Intelligent Generation of Candidate Sets For Genetic Algorithms in Very Large Search Spaces

Julia R. Dunphy Ph.D.
Jose J. Salcedo M.S.
Keri S. Murphy

Julia.Dunphy@jpl.nasa.gov
Jose.Salcedo@jpl.nasa.gov
Keri.Murphy@jpl.nasa.gov

Jet Propulsion Laboratory, California Institute Of Technology, Pasadena, California USA

1.0 Introduction

For several years we, at JPL, have been working on how to select safety measures for space missions in an optimal way. The main limitation on the measures that can be performed is, of course, cost. There are often hundreds of possible measures which may be taken, ranging from simple inspections to using complex environmental simulators. Each measure has an associated cost and an effectiveness that describes its potential to reduce the risk to the mission goals. Making the trades between measures is still mostly manual and often human biases show up in the final selected. The DDP (Defect Detection and Prevention) project is an attempt to take the human bias out of the process and make all selections of safety measures as optimal as possible. However, using a computer search of such an enormous search space is not practical if every combination is evaluated. It was therefore decided to use an evolutionary algorithm to improve the efficiency of the search. A naïve approach might be to take random combinations of measures as candidates, evaluate the utility of each combination, and then use genetic algorithm methods based on high efficiency and low cost. However, this would lead to many sets of solutions which were wildly expensive and so unfeasible. In fact, this method is quite likely to yield a best 100 solution set without any members that could come in within the assigned budget. So we decided to try and generate only candidates that were affordable from the outset.

2.0 The DDP Model

A PACT (Prevention, Analysis, Control or Test) is a way of reducing the likelihood of one or more Failure Modes (FMs) occurring. Each requirement is given an arbitrary

weight of importance to the project and an FM, if it occurs, reduces the total recovered requirement weight (i.e. the percentage of the requirement weight that is likely to survive the FMs)

Problem statement

How do we choose the PACTs that are the most cost effective? There are typically hundreds of PACTs that might be employed with complex internal and external interactions. So since a typical solution to the problem may involve 20 to 30 PACTs, the number of combinations is enormous. Even after the constraint that the cost goal be met, the number of possible combinations that do not exceed budget allocation is still large but tractable.

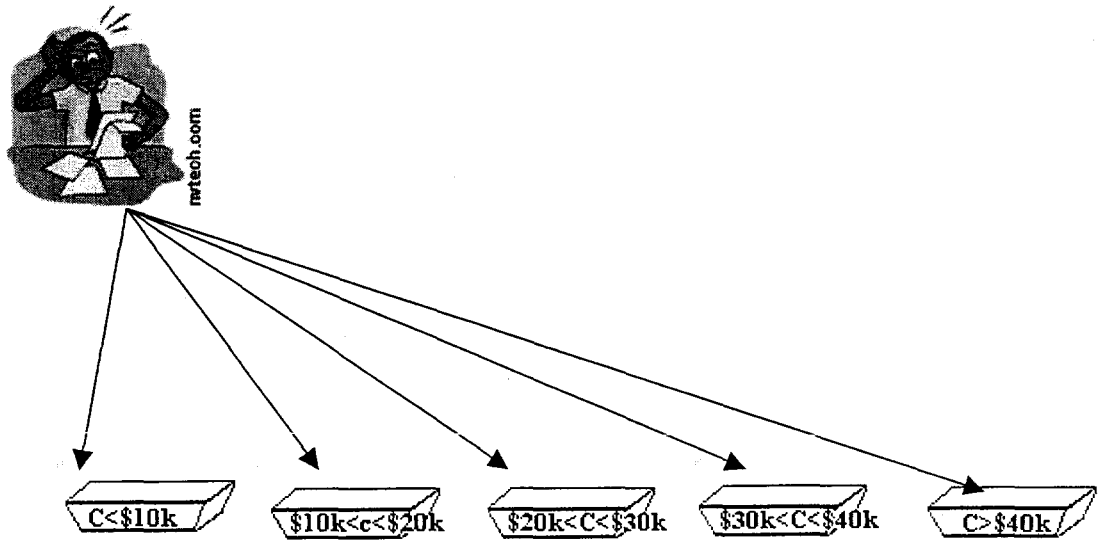
3.0 Sifting the combinations

Naïve searches of all possible candidate sets for those that meet the cost constraint is not practical. We are now actively investigating evolutionary techniques such as genetic algorithms to rank the solutions. The subject of this paper, however, is to describe the methodology for creating the candidate sets. We plan to publish our results concerning the efficiency of the algorithm later.

Binning strategy

To make the problem tractable, each PACT is assigned to a cost bin that can roughly be thought of as "Low Cost ,Inexpensive, Median, Expensive, and Very Expensive". The division into five bins is quite arbitrary and later we may try a larger number of bins.

The lowest and highest cost extracted from all the PACTs are used to determine the cost limits for each bin. After the bins limits are calculated, the individual PACTs are each assigned to a bin based on their cost.



	b1	b2	b3	b4	b5
Ave($k)	5.0	15	25	35	45
Max/Av	10	4	2	2	1

Example keys for a $360k budget (using average values for each bin):

{0,0,0,8,2} 8 X 35 + 2 X 45 = 370
{20,8,4,1,0} 20 X 5 + 8 X 15 + 4 X 25 + 45 = 365

Candidate solutions for the evolutionary algorithm are first identified by a key such as {4,1,6,3,2} indicating how many of each kind may be chosen. This key does not define the actual candidate solution but only the number of PACTs that can be extracted from each bin in turn to achieve the cost limit. We use integer arithmetic so that the actual cost of the candidate is only roughly equal to the maximum cost. The round-off errors can produce quite significant deviations from the maximum cost if they occur in the high value PACTs. The methodology ensures that we won't discard a solution that exceeds our coast maximum by some small amount. It is often possible to get a budget increase if significant increase in safety is achieved by a relatively small cost increase. The actual candidate then is extracted by another tuple that includes the actual indices within each bin such as {3,0,5,2,0}. Each value in the second tuple must, of course, be less than the value in the key.

Because of the fact that the costs are unrelated, it is quite possible that some bins have no members. If this is the case, we eliminate the bin but keep the cost limits in the other bins the same. It is also possible that the number of PACT combinations that meet the cost limit is very small. This may be due to either the cost limit being too high or too low. If the cost limit is so high that it can only be achieved by a very few candidates we automatically lower the cost limit. If the cost limit is too low, we declare that the evolutionary search is not possible. If the cost limit is low enough to leave about 1000 to 10000 possible candidates we replace the evolutionary search algorithm with a simple exhaustive search using the sequential mode defined below.

Each actual candidate solution is extracted by means of an iterator as described in the next section.

Iterator

New candidate solutions are created by the iterator. The iterator is created each time a new key is employed. The iterator can work in a random or sequential mode. In sequential mode the iterator will cycle through all the lowest cost PACTs (those in bin 1) and when all combinations that meet the key signature have been exhausted, it will change the selection from bin 2 by one member and then cycle bin 1 again. This is similar to the way normal counting works (e.g. units exceeding nine create a carry into the tens). This mechanism ensures that all possible candidates will have been generated eventually and each candidate will have roughly (but not exactly) the same cost. A complete candidate PACT set is then composed of a set of subcandidates, one each from each bin whose count in the tuple is not zero. In random mode, however, the iterator will generate subcandidates randomly in each bin. The iterator is programmed to make sure that each candidate is different by keeping track of those previously created.

A sequential iterator can completely encompass every combination within the bin if called a sufficient number of times.

Genetic Algorithm

The candidates are assembled into a population. Because of the selection method they are guaranteed to have roughly the same cost. They are rated by their total Recovered Requirement ability (as a percentage of total requirements in failure free environment)

Mutations

Generation to generation transformation is based on the work of Colin Williams at JPL on Quantum Computing Circuits. Normal Genetic Algorithm mutations, crossovers etc are employed with some new ideas. We use the tuple idea one more time to improve the efficiency of the algorithm. In our application it is possible to substitute an entire subcandidate at mutation time if desired. We therefore carry both normal (single PACT) substitution and subcandidate substitution. In either case, we are ensured that the cost boundary will be loosely obeyed.

4.0 Testing

We are now testing but have few if any results to report yet.