NASA-CR-202084

IN-31-CR

81445

System Design under Uncertainty: Evolutionary Optimization of the Gravity Probe-B Spacecraft

Samuel P. Pullen
Bradford W. Parkinson

Department of Aeronautics and Astronautics, Stanford University
Stanford, CA, 94305 U.S.A.

Abstract. This paper discusses the application of evolutionary random-search algorithms (*Simulated Annealing* and *Genetic Algorithms*) to the problem of spacecraft design under performance uncertainty. Traditionally, spacecraft performance uncertainty has been measured by *reliability*. Published algorithms for reliability optimization are seldom used in practice because they oversimplify reality. The algorithm developed here uses random-search optimization to allow us to model the problem more realistically. Monte Carlo simulations are used to evaluate the objective function for each trial design solution. These methods have been applied to the *Gravity Probe-B* (GP-B) spacecraft being developed at Stanford University for launch in 1999. Results of the algorithm developed here for GP-B are shown, and their implications for design optimization by evolutionary algorithms are discussed.

1 Introduction

Design for reliability has always been a critical concern for spacecraft developers because spacecraft, once launched, cannot be repaired after a serious failure without incurring extreme expense. As a result, all spacecraft are analyzed for *reliability*, or the probability of meeting the mission success criteria over time. Because spacecraft reliability calculations must be based on inaccurate failure rate data and questionable assumptions, these numbers are generally used only to show that they meet arbitrary specifications set by the customer.

This paper uses the flexibility of evolutionary optimization methods to overcome these obstacles. Traditional reliability optimization can make only simplified trade-offs between reliability and cost or weight, but global-search methods can handle any optimization function. Furthermore, since simulation can generate arbitrary function evaluations and since gradients are not required, we can adopt a more realistic model of component and subsystem reliability.

Variants of two well-known evolutionary methods are developed here for a general reliability design problem. The *simulated annealing* approach uses one trial solution which "evolves" in the search process, while the *genetic algorithm* maintains a population of solutions that evolve according to the concept of natural selection. Results for the former method for two different objective functions are discussed in detail, while a framework for genetic algorithm evolution is presented along with some preliminary conclusions. The results presented here already show significant improvements over traditional reliability optimization methods and suggest new paradigms for spacecraft reliability analysis.

2 Traditional Reliability Analysis

As mentioned above, reliability analysis is an established field, and it can form the basis for design optimization under uncertainty. Since the U.S. Government is a major customer, the handbooks it has published or influenced contain the generally accepted methods of reliability analysis [1,2,3]. These methods are based on the *exponential* failure distribution in which reliability is given by

$$R(t) = \exp[-\lambda t] \tag{1}$$

where λ is a constant failure rate found in tables [1,2]. This distribution is *memoryless*: the probability of failure over a given interval of time is independent of the length of time that has already passed. This assumption is often questionable, but the exponential distribution continues to be used because of its simplicity.

Most spacecraft contracts use (1) and the data in [1] to compute reliability predictions for the components of their design. Redundancy is usually built in to avoid *single-point failure modes*, which are events that by themselves cause mission failure. Using (1) to compute component reliability, series and parallel network reliability can be computed using the standard equations in [3] which assume that all failure modes are *independent*. The result is a system reliability prediction over the mission time line that must meet user specifications.

Since spacecraft are to a large degree unrepairable after launch, reliability is a key concern, but most systems engineers distrust handbook data and the assumptions present in the traditional model despite being obligated to do the computations. As a result, spacecraft tend to be overdesigned to "ensure" adequate reliability. This guarantees that the reliability specifications are met, but it does not help engineers make informed risk-based trade-off decisions.

3 New Spacecraft Reliability Model

The first step toward improving the traditional reliability model is to use a *Weibull* failure distribution that allows failure rates to vary with time. It is a generalization of the exponential distribution, and its success probability is given by

$$R(t) = \exp[-t^{\beta}/\alpha] \tag{2}$$

Here, α is the *scale* factor that expresses mean time-to-failure, and β is the *shape* factor that varies the effective failure rate over time ($\beta = 1$ gives the exponential distribution). In [4], actual spacecraft failure data is fitted to this failure model, and estimates for α and β for various spacecraft mission types are given. For spacecraft, β is around 0.12 (< 1), which indicates that spacecraft are more likely to fail early in a mission as design or manufacturing flaws become apparent. Failure rates decrease over time, as units that pass through this "burn in" period are more likely to last.

The uncertainty inherent in the handbook failure rates is another concern. Previously, we have created a model that assigns variances to failure rates for various components [5]. Using the data from [4], an algorithm for simulation-based reliability predictions has been developed. For each trial, an exponential failure rate is sampled from a Normal distribution with the mean published failure rate and the assigned variance. It is transformed to the Weibull distribution, and the component reliability is computed using (2) for that trial only. A significant number of samples are thus needed to obtain the resulting uncertainty distributions.

4 Optimization by Simulated Annealing (SA)

Simulated annealing (SA) is one of the simpler evolutionary-type algorithms used for global optimization. SA generates a random variant of the current trial solution and evaluates its objective function value. If the new value is superior, the new solution is accepted in place of the old one. If not, the new solution will still be accepted with a probability given by

$$P_{accept} = \exp\left[-\Delta V/T\right] \tag{3}$$

where ΔV is the difference in objective values and T is an "annealing temperature" that slowly decreases. Higher temperature increases the acceptance probability, so the algorithm is less likely to accept "backward" steps as time goes on [6,7,8].

The SA algorithm used in this study has a unique method of generating a new solution. Trial solutions are specified by a collection of "genes" that give the number of units of each component type to be included. In the spacecraft case, the solution (a string of integers) is broken down into functional subsystems (as shown in Table 3). Each time a new solution is generated, at least one of the changeable subsystems (all but the first two) is randomly selected for modification. Those not selected retain the same values as in the last solution.

For each component in a subsystem to be changed, a pair of staircase functions is computed based on the current number of units (*nc*) and the minimum and maximum number allowed for that component (*m* and *M* respectively). The minimum number is the number needed for mission success, and the maximum for a given case is the *lesser* of two quantities: *2 nc* or the absolute maximum number allowed. The probability function for the number of units in the new solution is

$$\begin{aligned}
P\left[\text{new} = \text{nx} \mid \text{old} = \text{nc}\right] &= 2.5/\{2(M-m)\} && \text{for } nx = nc \\
&= \frac{1-2.5/\{2(M-m)\}}{\sum_{i=1}^{M-nc+1} i}\, nx && \text{for } M \geq nx > nc \\
&= \frac{1-2.5/\{2(M-m)\}}{\sum_{i=1}^{nc-m} i}\, nx && \text{for } nc > nx \geq m
\end{aligned} \tag{4}$$

Equation (4) creates a "stairstep" distribution that peaks when $nx = nc$. Retaining the current number of units is thus quite probable. The more different a new number is, the less likely it is to be selected. Note that there is an equal probability of the new number being either higher or lower than *nc*. This probability function clearly makes large changes unlikely; so new solutions take on an "evolutionary" character.

As noted above, the use of an arbitrary probability model requires *Monte Carlo simulations* to evaluate the objective function for each new design solution. Each simulation step consists of a time simulation of a mission given the system reliability model. Since the unit reliabilities are unknown random variables with the distributions discussed in Section 3.0, these must be re-sampled from a random number generator and the mission reliability recomputed.

Using the canonical SA algorithm in [7], Table 1 gives the parameters used for this research. Note that adaptation as discussed in [6] for continuous objective functions is not used. The convergence tolerance in Table 1 represents a comparison between "evolving" evaluations of a solution that has not been replaced in at least one constant-temperature period (300 new solutions). If this occurs, a new simulation evaluation of the current solution is conducted, and its new evaluation is the weighted average of new and old evaluations. For example, if the solution has not changed in the last 3 temperature iterations (900 new solutions), the new evaluation will be:

$$\textit{new evaluation} = (\,3\,(\textit{old evaluation}) + \textit{new simulation result}\,)\;/\;4 \qquad (5)$$

When the new evaluation differs from the old by less than the convergence tolerance, the algorithm prints the best solution found thus far and exits. The algorithm also exits when the current temperature falls to a point (10 times the tolerance) at which acceptance of any lower-evaluation solution becomes exceedingly unlikely.

SA Parameter	Value	Notes
Initial temp.	2×10^6	$V_{max} = 15 \times 10^6$
Num temp.	300	# iter. for given T
Temp. mult.	0.90	dec. after 300 iter.
No. simulations	500	per function eval.
Converge tol.	0.003	

Table 1: Simulated Annealing Parameters

GA parameter	Value	Notes
Population size	25	duplic. poss.
Crossover rate	0.6	after reprod.
Mutation rate	0.01	use eq. (4)
No. simulations	500	per func. eval.
Converge tol.	0.01	

Table 2: Genetic Algorithm Parameters

5 Optimization by Genetic Algorithms (GA)

In the canonical genetic algorithm (GA) format [9], chromosomes, or members of a population of trial solutions, are expressed as binary numbers (0-1), and the standard genetic evolution operators are designed for this type of population. For the system design application, however, the format used for the SA algorithm in the previous section is much more natural. Thus, variants of the GA operators for this genetic format must be developed. The current versions of these operators are discussed here. Testing of these operators is progressing, and results along with updated operators will be given in a future paper.

Previous research on using Monte Carlo simulation to evaluate the objective function (or *fitness*) of population members has provided insight into the GA design parameters used here [10]. These are given in Table 2. The revised operators are:

Reproduction: Roulette wheel selection is used to choose solutions for the next generation (before crossover). Since the evaluations tend to be similar, the fitnesses are linearly normalized from best to worst by multiplying the difference between the best fitness and a given fitness by 10. The best solution is always reproduced into the next generation (*elitism*), and the weighted-average equation (5) is used when applicable to update (rather than replace) the fitness of the best solution [9].

Crossover: Subject to the crossover rate in Table 2, after reproduction, two solutions are "mated" together to produce *one* offspring. The two parents simply average their unit numbers for each component type within a randomly selected (using the procedure for selecting subsystems in SA) crossover window to give the number of units in the offspring (randomly rounded up or down if *n*.5). The crossover rate determines the ratio of offspring to reproduced strings in the next generation, as N_{cr} (= 0.6 N) solutions are crossed over in N_{cr} combinations to produce N_{cr} offspring.

Mutation: Each gene (number of units for a given component) is subject to random mutation with a probability given in Table 2. If a gene is mutated, the current number of units is replaced according to the SA probability equation (4). This function has a high probability of retaining the same number of units; so the mutation rate is inflated to compensate.

Population Convergence: The convergence test is conducted after fitness evaluation but before the next generation is reproduced. If the average fitness of the population differs from that of the best solution in the population by less than the tolerance given in Table 2, the algorithm stops and outputs the final solution population.

Given these operators, the similarity of SA and GA for this application is clear. The two key differences are "evolving" one solution as opposed to many solutions, and using the SA random-perturbation search with acceptance function (3) as opposed to using the genetic-based operators, which search the objective function by *hyperplanes*.

6 The *Gravity Probe-B* Spacecraft and Objective Functions

While the algorithms detailed above are generally applicable to design optimization problems, the results published here focus on the *Gravity Probe-B* (GP-B) spacecraft being developed by Stanford University under a NASA contract. By orbiting a spacecraft in polar low-earth orbit and using drag-free control to remove disturbances caused by particle impacts, gravity gradients, and the like, it is possible to monitor two relativistic effects on bodies in orbit around a massive object [11]. Since these effects are tiny compared to Newtonian disturbances, extremely precise gyros and readout sensors, a science telescope for precise inertial reference, and an extremely accurate drag-free attitude controller are required [12,13].

The GP-B satellite is divided into two sections. The *experimental payload* is built around the *probe*, which contains the gyros, sensors, proof mass, telescope, gas lines, and electronics, and the *dewar*, which surrounds the probe with superfluid helium to keep its temperature in the cryogenic range needed by the sensors. This equipment has never flown before; so its reliability is uncertain. The *spacecraft bus*, which supports payload operations in space, is being developed separately by Lockheed Missiles and Space Company (LMSC). Figure 1 shows a drawing of the GP-B spacecraft.

Our work uses the separation between payload and bus to focus on spacecraft bus optimization given the uncertain reliability of the launch system and the payload

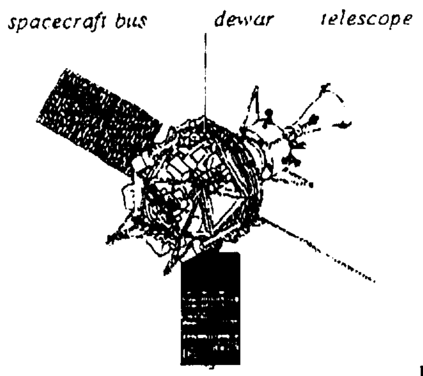

Figure 1: GP-B Spacecraft

(which LMSC cannot control). It is thought that the bus should be made very simply and reliably to not add unnecessary failure modes to an already high-risk situation. This logic will be tested by the optimization carried out here.

The optimal design is driven by the form of the objective function used to model the *utility*, or relative values of outcomes, of the decision maker. The objective function for LMSC is assumed to be the fee, or profit as a percentage of the spacecraft bus cost, it is to receive depending on the outcome of the mission. *Caveat*: the following objective functions are based on generalizations and simplifications of the NASA contracts for GP-B: they do not explicitly give the true LMSC fee agreement.

To represent constraints, *penalty functions* are applied which subtract costs that for exceeding spacecraft weight, volume, and power constraints:

$$\textit{LMSC value} = \textit{award fee} + \textit{cost fee} + \textit{on-orbit fee} - \textit{penalty costs} \quad (6)$$

The Stanford objective function is instead focused only on achieving a successful science mission, so it is dominated by on-orbit performance:

$$\textit{Stanford value} = \textit{on-orbit value} + \textit{cost savings} - \textit{penalty costs} \quad (7)$$

For LMSC, the on-orbit fee percentage of the baseline bus cost of \$ 100 million is given by the following equation, assuming a spacecraft bus failure ends the mission. If there is no failure, LMSC gets the maximum of 6%. If a launch or payload failure ends the mission, the LMSC fee percentage is zero unless at least six months of success are obtained, in which case this equation for the maximum fee (MPF) is used:

$$\mathrm{MPF} = \left[\frac{(\mathrm{tm}-6)^5}{216\,|\mathrm{tm}-6|} + 6\right]\mathrm{PFF} - 6 \quad (8)$$

where tm is the number of months of successful science data-taking, and PFF is an independent, subjective performance evaluation made by NASA. This equation is also used for the Stanford on-orbit fee except that the result is normalized to one by dividing by 6% (the best possible result), since it is the only driving factor.

For LMSC, NASA regulations set the minimum overall fee to 0% and the maximum to 15% (even though (6) can give a wider range of values). Note that changing the redundancy of the spacecraft bus design primarily changes the on-orbit value, unit costs, and penalty only. Since much of the LMSC award fee is independent (for our purposes) of the bus design, it seems that greater improvement can be expected for the Stanford function.

Also note that these functions do not employ *risk aversion* to express nonlinear preferences for very good or very bad results. The functions (6) and (7) are based on *expected values* only. However, Stanford places a much greater penalty on a mission that produces no useful science data. LMSC must also worry about the consequences of a spacecraft failure to its reputation. Future optimization runs will experiment with how exponentially scaled risk-averse values affect the optimal solutions.

7 Design Optimization Results

Optimal design runs using both the LMSC and Stanford objective functions and the SA algorithm have been conducted. Attempts have also been made to determine which input parameter changes show the most sensitivity in the optimal results.

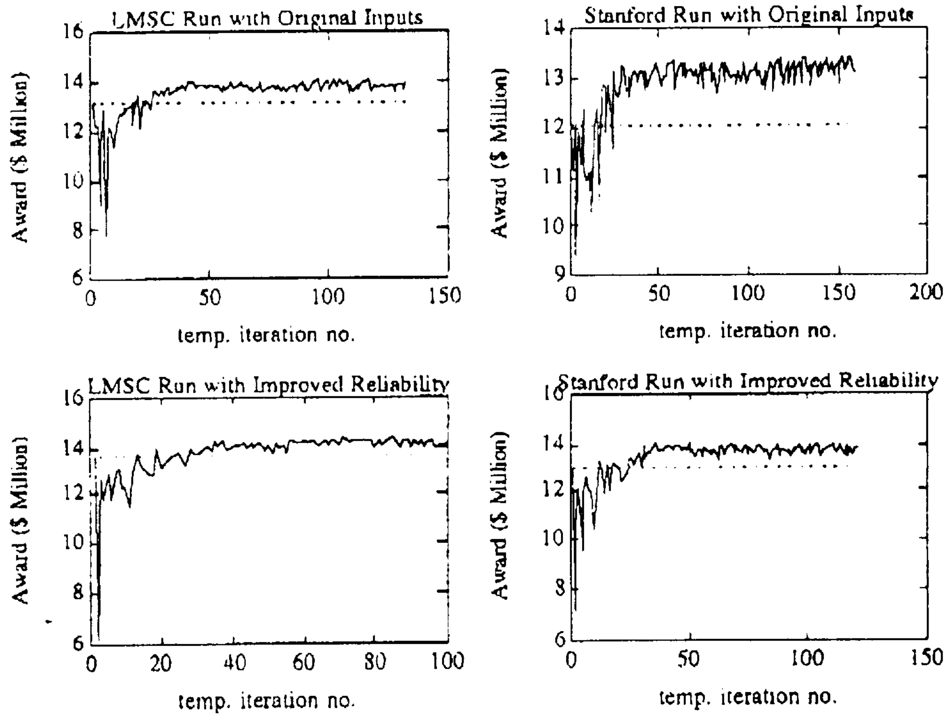

Figure 2: Results of Optimization Runs

Figure 2 shows the evolution of the LMSC and Stanford objective values for two typical runs. The dashed line represents the initial evaluation of the LMSC baseline design. Each evaluation point represents the value of the current SA solution at the end of each constant-temperature iteration. The last points on these plots represent the latest evaluations of the best solutions found. The upper plots are for runs with the "best guess" reliability models for launch and payload operations, both of which (mean reliability of 0.93) are considerably worse than for the baseline spacecraft bus design (0.975). The lower plots are for runs in which the launch and payload models were improved and uncertainty was reduced. In these latter cases, we expect the resulting value uncertainty to decrease, although there is less room to improve over the baseline result. Convergence appears rapid, but the time scale is misleading, since the termination tolerance is deliberately made small to run more reliability simulations and study the steady-state variance of the optimal design values.

Table 3 contains the minimum numbers of working components, the LMSC baseline design, and the best solutions for the "original reliability" case (L = LMSC, S = Stanford). The algorithm was initialized with the baseline solution, so it is to be expected that the objective value would drop as the SA algorithm explores a range of options. Table 4 has mean result values that show 5.1% improvement over the basic baseline design for the LMSC case and 9.2 % improvement for the Stanford case. The greater improvement for the Stanford cases reflects the greater optimization focus on on-orbit performance in that case. Running on a Sparc-10 workstation, "official" convergence was obtained in about 10-18 hours, but as noted above, the tolerance could have been increased (stopping the runs earlier) with little change in the results.

Component	Min.	L base	L opt.	S opt.	Component	Min.	L base	L opt.	S opt.
Structure	1	1 (nc)	1	1	telem. proc.	1	2	3	4
Thermal	1	1 (nc)	1	1	remote proc.	1	2	4	2
SA string	92	96(nc)	96	96	flight comp.	1	2	2	3
power reg.	1	2	1	3	solid-st. rec.	1	2	1	1
NiCd batt.	1	2	1	1	P/Y gyro as.	1	2	1	1
pwr. cont.	1	1 (ir)	2	2	star tracker	1	2	3	2
omni ant.	1	4	1	3	R/Y gyro as.	1	2	1	1
circ. switch	1	2	2	1	thrusters	16	18	24	22
RF switch	1	2	1	4	ATC elect.	1	2	2	2
trans. sw.	1	2	1	1	mass trim	1	2	2	1
trnspond.	1	2	4	3	SA release	4	4 (ir)	8	7
cmd. proc.	1	2	3	4	SA separ.	1	1 (ir)	2	1

(nc) cannot be changed (ir) internally redundant

Table 3: GP-B Spacecraft Optimal Design Results

Note in Table 3 that the optimal designs show some consistent patterns when compared to the baseline solution. While the baseline has the same redundancy for all components, the SA-generated solutions remove redundancy from less-risky or cost-effective components and add redundancy to components that have high failure rates and/or failure rate uncertainties. This is not surprising, and it points out the potential sub-optimality of the traditional method of allocating redundancy. The use of penalty functions works well, as no "invalid" solutions were accepted after the first 30 iterations. For certain components, the optimal redundancy changes significantly from one run of the SA algorithm to the next, and there are places where a human designer could adjust the results using his own qualitative design knowledge. Interestingly, attempts by the authors to do this have not yet been able to achieve a higher objective value. This suggests that the objective function is *insensitive* to components whose optimal unit numbers vary noticeably between optimization runs.

However, the Monte Carlo function evaluation uncertainty needs to be addressed. Using the Central Limit Theorem (CLT), the simulated function evaluations can be considered to be approximately Normally distributed. From Figure 2, we see that convergence around a limited range of result values is typically obtained after about 30 iterations. Table 4 contains the mean values (μ) and standard deviations (σ) of the objectives for all four cases. This level of variability does not seem to confuse the SA algorithm since convergence around the mean result is not lost after it is acquired.

Case	μ ($ M)	σ ($ M)	% Imp.	Case	μ ($ M)	σ ($ M)	% Imp.
LMSC orig.	13.81	0.143	5.14	Stan. orig.	13.12	0.185	9.23
LMSC imp.	14.13	0.141	3.62	Stan. imp.	13.75	0.172	6.01

Table 4: Summary of Optimization Results

One approach to handling the variability question would be to monitor the statistics in Table 4 in "real time" and to stop the Monte Carlo evaluations when sufficient certainty is attained. This idea is developed for GA's in [14]. The measure of statistical certainty could be based on a measure of risk aversion to the variability of the award result. This approach will be examined in follow-on research.

Since the SA procedure gives good results and is flexible enough to adapt to a wide variety of design problems, our work on the separate genetic algorithm approach is attempting to compare and contrast the two methods as adapted to this reliability design problem. While the SA approach chooses subsystem groups in which to make random modifications, the GA crossover operator, since it searches by hyperplanes at the component level, may be better able to isolate the individual components that are critical to overall performance. Better search-by-component might avoid the differences between unit numbers seen in different runs of the SA procedure. However, the GA approach will consume more computer time per solution evaluation and may not give significantly better results. Even so, our ability to modify the canonical SA and GA approaches to fit a given design problem or apply domain-specific knowledge suggests that various mixes of operators from both algorithms may give the best results.

8 Summary and Conclusions

The results shown here seem to justify the use of global optimization for this spacecraft design problem. Using simulated annealing, improvement can be obtained relative to the LMSC baseline design even though the objective functions are only partially sensitive to changes in component redundancy. Convergence of the global search did not take very long considering the complexity of the reliability analysis and the variance of the Monte Carlo simulations. The smooth pattern of convergence demonstrates that evolutionary global search is a useful way of conducting reliability optimization based on the new reliability model. Although the variance of the objective values is significant, it does not prevent the algorithm from converging.

The usefulness of this type of design optimization algorithm is threefold. First, it allows search of the design space to globally optimize an *arbitrary* value function based on an arbitrary system performance model. Second, it demonstrates the flexibility of evolutionary global optimization using simulation evaluations, since it accommodates domain-specific modifications to canonical SA and GA that improve search efficiency. Lastly, it is a design tool that allows the user to complement the computerized search by making manual variations to the optimal result in an attempt to gain further improvement.

The genetic algorithm-based optimization procedure is now beginning testing using the same reliability model. Other areas of follow-on work include simulation variance monitoring and reduction methods and applying risk aversion to the objective function evaluation. This application of evolutionary search strategies to optimal design under uncertainty should be very useful for a wide range of real-world projects.

References

1. *Reliability Prediction of Electronic Equipment* (MIL-HDBK-217E). Washington, D.C.: Department of Defense, October 1986
2. *Nonelectric Parts Reliability Data 1991* (NPRD-91). Rome, N.Y.: Reliability Analysis Center, 1991
3. *Reliability Modeling and Prediction* (MIL-STD-756B). Washington, D.C.: Department of Defense, November 1981
4. H. Hecht, M. Hecht: "Reliability Prediction for Spacecraft" (AD/A164-747). Los Angeles, CA.: SoHar Inc., December 1985
5. S. Pullen, B. Parkinson: "A New Approach to Spacecraft Reliability Analysis Using Tabulated Failure Rates". Stanford Univ., Unpub. Manuscript, June 1993
6. D. Vanderbilt, S.G. Louie: "A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables". *Journal of Computational Physics*, 56, 259-271 (1984)
7. A. Karimi, A.V. Sebald, S. Isaka: "Use of Simulated Annealing in Design of Very High Dimensioned Minimax Adaptive Controllers". In: *Asilomar Conference on Signals, Systems, and Computers*. Pacific Grove, CA. 1989, pp. 116-118
8. W. Press, S. Teukolsky, W. Vetterling, B. Flannery: *Numerical Recipes in C, Second Edition*. New York: Cambridge University Press 1992
9. L. Davis (ed.): *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold 1991
10. J. Grefenstette, J.M. Fitzpatrick: "Genetic Search with Approximate Function Evaluations". In: *Proceedings of the First International Conference on Genetic Algorithms*. Pittsburgh, PA. 1985, pp. 112-120
11. C.W.F. Everitt: "Gravity Probe B: I. The Scientific Implications". In: H. Sato, T. Nakamura (eds.): *Proceedings of the Sixth Marcel Grossmann Meeting on General Relativity*. Singapore: World Scientific 1992, pp. 1632-1644
12. D. Bardas, *et. al.*: "Gravity Probe B: II. Hardware Development: Progress Towards the Flight Instrument". In: *Ibid.*, pp. 382-393
13. Y.M. Xiao, *et. al.*: "Gravity Probe B: III. The Precision Gyroscope". In: *Ibid.*, pp. 394-398
14. A.N. Aizawa, B.W. Wah: "Dynamic Control of Genetic Algorithms in a Noisy Environment". In: *Proceedings of the Fifth International Conference on Genetic Algorithms*. Urbana-Champaign, IL. 1993, pp. 48-55