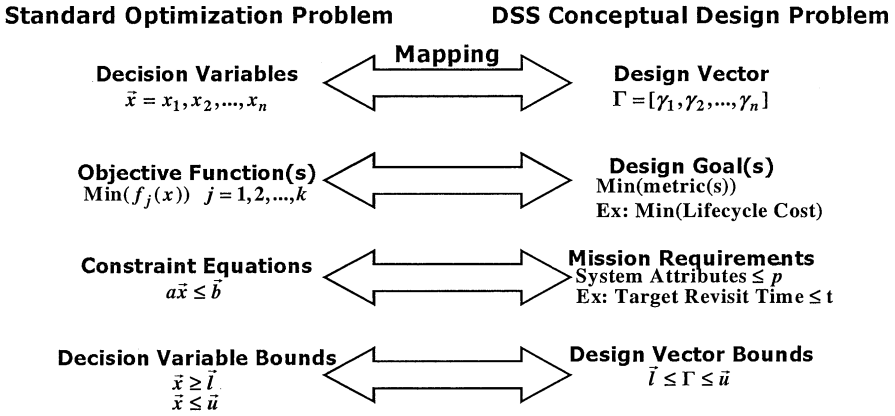


Fig. 2 Transformation between a standard optimization problem and the DSS conceptual design problem.

in the design vector defines a unique system architecture. Thus, the design vector represents the set of design variables with which the DSS architecture may be optimized. The remaining design parameters are placed in a constants vector. The constants vector contains parameters that influence the design of the architecture, but that are held constant when comparing different architectures. For example, the areal density of the satellite solar arrays is a parameter that is placed in the constants vector.

In an optimization problem, the decision variables combine with the cost coefficients to create the objective function to be optimized, that is, minimized or maximized. In a DSS conceptual design problem, the design goal(s) important to the customer are optimized. Such goals may be to maximize system performance, minimize life-cycle cost, maximize system reliability, etc. Herein lies an important difference between standard optimization problems and the DSS conceptual design problem. Whereas the objective function in a standard optimization problem may commonly be written and evaluated as a direct closed-form analytical relationship between the decision variables and cost coefficients, no such algebraic relationship exists between the variables in the design vector and the design goals in the DSS conceptual design problem. Rather, the relationship between the design vector and the mission goals may be thought of as a black box. The black box is actually a sequence of multidisciplinary mission models[14,15] that relate the input design and constants vectors to the output metrics and attributes. In other words, the black box is a multidisciplinary simulation of a single DSS architecture.

The constraint equations in a standard optimization problem place limits on certain combinations of the decision variables: Combinations of the decision variables that violate one or more constraint equations represent infeasible solutions. Likewise, mission requirements place limits on the allowable values for different mission attributes: System architectures that violate one or more of the mission requirements represent unacceptable architectures. Sample DSS design requirements might be a maximum allowable target revisit time or a minimum required signal-to-noise ratio (SNR) for a given payload measurement. Every mission requirement can be represented as a constraint equation in the problem formulation. Every architecture that is explored by an MDO algorithm will be checked against the entire set of constraint equations, that is, against the entire set of mission requirements. If any of the constraint equations are violated, then that system architecture is classified as an infeasible solution, and the algorithm proceeds to a new design.

The final set of constraint equations in the standard optimization problem place upper and lower bounds on each of the decision variables. Similar bounds may be placed on each of the design variables in the DSS design vector.

MMDOSA Methodology

The MMDOSA methodology entails seven steps. First, create the generalized information network analysis (GINA)[16] model: 1) define the mission objective and conceptual design phase objective, 2) transform the space system into an information network, 3) develop system metrics, 4) partition the conceptual design problem, 5) develop the simulation software, and 6) benchmark the simulation. Second, perform univariate studies: 1) vary one parameter within a baseline system architecture along a range of values and measure how the system attributes change, 2) develop an initial feel for the DSS local tradespace, and 3) identify model fidelity problems. Third, take a random sample of the tradespace: 1) obtain a glimpse of the global tradespace, 2) identify initial bounds for each optimization metric, and 3) obtain statistical data to tailor optimization algorithms. Fourth, formulate and apply optimization algorithms: 1) apply the single-objective optimization simulated annealing algorithm to identify the best family(s) of system architectures on the basis of the metric of interest and 2) apply the multiobjective optimization simulated annealing algorithms to identify the Pareto optimal set of system architectures with respect to all of the decision criteria. Fifth, interpret results, including sensitivity analysis: 1) identify the most influential design variables and 2) identify the most important high-fidelity models. Sixth, iterate: 1) increase the fidelity of critical models, 2) modify the simulated annealing algorithm cooling schedule and degrees-of-freedom parameter, 3) warm start a new optimization run, and 4) run additional trials of the same optimization algorithm. Seventh, converge on system architectures to be focused on in the next phase of the design process: 1) identify the best family(s) of system architectures from the single-objective optimization and 2) identify the Pareto optimal set of system architectures from the multi-objective optimization.

Table 1 Quality of service metrics for TPF

Capability	TPF equivalent
Isolation	Angular resolution, null depth
Integrity	SNR
Rate	Images per unit time
Availability	% Time in use

Step 1: Create the GINA Model

GINA for DSS is a systems engineering and architecting framework that enables the comparison of different design architectures for a given DSS mission.[16] The fundamental premise behind the GINA methodology is the assertion that most satellite systems are information disseminators that can be represented as information transfer networks.[9] Transforming the system representation from the physical domain to the information network domain enables the systems engineer to compare what physically appear to be very different architectures with the same set of quantitative metrics.

This new representation allows the systems engineer to use a body of mathematics that has been developed to analyze data networks. In information network theory, the capability of an architecture is characterized by four capability quality of service parameters: signal isolation, information rate, information integrity, and the availability of these services over time.[9] Once formulated, these four parameters serve as the minimum instantaneous capability requirements the DSS must meet to satisfy the customer. Table 1 identifies these metrics for the TPF mission. At this point, the systems engineer must decide which of these capability parameters will become hard requirements the system must meet to be considered feasible and which parameters will serve to distinguish architectures. For TPF, the isolation, integrity, and availability parameters are hard requirements every potential design must meet, whereas the imaging rate will differ between architectures.

Whereas the four capability quality of service metrics measure how well the DSS architecture meets the capability requirements at any instantaneous point in time, the performance metric measures how well the architecture satisfies the demands of the market over the entire life of the mission. The performance metric is computed by coupling the outputs of the capability model with the outputs of a Markov reliability model, which determines both the probability that the DSS will continue to function over a given amount of time and the likelihood with which the DSS will function in different partially failed states throughout the mission.[17] The coupling equation is called the utility function. A generalized utility function that may be applied to compute the performance of any DSS is

$$U = \int_0^L \sum_{i=1}^{n} C_i\, p_i(t)\, \mathrm{d}t \tag{1}$$

U is the total utility or performance, that is, number of images for TPF; L is the mission lifetime; n is the total number of operational states; C_i is the capability in each state i; that is, imaging rate for TPF; and $p_i(t)$ is the probability of being in each operational state i as a function of time t. The remaining GINA metrics for comparing DSS architectures are the life-cycle cost, cost per function, and adaptability metrics.[16]

Once the GINA metrics have been identified, they are matrixed against the design vector to determine how each design variable affects the system metrics to be measured.[18] In this manner, the models necessary to relate the design vector and constants vector to the metrics with which the DSS will be optimized are abstracted.

These models are then coded and integrated in a software environment to create a multidisciplinary system simulation whose outputs are the GINA metrics.

Step 2: Perform Univariate Studies

Once a GINA model has been developed for the DSS being designed and confidence in the output of the model has been established, the question arises of how to use the model to explore the DSS tradespace. The first method of tradespace exploration that systems engineers naturally tend to use is to establish a baseline design, vary one parameter within the baseline design along a range of values, and see how the attributes of the design change.[6] This method is known as a univariate study. Univariate studies investigate how a single system architecture parameter, that is, simulation output variable or GINA metric, varies as a function of a single input parameter, that is, design vector or constants vector variable. Univariate studies begin to give the systems engineer a feel for the tradespace and how different design decisions affect the tradespace.

Although useful, univariate studies by themselves do not give the systems engineer a complete view of the DSS tradespace. They fail to tell the entire story because they ignore the inherent design couplings that are present within every DSS conceptual design problem. In other words, because only one design parameter is varied at a time, no insight is provided as to how the system attributes change when two or more design parameters are varied simultaneously. In real-world design problems, systems engineers do not hold everything constant and vary only one parameter at a time, but rather trade multiple parameters against each other simultaneously. The real power of the MMDOSA approach lies in the ability of the system engineer to change multiple design parameters simultaneously, that is, vary the entire design vector at once, and see how the system attributes change, that is, how the global tradespace varies. This simultaneous variation of multiple parameters will be done by the optimization algorithms developed in step four of the MMDOSA methodology. Before these algorithms can be used, however, one must first take a random sample of the tradespace.

Step 3: Take a Random Sample of the Tradespace

A random sample provides the systems engineer with preliminary data on how the system architecture metrics vary throughout the global tradespace and, in accordance with the principles of random sampling theory, provides critical data required to formulate the DSS tradespace search algorithms developed in step 4.

A common problem in statistics is how to assess a particular distribution without complete knowledge of the distribution. This problem leads to the field of statistical inference, which provides a methodology for deriving data on a population, that is, the full set of architectures in the DSS tradespace, after measuring only a small subset, or sample, of the population. A random sample is a sample collected in such a way that every unit in the population is equally likely to be selected.[19]

In many cases, it is impractical to measure the entire population. For DSSs, the computation time required to assess accurately a single architecture makes it impractical, because of both computational time and memory constraints, to simulate, measure, and evaluate the full DSS tradespace. Therefore, the systems engineer must depend on a random subset of measurements from the complete tradespace to help make inferences concerning the global tradespace.

Within the TPF global tradespace of 640 architectures, 48 architectures, 7.5% of the tradespace, were randomly sampled via a Monte Carlo simulation. Figure 3 shows the results from the random sample for the cost per image metric, which the TPF design will be optimized with respect to. The data gathered in the random sample of the tradespace can be used to place initial bounds on each of the design parameters to be optimized. For example, the minimum cost per image (CPI) TPF architectures found via optimization algorithms should yield a lower CPI than the minimum CPI value of $474,000, which was the lowest CPI value measured in the random sample. The data gathered from the random sample will also be used in step 4 to set the T and Δ parameters. To summarize, random sampling provides the systems engineer with a glimpse of the global tradespace, initial bounds on each of the metrics the design will be optimized with respect to, and statistical data that will be used to tailor the optimization algorithms.

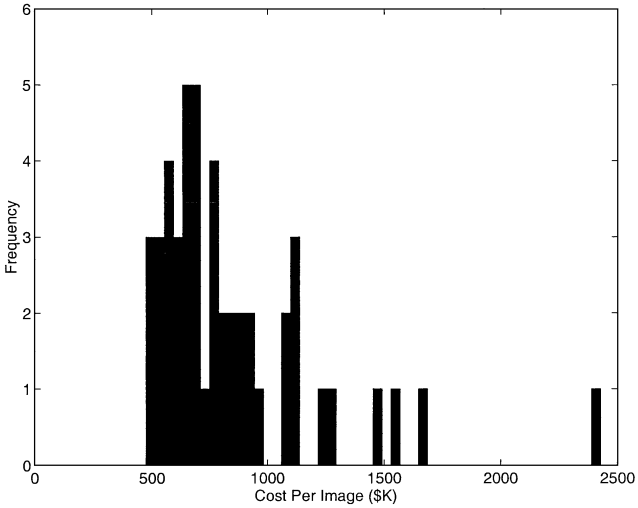


Fig. 3 TPF cost per image random sample.

Step 4: Formulate and Apply Optimization Algorithm(s)

Single-objective and multi-objective optimization algorithms form the core of the MMDOSA methodology. These algorithms help the systems engineer find the best architectures for a DSS based on the metric(s) of interest without having to resort to complete enumeration of the tradespace. This section develops and analyzes the performance of several optimization algorithms on both single-objective and multi-objective DSS conceptual design problems.

Single-Objective Optimization

In a single-objective optimization problem, one wishes to find the best DSS architecture on the basis of a single metric. The results from a single-objective optimization exercise should yield the most promising family(s) of system architectures to be carried forward into the next phase of the design process. This section summarizes the performance of several different single-objective optimization techniques on the single objective DSS conceptual design problem.

Comparison of Single-Objective Optimization Techniques

Four separate single-objective optimization techniques were investigated for their applicability to the conceptual design of DSSs. The four optimization techniques tested were Taguchi's method, heuristic simulated annealing, a pseudogradient search, and a univariate search. Each technique was applied to the problem of developing a system architecture that minimizes the CPI metric for the TPF mission.

The TPF mission is currently in the conceptual design phase, and several widely varying system architectures ranging from structurally connected to tethered to separated spacecraft arrays have been proposed. For the purpose of this case study, four design parameters are isolated as the key independent variables in the design problem: heliocentric orbital radius γ_1, collector connectivity/geometry γ_2, number of collector apertures γ_3, and collector aperture diameter γ_4. Together, these four parameters make up the TPF design vector $\boldsymbol{\Gamma}$:

$$\boldsymbol{\Gamma} = [\gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \gamma_4] \tag{2}$$

Table 2 lists the range of discrete values considered for each design vector variable γ in this case study. Each design vector defines a unique TPF mission architecture. In this case, the design vector was intentionally kept small to allow for the use of complete enumeration to verify the global optimum. Taking into account every possible

Table 2 TPF design vector

Design vector variable	Γ	Range	Discretization
Heliocentric orbital radius, AU	γ_1	1.0–5.5	0.5
Collector connectivity/geometry[a]	γ_2	Discrete	Unique
Number of collector apertures	γ_3	4–10	2
Diameter of collector apertures, m	γ_4	1–4	1
Total number permutations		640	

[a]Structurally connected one-dimensional or two-dimensional arrays, separated spacecraft one-dimensional or two-dimensional arrays.

combination of the variables in Table 2, the tradespace of this case study contains 640 different design vectors, or 640 unique TPF system architectures. This set of architectures is defined as the global tradespace.

The objective for each of the optimization techniques is to find the TPF architecture that minimizes the CPI metric. The only constraints in the problem are those on the capability isolation, integrity, and availability requirements and the allowable values for each variable in the design vector. The TPF conceptual design optimization problem may be represented mathematically as follows.

$$\begin{aligned}
&\text{Objective:} && \text{Min}\left[\frac{\sum_{y=1}^{5}\Phi_y(\mathbf{\Gamma})}{\sum_{y=1}^{5}\Psi_y(\mathbf{\Gamma})}\right]\\
&\text{Constraints:} && \text{Subject to}\\
&\quad\text{Isolation} && \theta_r \geq 20 \text{ marcsec}\\
&\quad && \Omega \leq 10^{-6}\\
&\quad\text{Integrity} && \text{SNR}_s \geq 5\\
&\quad && \text{SNR}_{\text{ms}} \geq 10\\
&\quad && \text{SNR}_{\text{ds}} \geq 25\\
&\quad\text{Availability} && A_v \geq 0.95\\
&\quad\text{Bounds} && 1.0 \text{ AU} \leq \gamma_1 \leq 5.5 \text{ AU}\\
&\quad && 4 \leq \gamma_3 \leq 10\\
&\quad && 1 \text{ m} \leq \gamma_4 \leq 4 \text{ m}
\end{aligned} \tag{3}$$

where AU stands for astronomical units, y is the year of the mission, Φ is cost, Ψ is the number of images, that is, surveys plus medium spectroscopies plus deep spectroscopies, SNR_s is the survey mode SNR, SNR_{ms} is the medium spectroscopy mode SNR, and SNR_{ds} is the deep spectroscopy mode SNR (all in decibels), and A_v is the system availability.

Experiment Results

After each of the four optimization techniques have been applied to the TPF conceptual design problem, their performance at finding good solutions after evaluating less than 8% of the total system tradespace may be analyzed. To determine the accuracy of these four optimization methods, the CPI metric was computed for all 640 TPF design vectors. When the best solution from each optimization method is compared with the true optimal solution found by complete enumeration, the performance of each optimization technique can be quantitatively assessed. Although complete enumeration to find the true optimal solution was possible for this controlled experiment, an expansion of the design vector $\mathbf{\Gamma}$, both by including more independent variables γ_i and considering more possible values for each independent variable, could quickly make full enumeration of all potential design vectors a computationally impractical approach. Typical DSS conceptual design problem tradespaces contain upwards of one million different architectures.[18] The lessons learned from exercising these four algorithms on this purposefully small design problem can be applied to larger DSS design problems in which the complete tradespace cannot be enumerated.

From complete enumeration of all 640 possible design vectors, the true optimal solution to the design problem was found to be an architecture with a structurally connected two-dimensional configuration located at 4 AU, with eight collector apertures, each of which is 4 m in diameter. This TPF architecture minimized the CPI metric at a value of $469,600 per image.

Table 3 Simulated annealing (SA), pseudogradient search (PGS), and univariate search algorithm (US) results, thousands of dollars

Trial	SA CPI	PGS CPI	US CPI
1	493.8	470.8	469.6
2	470.0	470.0	469.6
3	469.6	469.6	513.8
4	505.1	520.9	469.6
5	470.8	469.6	526.0
6	470.8	469.6	513.8
7	493.8	532.6	470.8
8	470.0	469.6	525.9
9	498.2	470.8	469.6
10	496.4	483.0	473.9
MSE[a] (thousands of dollars/image)^2	397.1	678.3	1027.8

[a]True optimal solution = $469,600.

The Taguchi analysis converged on an architecture with a CPI of $499,100, within 6.1% of the true optimal solution. Because the three remaining optimization techniques may vary as a function of their starting point, are probabilistic in nature, and can converge on local minima, multiple trials were executed for each technique. Each trial began with a different, randomly generated initial design vector. The mean square error (MSE), which measures how well each algorithm consistently finds the best solutions, may be computed as

$$\text{MSE} = \left[\sum_{i=1}^{n}\left(\text{CPI}_i - \text{CPI}^*\right)^2\right] \Big/ n \tag{4}$$

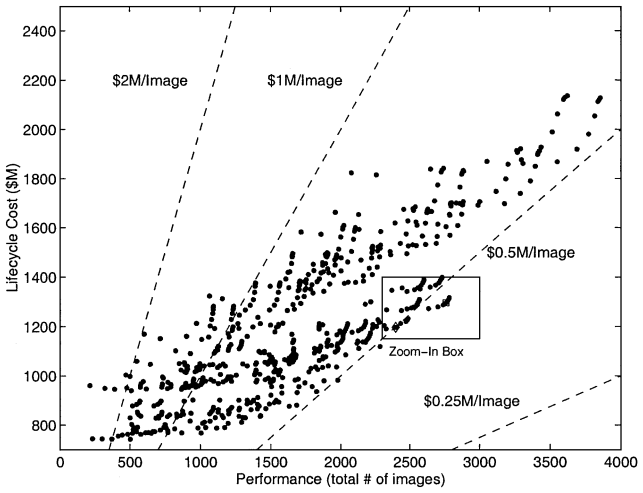
where each value of i represents a single trial of the optimization algorithm, n is the total number of trials, CPI_i is the cost per image of the best system architecture from trial i, and CPI^* is the cost per image of the optimal TPF architecture within the global tradespace. Table 3 lists the results of 10 trials for each algorithm. The heuristic simulated annealing approach exhibited the lowest MSE between the algorithm solutions and the true optimal solution and, thus, exhibits better performance than the pseudogradient and univariate search algorithms in terms of consistently finding the best TPF system architectures.

Figure 4a shows the global tradespace for the TPF conceptual design case study. The dashed diagonal lines represent lines of iso-CPI, that is, lines of constant CPI. All of the system architectures that fall on these lines exhibit the same cost effectiveness and, thus, are identical from the perspective of the objective function in Eq. (3), albeit with different levels of performance and life-cycle cost. Figure 4b shows the location of the true and Taguchi optimal solutions in the tradespace.

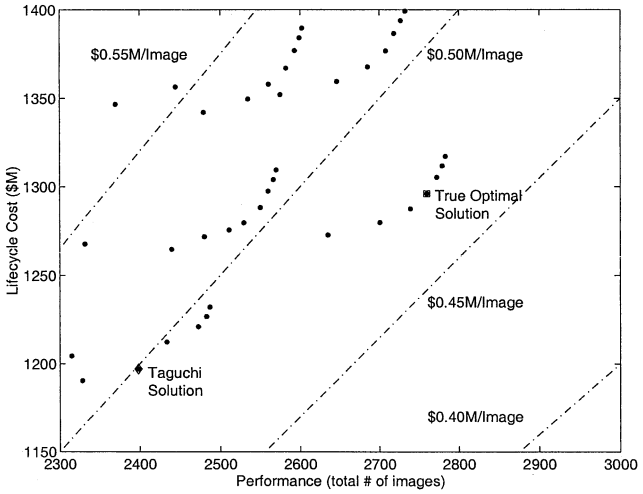
Also notice in Fig. 4b the existence of small arcs within the tradespace. Each arc may be thought of as a separate family of system architectures. Each arc contains a local minima with respect to the CPI metric. These local minima affect the solution of the DSS conceptual design optimization problem because the tradespace is nonconvex. In this case, the poor performance of the pseudogradient and univariate search algorithms as measured in Table 3 appears to be due to the susceptibility of these algorithms getting stuck in these local minima. Conversely, heuristics techniques such as simulated annealing are intended for nonconvex problems with multiple local minima,[20] and thus, the simulated annealing algorithm performs the best as measured by the MSE metric in Table 3. As a result of the observations from this experiment, the simulated annealing algorithm forms the core of all of the single-objective and multi-objective optimization algorithms used in the MMDOSA methodology.

Simulated Annealing

Simulated annealing is a heuristic technique developed in the early 1980s that mathematically mirrors the cooling of a material



a) TPF global tradespace



b) Tradespace zoom-in

Fig. 4 True and Taguchi optimal solutions.

to a state of minimum energy.[21] If a material cools too quickly, the crystals within the material end up in a suboptimal configuration. Likewise, the premise behind the simulated annealing algorithm is the assertion that if a solution is converged on too quickly that solution will be suboptimal.

First, the objective function, or system energy, $E(\mathbf{\Gamma})$ to be minimized by the algorithm must be defined, where $\mathbf{\Gamma}$ is the design vector:

$$E(\mathbf{\Gamma}) = f(\gamma_1, \gamma_2, \ldots, \gamma_n) \tag{5}$$

Each γ represents a different DSS design vector variable, such as the orbital altitude or total number of satellites. The algorithm begins with an initial state vector $\mathbf{\Gamma}_i$ containing randomly chosen values for each design vector variable within the bounds placed on that variable. The value of the objective function $E(\mathbf{\Gamma}_i)$ is then computed for this design vector. Next, the design vector is randomly perturbed to find a new design vector $\mathbf{\Gamma}_{i+1}$ in the neighborhood, where a neighbor is a design vector that shares some identical entries with the original design vector, of the current design vector, and the new $E(\mathbf{\Gamma}_{i+1})$ is computed. If $E(\mathbf{\Gamma}_{i+1}) < E(\mathbf{\Gamma}_i)$, then $\mathbf{\Gamma}_{i+1}$ is accepted as the new design vector. If $E(\mathbf{\Gamma}_{i+1}) > E(\mathbf{\Gamma}_i)$, $\mathbf{\Gamma}_{i+1}$ can still be accepted as the new design vector with a probability

$$\text{probability}(E) = \exp(-\Delta/T) \tag{6}$$

where

$$\Delta = E(\mathbf{\Gamma}_{i+1}) - E(\mathbf{\Gamma}_i) \tag{7}$$

T is a parameter that decreases as the number of iterations in the optimization increases. Thus, the likelihood of accepting a design vector with a greater system energy decreases over time. The ability to accept a less optimal $\mathbf{\Gamma}_{i+1}$ reduces the chance that the algorithm's solution will become trapped in local minima. This procedure is repeated until no new solutions are accepted after a specified number of iterations or until the algorithm is manually terminated. In theory, the algorithm converges to the optimal solution as the system temperature lowers toward zero. Unlike linear or integer programming methods in convex tradespaces, however, heuristic algorithm solutions in nonconvex tradespaces are not guaranteed to be globally optimal.

Table 4 SA algorithm

Step	Description
1	Choose a random design vector $\mathbf{\Gamma}_i$, select initial system temperature, and outline cooling schedule.
2	Evaluate $E(\mathbf{\Gamma}_i)$ via the GINA simulation model.
3	Perturb the current design vector $\mathbf{\Gamma}_i$ to obtain a neighboring design vector $\mathbf{\Gamma}_{i+1}$.
4	Evaluate $E(\mathbf{\Gamma}_{i+1})$ via the GINA simulation model.
5	If $E(\mathbf{\Gamma}_{i+1}) < E(\mathbf{\Gamma}_i)$, $\mathbf{\Gamma}_{i+1}$ is the new current design vector.
6	If $E(\mathbf{\Gamma}_{i+1}) > E(\mathbf{\Gamma}_i)$, then accept $\mathbf{\Gamma}_{i+1}$ as the new current design vector with a probability $\exp(-\Delta/T)$ where $\Delta = E(\mathbf{\Gamma}_{i+1}) - E(\mathbf{\Gamma}_i)$; otherwise, $\mathbf{\Gamma}_i$ remains the current design vector.
7	Reduce system temperature according to the cooling schedule.
8	Repeat steps 3–7 until the algorithm terminates.

Table 4 summarizes the steps in the simulated annealing algorithm. For the TPF case study, CPI replaces system energy as the metric to be minimized. Initial system temperature T_i was set to 1000 and was multiplied by a factor of 0.85 after each iteration. The algorithm was terminated after the completion of 48 iterations.

Multi-Objective Optimization

The preceding section considers only single-objective optimization problems. Whereas single-objective optimization provides a powerful tool to explore the tradespace of a DSS, there are times when several important decision criteria are present simultaneously and it is not possible to combine these criteria into a single number.[22] These problems are classified as multi-objective, or multicriteria, optimization problems, problems in which one attempts to optimize several separate criteria simultaneously. Such problems are important to the DSS conceptual design problem because true systems methods handle trades between multiple objectives, not just a single-objective function. In real systems engineering problems, one has to balance multiple requirements while trying to simultaneously achieve multiple goals. Therefore, any optimization methodology for DSS conceptual design needs to be able to handle multi-objective functions.

Differences Between Single-Objective and Multi-Objective Optimization

Several important differences exist between single-objective and multi-objective optimization problems. At this point, it becomes necessary to define some terminology, which will be done within the context of the example in Fig. 5.

Figure 5 shows the life-cycle cost vs performance for five different architectures of a separated spacecraft telescope system. A single-objective optimization problem is a problem in which one seeks the best, that is, highest or lowest, value of a well-defined objective. Three possible single-objective functions for the example DSS optimization problem are

$$\begin{gathered}\text{Minimize (life-cycle cost)} \\ \text{Maximize (system performance)} \\ \text{Minimize}\left(\frac{\text{life-cycle cost}}{\text{system performance}}\right)\end{gathered} \tag{8}$$

If the problem is convex for a minimization-objective function or concave for a maximization-objective function, there will exist only one true optimal solution to the problem. If the problem is nonconvex/nonconcave, there may exist more than one globally optimal solution, but each globally optimal solution will have the same

Table 5 Solutions to the single-objective separated spacecraft telescope design problem

Objective function	Solution	Optimal objective function value
Minimize (life-cycle cost)	Design 1	\$0.5 billion
Maximize (performance)	Design 3	2000 images
Minimize (cost per image)	Design 2	\$0.57 million/image

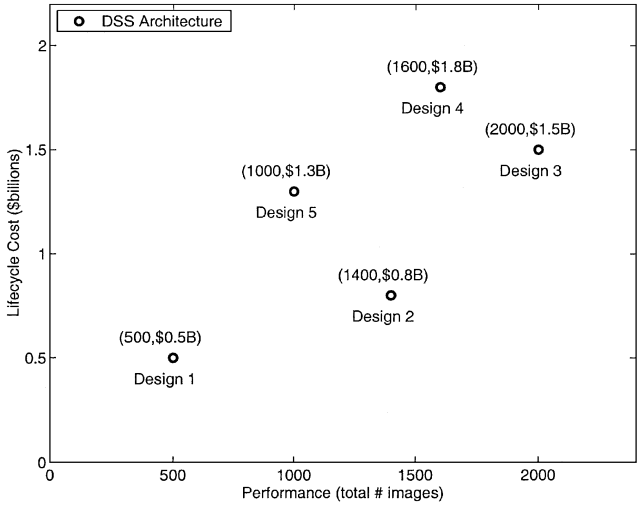


Fig. 5 Five theoretical separated spacecraft telescope design architectures (multi-objective).

objective function value. Table 5 lists the solution for each objective function in Eq. (8) to the sample problem in Fig. 5.

In step with the separated spacecraft telescope design problem, now let performance and life-cycle cost be separate goals in the same objective function. An ideal system design will maximize system performance while simultaneously minimizing life-cycle cost:

$$\text{Maximize (system performance) and}\quad \text{Minimize (life-cycle cost)} \tag{9}$$

Suppose that one is considering the five system architectures in Fig. 5, each with a different performance and life-cycle cost. At first glance, it is not clear which architecture is optimal according to Eq. (9). The notion of optimality in multi-objective problems is often not as obvious as in single-objective problems. Designs 1, 2, and 3 appear to be the best choices. However, none of these three designs is the best along both dimensions. In other words, there exist tradeoffs between each of these three system architectures. For example, moving from design 1 to designs 2 or 3 improves the performance metric, but at the expense of the life-cycle cost metric: Both metrics can not be improved simultaneously. In optimization terminology, these three architectures are defined to be nondominated, or noninferior, because there exist no better architectures than these for all decision criteria.[22]

On the other hand, designs 4 and 5 are poor choices because both of these architectures are dominated by other architectures within the tradespace. A dominated solution is a solution that is worse than at least one other solution in the tradespace for all decision criteria. Design 4 is dominated by design 3 because 1600 images < 2000 images and \$1.8 > \$1.5 billion. Likewise, design 5 is dominated by design 2 because 1000 images < 1400 images and \$1.3 > \$0.8 billion. Architectures 4 and 5 are clearly suboptimal and are classified as dominated, or inferior, architectures.

In this example problem, as in all multicriteria problems, instead of obtaining a single answer, a set of solutions {design 1, design 2, and design 3} have been obtained that are not dominated by any other solutions, the Pareto optimal (P-optimal) set. This solution set is a key point; Multi-objective problems can have more than one solution, whereas single-objective problems have only one true solution. Pareto optimality does not tell the systems engineer which architecture is best, but rather provides the systems engineer with a subset of the most efficient design solutions within the vast tradespace. Because the multi-objective problem formulation more accurately represents real-world DSS conceptual design problems, determining the P-optimal set within the vast DSS design tradespace will now become the focus of the MMDOSA methodology.

Searching for a Single Architecture in the P-Optimal Set

The first foray into applying multi-objective optimization to the conceptual design of distributed satellite systems involves searching for a single system architecture in the P-optimal set. Recall that Eq. (3) gives the mathematical formulation for the single-objective optimization instance of the TPF conceptual design problem. For the multi-objective case, this formulation must be rewritten as follows.

$$\begin{aligned}
&\text{Objective:} && \text{Min}\left[\sum_{y=1}^{5}\Phi_y(\boldsymbol{\Gamma})\right] \text{ AND Max}\left[\sum_{y=1}^{5}\Psi_y(\boldsymbol{\Gamma})\right]\\
&\text{Constraints:} && \text{Subject to}\\
&\quad\text{Isolation} && \theta_r \geq 20 \text{ marcsec}\\
&&& \Omega \leq 10^{-6}\\
&\quad\text{Integrity} && \text{SNR}_s \geq 5\\
&&& \text{SNR}_{\text{ms}} \geq 10\\
&&& \text{SNR}_{\text{ds}} \geq 25\\
&\quad\text{Availability} && A_v \geq 0.95\\
&\quad\text{Bounds} && 1.0 \text{ AU} \leq \gamma_1 \leq 5.5 \text{ AU}\\
&&& 4 \leq \gamma_3 \leq 10\\
&&& 1 \text{ m} \leq \gamma_4 \leq 4 \text{ m}
\end{aligned} \tag{10}$$

The decision logic for moving from one current baseline solution to a new current baseline solution within the simulated annealing algorithm must be defined. For the original single-objective minimization problem, this logic is

$$\begin{aligned}
&\text{IF } E(\boldsymbol{\Gamma}_{i+1}) < E(\boldsymbol{\Gamma}_i) \quad \text{OR} \quad \chi < e^{-\Delta/T}\\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_{i+1}\\
&\text{ELSE}\\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_i\\
&\text{END}
\end{aligned} \tag{11}$$

where $E(\boldsymbol{\Gamma}_{i+1})$ is the CPI of the new design vector, i is the iteration number within the simulated annealing algorithm, χ is a value created by a random number generator, and Δ is the difference in the objective function value.

For multi-objective problems, the decision logic when searching for a single point in the P-optimal set must be changed to

$$\begin{aligned}
&\text{IF } E_n(\boldsymbol{\Gamma}_{i+1}) < E_n(\boldsymbol{\Gamma}_i) \quad \text{for ALL} \quad n = 1, 2, \ldots, N\\
&\text{OR } \chi < e^{-\Delta/T}\\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_{i+1}\\
&\text{ELSE}\\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_i\\
&\text{END}
\end{aligned} \tag{12}$$

where $E_n(\boldsymbol{\Gamma}_{i+1})$ is the objective value along dimension n of the N-dimensional problem and Δ is the mean normalized difference[18] in the objective function value. The new architecture is accepted by

the simulated annealing algorithm as the new current baseline architecture if the new architecture rates better for all decision criteria than the current baseline architecture or randomly. For the multi-objective TPF example in Eq. (10), this translates into an architecture that has both a lower life-cycle cost and a greater performance, that is, produces more images, than the baseline architecture. This logic will be referred to hereafter as the multi-objective single-solution simulated annealing algorithm.

The decision logic in Eq. (12) holds only when the goal is to minimize all N decision criteria. In practice, any maximization problem may be rewritten as a minimization problem by multiplying the objective function by negative one. When the predefined nomenclature for the TPF case study is used, the decision logic in Eq. (12) may be written for the TPF multiobjective optimization problem [Eq. (10)] explicitly as

$$\begin{aligned}
&\text{IF } \sum_{y=1}^{5} \Phi_y(\boldsymbol{\Gamma}_{i+1}) < \sum_{y=1}^{5} \Phi_y(\boldsymbol{\Gamma}_i) \\
&\text{AND } \sum_{y=1}^{5} \Psi_y(\boldsymbol{\Gamma}_{i+1}) > \sum_{y=1}^{5} \Psi_y(\boldsymbol{\Gamma}_i) \\
&\text{OR } \chi < e^{-\Delta/T} \\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_{i+1} \\
&\text{ELSE} \\
&\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_i \\
&\text{END}
\end{aligned} \tag{13}$$

Figure 6 shows how the multi-objective single-solution simulated annealing algorithm searches the TPF tradespace with this decision logic. In Fig. 6, a zoom-in of the region within the global TPF tradespace where the algorithm has traversed is shown. The dashed line plots the path of the current baseline solution $\boldsymbol{\Gamma}_b$ in Eq. (13). The scatter in the path of the dashed line to worse solutions is due to the random moves, that is, third inequality in Eq. (13), of the algorithm. The solid line illustrates the path of the best solution, that is, the anticipated P-optimal system architecture. The vertical and horizontal lines illustrate the bounds the algorithm places on the tradespace on moving to each best solution, that is, the first and second inequalities in Eq. (13). These bounds constrain the algorithm to move toward the design architectures located in the lower right corner of the TPF tradespace as illustrated by the arrows emanating from the bounds. With the development of this new simulated annealing algorithm, there now exists a method for handling multi-objective DSS design problems.

Searching for Multiple Architectures in the P-Optimal Set Simultaneously

The multi-objective algorithm described in the preceding section finds only a single architecture in the P-optimal set during each run. In fact, the continual application of bounds on each objective function prevents the algorithm from finding many of the system architectures in the P-optimal set, depending on the location of the random initial starting point. It might be more efficient to search simultaneously for multiple points in the P-optimal set during the course of a single run of the simulated annealing algorithm. This concept will be the focus of the new simulated annealing algorithm developed here, referred to hereafter as the multi-objective multiple-solution simulated annealing algorithm.

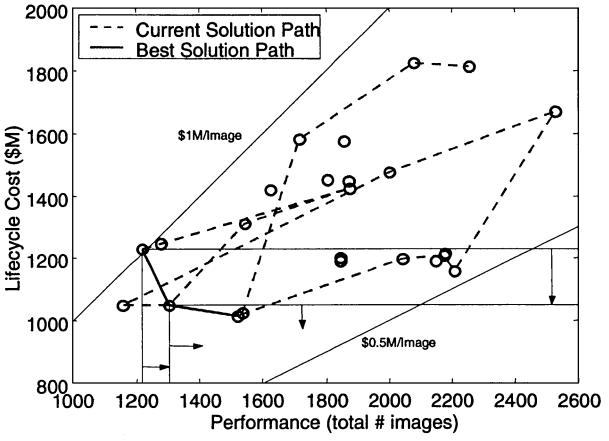


Fig. 6 Multi-objective single-solution simulated annealing search of the TPF tradespace zoom-in ($i = 48$ iterations).

To find multiple P-optimal design architectures at once, the simulated annealing algorithm derived in the preceding section will need to be modified in two major ways. First, the decision logic in Eq. (12) for changing the current baseline solution must accommodate moving in any direction toward and along the Pareto boundary/front. Second, all past design points traversed by the algorithm must be kept in memory so that every new system architecture may be checked against the past design points to compute the continually evolving P-optimal set.

For the TPF example, the problem formulation remains the same as stated in Eq. (10). During each step of the multi-objective multiple-solution simulated annealing algorithm, the newest design architecture is checked against the following decision logic, assuming the objective is to minimize all decision criteria, to determine whether or not the new design architecture is a candidate for Pareto optimality:

$$\begin{aligned}
&\text{IF } E_n(\boldsymbol{\Gamma}_k) < E_n(\boldsymbol{\Gamma}_i) \\
&\text{for ALL } n = 1, 2, \ldots, N \quad \text{and} \quad k = 1, 2, \ldots, i-1 \\
&\quad \boldsymbol{\Gamma}_i \notin P \\
&\text{ELSE} \\
&\quad \boldsymbol{\Gamma}_i \in P \\
&\text{END}
\end{aligned} \tag{14}$$

where $\boldsymbol{\Gamma}_k$ represents all of the previous points, that is, system architectures, traversed by the algorithm; i is the current iteration number within the simulated annealing algorithm; N is the number of dimensions, that is, number of decision criteria, design goals, or objective functions, in the conceptual design problem; and P is the candidate set of P-optimal architectures. Equation (14) states that if the new system architecture $\boldsymbol{\Gamma}_i$ is dominated by any of the existing system architectures $\boldsymbol{\Gamma}_k$, then the new architecture is not a candidate for Pareto optimality. If, however, the new architecture $\boldsymbol{\Gamma}_i$ is not dominated by any of the other architectures $\boldsymbol{\Gamma}_k$ that have been searched up to iteration i within the algorithm, then it is a candidate for Pareto optimality.

Once all of the iterations within the simulated annealing algorithm have been completed, the true P-optimal set P^* is extracted from the candidate P-optimal set P in the following manner:

$$\begin{aligned}
&\text{FOR } k = 1{:}i \\
&\quad \text{FOR } z = 1{:}i \\
&\quad\quad \text{IF } E_n(\boldsymbol{\Gamma}_z) < E_n(\boldsymbol{\Gamma}_k) \quad \text{for ALL} \quad n = 1, 2, \ldots, N \\
&\quad\quad\quad \boldsymbol{\Gamma}_k \notin P^* \\
&\quad\quad \text{ELSE} \\
&\quad\quad\quad \boldsymbol{\Gamma}_i \in P^* \\
&\quad\quad \text{END} \\
&\quad \text{END} \\
&\text{END}
\end{aligned} \tag{15}$$

where i is the size, that is, the total number of system architectures, of P. Equations (14) and (15) detail how the P-optimal set of architectures is extracted during a single run of the multi-objective multiple-solution simulated annealing algorithm. The actual path the simulated annealing algorithm takes through the tradespace from one current baseline architecture $\boldsymbol{\Gamma}_b$ to another current baseline architecture still incorporates randomness to prevent entrapment in

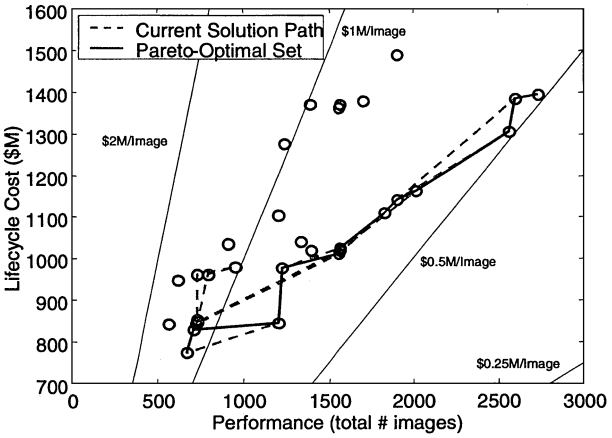


Fig. 7 Multi-objective multiple-solution simulated annealing search of the TPF tradespace zoom-in ($i = 48$ iterations).

local minima and follows the decision logic:

$$\begin{aligned}&\text{IF } \boldsymbol{\Gamma}_{i+1} \in P \quad \text{OR} \quad \chi < e^{-\frac{\Delta}{T}}\\ &\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_{i+1}\\ &\text{ELSE}\\ &\quad \boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_i\\ &\text{END}\end{aligned} \tag{16}$$

where i is the current iteration number within the simulated annealing algorithm.

Figure 7 shows how the multi-objective multiple-solution simulated annealing algorithm searches the TPF tradespace with the new decision logic outlined in Eqs. (14–16). Figure 7 shows a zoom-in of the region within the TPF tradespace where the algorithm has traversed. The dashed line shows the path of the current baseline solution $\boldsymbol{\Gamma}_b$ in Eq. (16). Points without a solid line passing through them represent system architectures that are dominated by one or more other system architectures. Points with a solid line passing through them represent the system architectures that comprise the P-optimal set P^*. The solid line connecting these points defines the P-optimal boundary, also known as the P-optimal front, of this explored subset of the TPF tradespace.

As before, the path of this new multi-objective multiple-solution simulated annealing algorithm will depend on the initial starting point. However, unlike the earlier multi-objective single-solution simulated annealing algorithm, this algorithm does not use bounds to eliminate different portions of the tradespace from consideration. This in turn enables the algorithm to find many P-optimal system architectures during a single run as shown in Fig. 7. In fact, the efficiency of this new approach is illustrated by the close concurrence between the current solution (dashed line) and best solution (solid line) paths, especially when compared with the concurrence, or lack thereof, in the multi-objective single-solution simulated annealing algorithm shown in Fig. 6.

Two separate heuristic simulated annealing algorithms have now been developed to search for P-optimal system architectures within the conceptual design tradespace of any DSS. However, how does the systems engineer know how good the answer for the true P-optimal set P^* is? This question will be the subject of the next section.

Approximating the True Global Pareto Boundary with an Estimated Pareto Boundary

To assess how well an estimated Pareto boundary, that is, the Pareto boundary developed from looking at only a portion of the tradespace, approximates the true global Pareto boundary, that is, the Pareto boundary developed from looking at the entire tradespace, a mean error (ME) metric will be used. The process involves five steps. First, compute the true global P-optimal set by completely

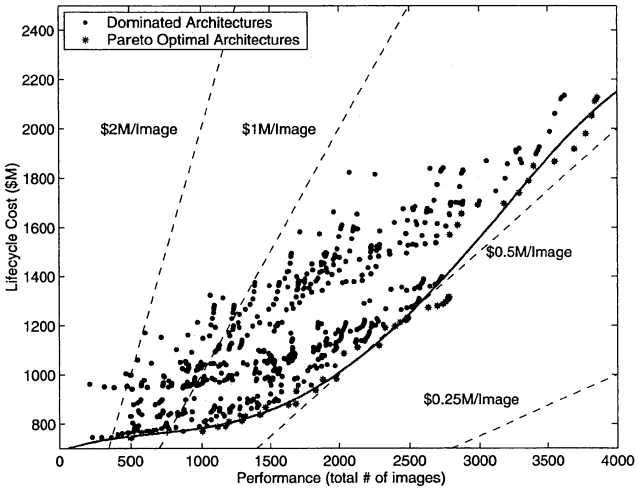


Fig. 8 TPF tradespace global Pareto boundary (optimal front).

enumerating the TPF conceptual design tradespace. Second, using the least-squares fit technique, fit a curve/equation to this P-optimal set of points to create the true global Pareto boundary. Third, compute the P-optimal architectures from the subset of the tradespace searched by an optimization algorithm. Fourth, fit a curve/equation to this P-optimal subset to approximate the estimated Pareto boundary. Fifth, compute the ME between the estimated Pareto boundary and the true global Pareto boundary as a metric of how well the estimated boundary approximates the true global boundary.

Figure 8 shows the global Pareto boundary for the TPF case study. Dots represent dominated architectures within the global tradespace. Asterisks represent nondominated architectures that comprise the P-optimal set. The solid line represents the least-squares fourth-order polynomial fit through the nondominated architectures to create the global Pareto optimal boundary along which one cannot improve system performance without increasing life-cycle cost. This curve is represented by the following fourth-order polynomial equation:

$$\hat{y} = -1.93e^{-11}x^4 + 1.47e^{-7}x^3 - 2.4e^{-4}x^2 + 0.208x + 688 \tag{17}$$

where $\hat{y}$ is life-cycle cost and x is system performance.

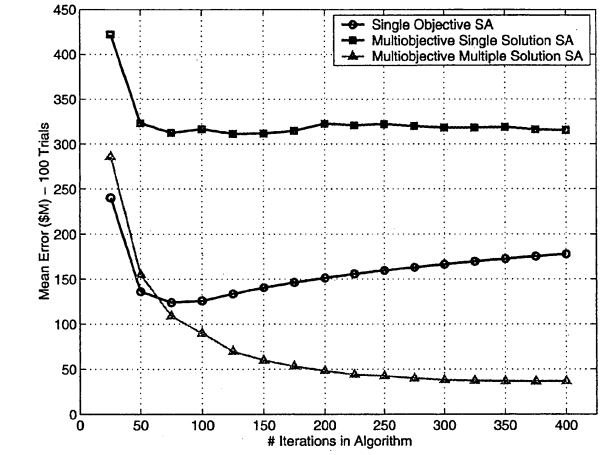
When an optimization algorithm then searches a subset of the TPF tradespace, an estimated Pareto boundary passing through the nondominated solutions in the search subset is computed in a similar manner. To assess how well the local Pareto boundary approximates the global Pareto boundary, the ME may be computed between the two curves:

$$\text{ME} = \sum_{x=i_{\min}}^{i_{\max}} \sqrt{(y_x - \hat{y}_x)^2} \Big/ k \tag{18}$$

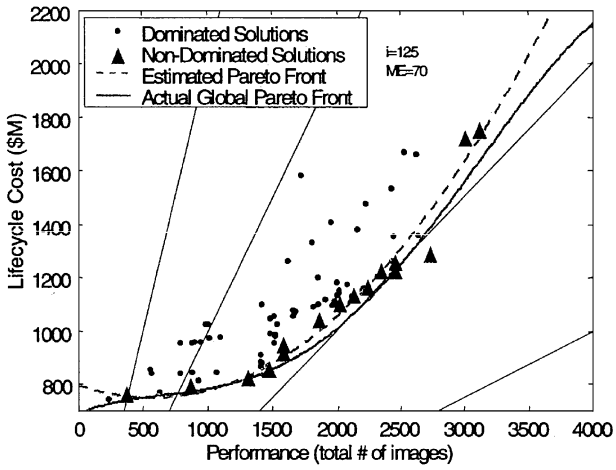
where y_x is the y value of the estimated Pareto curve for a given value of x, $\hat{y}_x$ is the y value of the true global Pareto curve for the same value of x, $i_{\max}$ minus $i_{\min}$ represents the span of the curve for which the approximation is evaluated, and k is the number of points compared between the two curves.

This approach has been applied to approximate the true global Pareto boundary via an estimated Pareto boundary in the TPF case study. The three following simulated annealing algorithms were tested to determine which one best finds the required nondominated solutions to approximate properly the true global Pareto boundary: 1) single-objective (minimize CPI) simulated annealing algorithm [Eqs. (3), (5–7), and (11)]; 2) multi-objective (minimize life-cycle cost and maximize performance) single-solution simulated annealing algorithm [Eqs. (10), (12), and (13)]; and 3) multi-objective (minimize life-cycle cost and maximize performance) multiple-solution simulated annealing algorithm [Eqs. (10) and (14–16)].

These three algorithms were placed in a test matrix in which each algorithm was applied 100 times from 100 different randomly



a)



b)

Fig. 9 a) ME between the estimated and true global Pareto boundary as a function of the number of iterations in each algorithm and b) approximation of the global Pareto boundary by the multi-objective multiple solution simulated annealing algorithm.

selected initial starting points to the TPF conceptual design problem and allowed from 25 to 400 iterations in increments of 25. The results from the test matrix appear in Fig. 9a.

Several interesting observations may be drawn from Fig. 9. First, the multi-objective single-solution simulated annealing algorithm provides the worst global Pareto boundary approximation. This poor performance is because this algorithm searches for only a single architecture in the P-optimal set, making it susceptible to concentrating on only a small portion of the global tradespace. Thus, the estimated Pareto boundary will closely approximate the true global Pareto boundary in this region, but will poorly approximate the true Pareto boundary in the remaining regions of the global tradespace, resulting in the poor ME values.

The single-objective simulated annealing algorithm provides the best approximation of the original three algorithms tested when 50 or fewer iterations are executed but then worsens as the total number of iterations increases. This worsening is an interesting phenomenon that may be attributed to the mechanics of the single-objective simulated annealing algorithm. This algorithm always searches for the same point, that is, the minimum CPI design, and, thus, always concentrates in this region of the tradespace. As the number of iterations increases, the algorithm is allowed to investigate more neighboring design solutions, increasing the bias in this region of the tradespace. Thus, the estimated Pareto boundary closely approximates the global boundary in the minimum CPI region of the tradespace, but poorly approximates the Pareto boundary everywhere else. This is why the single-objective simulated annealing algorithm performs worse as the number of iterations within the algorithm increases, the bias against the nonminimum CPI regions of the tradespace increases.

Beyond 75 iterations, the multi-objective multiple-solution simulated annealing algorithm performs best. This is to be expected because this algorithm was specifically designed to search for the entire set of P-optimal architectures. When this multi-objective heuristic algorithm hits the Pareto boundary, it fans out in both directions along the boundary. For this reason, the multiple-solution simulated annealing algorithm does not develop the same bias for a particular region of the tradespace that the two former algorithms do, leading to better performance in approximating the true global Pareto boundary (Fig. 9b). Based on these results for the TPF case study, the multi-objective multiple-solution simulated annealing algorithm best approximates the global Pareto boundary[23] and should, therefore, be used when searching larger multidimensional DSS conceptual design tradespaces in which complete enumeration is not possible.

All of the multi-objective DSS conceptual design problems considered up to this point contain only two objective functions: two decision criteria and two dimensions. However, the same approach, definitions, and formulas hold for a multi-objective problem with any number of decision criteria, any number of dimensions. Each additional decision criteria adds another dimension to the problem, but the concepts and methods of multi-objective optimization remain the same.

Step 5: Interpret Results (Sensitivity Analysis)

Once the single-objective and multi-objective optimization algorithms have been applied to the DSS conceptual design problem, the results must be analyzed and interpreted to ensure that the output makes sense and that the multidisciplinary GINA model is of sufficient fidelity to capture the relevant trades within the conceptual design problem. This process began in step 2 of the MMDOSA methodology with the application of univariate studies to investigate how a single system architecture attribute varies as a function of a single input parameter. After the optimization algorithms have been applied in step 4 of the MMDOSA methodology, an analysis of variance (ANOVA) may be executed on the examined system architectures to assess the relative impact of the different design decisions the systems engineer has control over on the design decision criteria. In this manner, ANOVA represents a sensitivity analysis tool for the interpretation of the MDO results.

The total sum squares (SS_T) is the sum of the squared differences between each observation y_i, that is, a system metric, and the arithmetic mean $\tilde{T}$ of the entire set of observations:

$$SS_T = \sum_{i=1}^{N} (y_i - \tilde{T})^2 \tag{19}$$

where N is the total number of observations. The SS_γ for each variable in the design vector is

$$SS_\gamma = \sum_{i=1}^{n} n_{\gamma i} (\tilde{y}_{\gamma i} - \tilde{T})^2 \tag{20}$$

where n is the total number of possible settings for design variable γ (example orbit = 1–5 AU, integer), $n_{\gamma i}$ is the number of observations for which design variable γ is at setting i (example orbit = 2 AU), and $\tilde{y}_{\gamma i}$ is the arithmetic mean of all of the observations for which design variable γ is at setting i.

With this information, the percentage of the total variation observed that can be attributed to each design vector variable may now be computed. This relative percentage contribution or relative influence (RI) indicates the relative power of each design variable to reduce variation. The RI may be computed percentagewise for each element of the design vector via

$$RI = [SS_\gamma / SS_T] \times 100\% \tag{21}$$

Figure 10 shows the ANOVA results for the TPF tradespace. As one can see, aperture diameter exerts by far the greatest relative influence

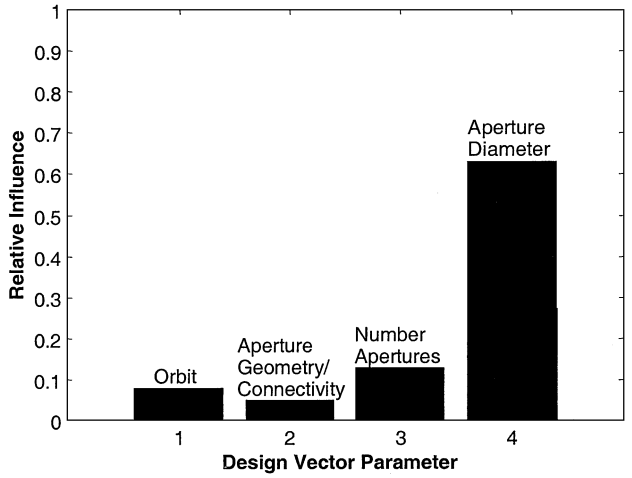


Fig. 10 ANOVA results for the TPF tradespace; relative sensitivity of CPI to design parameters.

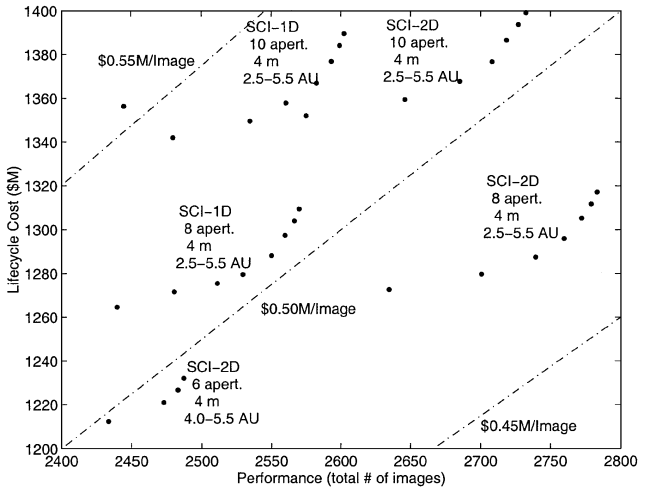


Fig. 11 Minimum CPI architecture families within the most cost-effective region of the TPF tradespace (zoom-in).

on the CPI metric. In comparison, the heliocentric orbital radius, aperture connectivity/geometry, and number of apertures exert less influence on the CPI metric. ANOVA provides the systems engineer with information on which design parameters give the most leverage over various aspects of the architecture.

Step 6: Iterate

Step 6 in the MMDOSA methodology involves an iterative process of examining the results from MMDOSA steps 4 and 5, making the appropriate changes to the GINA model and/or the optimization algorithms and rerunning the optimization algorithm(s). Possible modifications include increasing the fidelity of the critical models[18] as identified by the ANOVA sensitivity analysis, changing the cooling schedule and/or degree-of-freedom parameter[18,23] within the simulated annealing algorithm, warm starting a new optimization run from the final answer of a prior optimization run, and running additional trials of the same optimization algorithm. When an iterative approach to searching the tradespace is taken, the systems engineer gains confidence that the architectures converged on by the optimization algorithms are indeed among the best system architectures within the full DSS tradespace. In this manner, step 6 of the MMDOSA methodology leads to step 7.

Step 7: Converge on Best System Architectures

In the final step of the MMDOSA methodology, system architectures to be carried forward into the next phase of the space systems design and development process are identified based on the iterative optimization and sensitivity analysis results in MMDOSA steps 4–6. In the case of a single-objective DSS conceptual design problem, the recommended architectures are the best family(s) of architectures with respect to the metric of interest found by the single-objective simulated annealing algorithm (Fig. 11). In the case of a multi-objective DSS conceptual design problem, the recommended architectures are those that comprise the P-optimal set with respect to the selected decision criteria as found by the multi-objective multiple-solution simulated annealing algorithm (Fig. 8).

Conclusions

This paper has developed a methodology, MMDOSA, for mathematically modeling DSS conceptual design problems as optimization problems. Of the four classes of optimization techniques, Taguchi methods, heuristics, gradient methods, and univariate methods, tested on the single-objective optimization DSS conceptual design problem, the heuristic simulated annealing algorithm found the best system architectures with the greatest consistency due to the algorithm's ability to escape local optima within a nonconvex tradespace. Accordingly, the simulated annealing algorithm became the core single-objective algorithm within the MMDOSA methodology. Next, the problem scope was greatly increased by expanding from single-objective optimization problems to multi-objective optimization problems. This allowed a more accurate capture of the trades involved in real-world DSS conceptual design problems, and several methods were explored for approximating this global Pareto boundary with only a limited knowledge of the full DSS design tradespace. Both an algorithm that searches for a single P-optimal architecture and an algorithm that searches for multiple P-optimal architectures during a single run were developed.

In conclusion, this work makes three primary contributions to the state of the art in DSS conceptual design. These contributions are as follows: 1) development of a methodology for mathematically formulating and solving DSS conceptual design problems as nonlinear single-objective and multi-objective optimization problems, 2) determination that the heuristic simulated annealing technique finds the best [as defined by the metric(s) of interest] system architectures with greater consistency than Taguchi, gradient, and univariate techniques when searching a nonconvex DSS tradespace, and 3) creation of two new multi-objective variants of the core single-objective simulated annealing algorithm: the multi-objective single-solution simulated annealing algorithm and the multi-objective multiple-solution simulated annealing algorithm. Together, these contributions combine in the MMDOSA methodology to provide system engineers with a powerful, versatile systems engineering and architecting tool for the conceptual design of DSSs.

References

[1]Subramanian, J., Stidham, S., and Lautenbacher, C. J., "Airline Yield Management with Overbooking, Cancellations, and No-Shows," *Transportation Science*, Vol. 33, No. 2, 1999, pp. 147–167.

[2]Mathaisel, D. F. X., "Decision Support for Airline Schedule Planning," *Journal of Combinatorial Optimization*, Vol. 1, No. 3, 1997, pp. 251–275.

[3]Stettner, L., "Risk Sensitive Portfolio Optimization," *Mathematical Methods of Operations Research*, Vol. 50, No. 3, 1999, pp. 463–474.

[4]Mosher, T., "Applicability of Selected Multidisciplinary Design Optimization Methods to Conceptual Spacecraft Design," *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 1996, pp. 664–671.

[5]Riddle, E., "Use of Optimization Methods in Small Satellite Systems Analysis" 12th AIAA/USU Conf. on Small Satellites, Utah State Univ., Logan, UT, Paper SSC98-X-1, Sept. 1998.

[6]Shishko, R., and Chamberlain, R., *NASA Systems Engineering Handbook*, 2nd ed., NASA SP-610S, June 1995, pp. 3–7, 13–26, 67–90.

[7]Bearden, D. A., "A Methodology for Spacecraft Technology Insertion Analysis Balancing Benefit, Cost, and Risk," Ph.D. Dissertation, Dept. of Aerospace Engineering, Univ. of Southern California, Los Angeles, May 1999.

[8]*Systems Engineering Handbook: A "How To" Guide for All Engineers*, 2nd ed., International Council on Systems Engineering, Seattle, WA, 2002, pp. 19–28.

[9]Shaw, G. B., Miller, D. W., and Hastings, D. E., "Generalized Characteristics of Satellite Systems," *Journal of Spacecraft and Rockets*, Vol. 37, No. 6, 2000, pp. 801–811.

[10]Beichman, C. A., Woolf, N. J., and Lindensmith, C. A., *The Terrestrial Planet Finder (TPF): A NASA Origins Program to Search for Habitable Planets*, 1st ed., Jet Propulsion Lab., JPL Publ. 99-3, California Inst. of Technology, Pasadena, CA, May 1999, pp. 1–11, 49–55, 87–89.

[11]Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23.

[12]Mosher, T., "Conceptual Spacecraft Design Using a Genetic Algorithm Trade Selection Process," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 200–208.

[13]Matossian, M. G., "Earth Observing System Constellation Design Through Mixed Integer Programming," *Space Technology*, Vol. 16, No. 5/6, 1996, pp. 233–243.

[14]Miller, D. W., Curtis, A., De Weck, O., Frazzoli, E., Girerd, A., Hacker, T., Jilla, C. D., Kong, E. M., Makins, B., and Pak, S., "Architecting the Search for Terrestrial Planets and Related Origins (Astro)," Society of Photo-Optical Instrumentation Engineers, SPIE Paper 4013-74, March 2000.

[15]Curtis, A., De Weck, O., Frazzoli, E., Girerd, A., Hacker, T., Jilla, C. D., Kong, E. M., Makins, B., and Pak, S., *Architecting the Search for Terrestrial Planets and Related Origins*, MIT Space Systems Lab., Space Engineering Research Center Rept. 6-99, Massachusetts Inst. of Technology, Cambridge, MA, May 1999, pp. 49–141.

[16]Shaw, G. B., Miller, D. W., and Hastings, D. E., "Development of the Quantitative Generalized Information Network Analysis Methodology for Satellite Systems," *Journal of Spacecraft and Rockets*, Vol. 38, No. 2, 2001, pp. 257–269.

[17]Jilla, C. D., Miller, D. W., and Sedwick, R. J., "Application of Multidisciplinary Design Optimization Techniques to Distributed Satellite Systems," *Journal of Spacecraft and Rockets*, Vol. 37, No. 4, 2000, pp. 481–490.

[18]Jilla, C. D., "A Multiobjective, Multidisciplinary Design Optimization Methodology for the Conceptual Design of Distributed Satellite Systems," Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge, MA, May 2002.

[19]Bertsimas, D., and Freund, R. M., *Data, Models, and Decisions: The Fundamentals of Management Science*, 1st ed., South-Western College Publ., Cincinnati, OH, 2000, pp. 148–164.

[20]Brooks, R. R., Iyenger, S. S., and Rai, S., "Comparison of Genetic Algorithms and Simulated Annealing for Cost Minimization in a Multisensor System," *SPIE Journal of Optical Engineering*, Vol. 37, No. 2, 1998, pp. 505–516.

[21]Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 13 May 1983, pp. 671–680.

[22]Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed., Addison–Wesley, Boston, 1989, pp. 197–201.

[23]Jilla, C. D., and Miller, D. W., "Assessing the Performance of a Heuristic Simulated Annealing Algorithm for the Design of Distributed Satellite Systems," *Acta Astronautica*, Vol. 48, No. 5–12, 2001, pp. 529–543.

J. Korte
Guest Editor