Applied Mathematics and Computation 218 (2012) 4710–4730

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Aerospace design optimization using a steady state real-coded genetic algorithm

John D. Dyer [a,1], Roy J. Hartfield [a,*,2], Gerry V. Dozier [b,3], John E. Burkhalter [a,4]

[a] Auburn University, Auburn, AL 36849, United States
[b] North Carolina A&T, Greensboro, NC 27411, United States

ARTICLE INFO

Keywords:
Aerospace
Optimization
Rocket propulsion
Genetic algorithm
Real coded genetic algorithm
Steady state genetic algorithm

ABSTRACT

This study demonstrates the advantages of using a real coded genetic algorithm (GA) for aerospace engineering design applications. The GA developed for this study runs steady state, meaning that after every function evaluation the worst performer is determined and that worst performer is then thrown out and replaced by a new member that has been evaluated. The new member is produced by mating two successful parents through a crossover routine, and then mutating that new member. For this study three different preliminary design studies were conducted using both a binary and a real coded GA including a single stage solid propellant missile systems design, a two stage solid propellant missile systems design and a single stage liquid propellant missile systems design.

1. Introduction

Optimization of aerospace engineering applications using genetic algorithms (GA's) such as spacecraft controls [1,2], turbines [3], helicopter controls [4], flight trajectories [5], wings and airfoils [6,7], missiles [8–11], rockets [12], propellers [13] and inlets [14] have been preformed with great success. In some cases, real coded GA's have been shown to produce better results than binary coded GA's [15–17]. This success is the primary motivation for the current study involving aerospace applications. The goal of engineering design optimization is to find an optimum solution to a design problem. Optimization has evolved throughout the years from classical methods to modern evolutionary algorithms. Modern computers with their exceedingly fast computational times have enabled optimizers to become extremely efficient at solving very complex problems.

All GA's are based on the principles developed by John Holland in his book "adaptation in natural and artificial systems" [18]. Holland outlined the methods for successfully implementing population based adaptive optimizers. Holland's methods operate on the principle of survival of the fittest. In a computational sense, candidate solutions are assembled in a population and compared to one another, the weak die off and the strong are left to reproduce and mutate to produce better children.

The search for a more efficient optimizer for some of these aerospace applications arose due to the extended run times associated with long range missiles using the IMPROVE© optimizer (Implicit Multi-objective PaRameter Optimization Via Evolution) [19]. The IMPROVE© binary encoded generational GA has been used extensively for optimization of missile systems and has been shown to be a very versatile and robust method for optimization. A primary disadvantage of the binary

* Corresponding author.
E-mail address: rjh@eng.auburn.edu (R.J. Hartfield).
[1] Graduate Research Assistant, Aerospace Engineering, AIAA Member.
[2] Professor, Aerospace Engineering, Associate Fellow AIAA.
[3] Department Chair, Computer Science.
[4] Professor Emeritus, Aerospace Engineering, Associate Fellow AIAA.

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.07.038

Nomenclature

μ	mutation rate
σ	mutation amount
Ae	nozzle exit area
A^*	nozzle throat area
b2tail	tail semi-span
b2wing	wing semi-span
crtail	tail root chord
crwing	wing root chord
dbody	diameter of body
dnose	nose diameter
dstar	throat diameter
eps	epsilon-grain
f	propellant grain fillet radius
GA	genetic algorithm
lbody	length of body
LE	leading edge
lnose	length of nose
N	number of star points-grain
rbody	radius of body
Ri	propellant inner grain radius
rnose	radius of nose
Rp	propellant outer grain radius
TE	trailing edge
TOF	time of flight
xLEt	X-location of tail leading edge
xLEw	X-location of wing leading edge

coded GA comes from the fact that because all of the variables must be converted into a single bit string, the solution accuracy is dependant on the number of bits that can be used for the string.

Because of the large ranges associated with many of the design parameters the smallest resolution for the binary GA is generally limited to about 1% of the solution space for complex problems while the real coded GA is only limited to a double precision number. The two stage solid missile system model used in this study has 46 design parameters making up each member of the population, thereby compounding the resolution problems for the binary GA. Resolution is not a significant issue with a real coded GA because all of the variables remain real double precision variables. Dozier et al. [20], Unsal et al. [21] and Dozier et al. [22] demonstrated the ability for a real coded GA to achieve shorter run times as well as more accurate solutions for some applications. Hamming cliffs can also pose a problem for the binary GA because all of the design parameters are converted into a single bit string. For example if two integers 15, and 16 were represented by the bit strings 01111 and 10000 respectively , the GA would have to change all of the bits simultaneously to change from 15 to 16. Mutation and crossover do not always solve this problem. Hamming cliffs are not possible with the real GA because the design variables remain real coded.

Another advantage of the real GA created for this study is that it uses steady state optimization unlike the generational optimization used by the binary GA. The key difference is that in the steady state GA for each generation only the worst performer is thrown out and replaced by a new member, whereas for the generational GA all of the members of the population are thrown out and replaced (expect in elitist mode when the best member remains in the next generation) using a similar tournament routine. For complex problems the steady state GA may not be as efficient as the generational GA because of its lack of diversity. For the steady state GA, once the survivors have had a chance to crossover (i.e. pass genetic material back and forth through their variables), the new member replacing the worst performer is run back through the objective function. This process continues, with the parents producing on average better offspring, until the maximum number of generations (user specified) is reached. There are proofs [23,24] which show why this process produces increasingly superior performers in a population, but a simplistic view is that a good parent mated with another good parent, is more likely to produce good offspring than two poor parents when mated. This is not to say that two good parents cannot produce poor performers. Rather, when two good performers exchange genes, statistically the resulting offspring have a higher chance of outperforming their parents.

2. Real-coded GA methodology

The real and binary GA's used in this study both operated using the same tournament style evolution of a population. They each work with a number of candidate solutions to solve a particular problem. A data structure known as an individual