16th International Mirce Symposium
December 6-8, 2006, Woodbury Park, Exeter

OPTIMISATION OF SYSTEM RESOURCES IN RELIABILITY AVAILABILITY & MAINTAINABILITY PROBLEMS USING GENETIC ALGORITHMS

Aviv GRUB, Andy J. KEANE, University of Southampton

We propose an application of Genetic Algorithms to the output from Reliability Availability & Maintainability models of complex (mainly industrial) systems to reduce computational costs while maintaining a desired level of fidelity of the model. The paper notes the fundamental difficulty of applying resource optimisation to models of complex systems given the dimensional and computational costs involved in realising it. To date, several analytic approaches have been suggested to provide solutions, which are very fast, but fundamental question marks have arisen regarding their fidelity. By their nature they are limited in the complexity of problems they can tackle. A hybrid approach, which utilises Monte Carlo Simulations together with an analytic metric for the search process has also been proposed. This hybrid approach preserves the model's accuracy and fidelity, but as it involves a study of the model's physics is limited to spare parts and repair teams as the resources, and to the availability as the objective function in the optimisation. The approach proposed here also uses Monte Carlo Simulations, but the difference is that the Genetic Algorithms approach does not require any prior study of the model's underlying structure, instead referring to the Monte Carlo Simulations as a black box engine, which yields an accurate set of outputs for a given set of inputs. Hence, the Genetic Algorithms approach can be applied for the optimisation of a wider resources space and for any desired objective function defined and calculated in the model. At the same time, the application of the Genetic Algorithms is found to be an efficient and relatively cheap search method for optimisation.

1 - Introduction

In this work we are trying to reduce the computational costs of resources[1] optimisation in Reliability, Availability & Maintainability (RAM) models of complex systems by applying Genetic Algorithms (GA) for the search process. Optimisation processes in these types of models are often prohibitively expensive since the resources space has many degrees of freedom, consisting of a large number of solution points, where each solution requires a complete Monte Carlo Simulations (MCS), which are often slow making the entire process expensive. MCS must be applied in such problems since generally the form of the equations which govern the solution is a multidimensional integral transport equation (the System Transport Equation) or a set of simultaneous integral equations. The System Transport Equation stems from the Boltzmann Transport Equation for neutral particles. A full discussion can be found in Dubi, 2000. One way of bypassing this obstacle is to use a hybrid process which uses fewer MCS runs to provide information to an analytic algorithm using bulk parameters to search within the resources topology with reliable references. In

[1] Every logistic element, which can support the system availability (e.g. spare parts, depots, repair teams, tools, vehicles etc.) may be regarded as resource. Resources may also be referred to as supporting other target functions rather than Availability, such as power generation, profit etc.; in such cases, Resources can also be 'good weather', papers (on which a 'good contract' is put together) and so on.

that manner, the analytic algorithm learns from the MCS and the analytic predictions are verified and corrected.

Although the Hybrid optimisation is very effective and accurate, it requires a full understanding and study of the model to set it up. Moreover, it is limited to spare parts and repair teams as the resources and so far has been tailored only to availability as a response function. We wish to expand the optimisation to any controlled parameter of the problem's input, such as preventive maintenance (PM) plan or alternative management policies, different logistic structures etc., and to apply the search process to other functions, such as to Whole Life Cost (WLC), profit, safety related issues and so on.

We start with a definition to our optimisation problem. We then introduce the concept of Sufficiency, which will be utilised also for the GA at a later stage. After that and, before we introduce the Hybrid method we introduce and describe the studied model scenario. As a starting point for the use of the GA we take a series of random samples from the Resources space, which we follow by application of the GA, enabling a more efficient search process.

2 - Optimisation Definition and Difficulty

The computational time needed to simulate a realistic RAM problem for which all the input parameters are given is normally between minutes and days. Some of the input parameters in such problems cannot be readily controlled to yield desirable results. For example, it is not possible to reduce the lightning rate which may cause delays on vessels on their way to support a failed wind turbine in an offshore wind farm. Some other input parameters can be readily changed; for example, the number of spare parts of each replaceable type of component, the number of repair teams, the number and frequency of preventive maintenance tasks and any other controllable parameters representing supportive Resources for the system performance. In such circumstances we try to find the set of input values which will maximise performance. For example, for ageing systems, preventive maintenance may be vital once in a while; but how much? "Not at all", will result in a strongly increasing failure rate, but on the other extreme, too much will cause the system to be always under maintenance and hence unavailable and therefore not performing. So, there will be an optimal maintenance plan for which the performance will maximise. Another example would be spare parts allocation. This is slightly different since the performance is a monotonic function of the spare parts allocation. The more spares the better the performance; so the optimum would be the maximum. But, spare parts cost money, where the latter can be regarded as a constraint on the former. Then, we may define spare parts optimisation as one of the following:

1. For a given budget, provide the best performance and find the spare parts allocation which will yield it.
2. For a minimal required performance, find the cheapest spares mixture which will maintain it.

   For a minimal required performance, find the cheapest spares mixture which will maintain it.
3. Minimise the entire cost (or maximise profit) and find the corresponding spare parts mixture with the maintenance plan to provide it.

The Resources space also has a great many degrees of freedom. In order to build up the topology of the performance as function of all the Resources, one must execute an unrealistic number of MCS, and each might last minutes, hours or days. Therefore, running a search algorithm testing each and every possible combination of Resources, and for each, obtaining the objective function, in order to find the optimum point, would be unrealistic. So, we must compromise. In such circumstances a hybrid approach can be used. Here the hybrid approach uses a fast analytic search algorithm which learns from MCS. This will be introduced shortly, but it requires acquaintance with the idea of Sufficiency, which will be used in the GA process as well, and is therefore defined next.

3 - Sufficiency

Let q_i^{∞} denote the number of spare parts of type i that should be allocated in order to achieve full confidence for never having lack of a spare. Given the stochastic nature of spare parts consumption, q_i^{∞} approaches infinity for any time interval and certainly for the whole service time of the system. "Full confidence" however means with 100% probability. This probability is known as 100% Sufficiency. One can ask though for a smaller Sufficiency for which q_i^{∞} would become a finite number. For example, 98.5% sufficiency for 7 spare parts of type 'Gearbox', means that for 'Gearbox', given that 7 spares will be stored, with probability of 98.5% there will be never lack of a spare (upon all demands from the storage). Due to the stochastic nature of the variables in the problem, only when the number of allocated spare parts approaches infinity, does the sufficiency approach 100%. Hence, going from infinity to 7 spare units, would mean giving up 1.5% sufficiency. Therefore, in that case, allocating 7 spares or 7,000 spares is expected to have a negligible difference in the effect on the spare parts availability and even smaller effect on the performance.

Normally, when attempting to allocate more spare parts than the number which provides 95% sufficiency, the performance will become degenerated due to saturation concerning spare parts. In most problems, 98.5% sufficiency can be regarded as the upper bound allocation for spare parts.

It is important to note that this level of sufficiency does not provide an optimal solution to the allocation, but rather an extremely expensive one. Again, this would be the highest spares allocation, obtaining the best performance with regard to spares support, from which the spare parts provisioning is to be reduced towards the optimum.

4 - The RAM System Model

The model studied here is complex with respect to any analytical calculation but fairly simple when modelling using a simulation modelling platform. The simulation modelling tool used to build and calculate this model is SPAR™ (courtesy of Clockwork Solutions Ltd.).

In the problem studied here there are twenty three systems deployed in three different fields. All twenty three systems have identical structure and for each of which the operational condition is represented by the Reliability Block Diagram (RBD) shown in figure 1, where the generic name for a component is a Line Replaceable Unit (LRU)

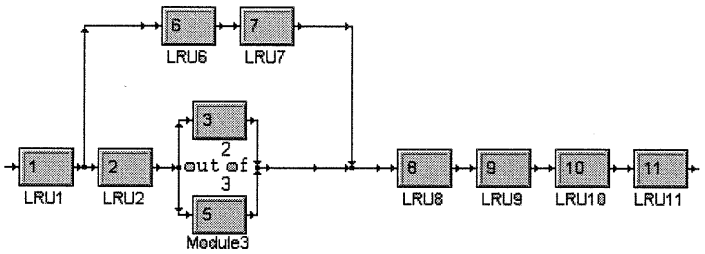

Figure 1 - Reliability Block Diagram (RBD) of the system - operational configuration

Every system comprises eleven LRUs selected from six different types. The distinction among different types is a matter of the design of the model concept. Every type may have unique properties such as failure data, repair data etc.; on the other hand, there can be different types of LRUs with similar inherent properties and other attributes such as identical cost or, there can be two LRUs of the same type which may have different inherent properties (e.g. two engines operating at different loads). However, the attribute according to which a type will unify a set of different LRUs will be whether or not all members of that type share the same spare parts. Hence, the number of types in a problem will be the different number of spare parts types. In this problem, eleven LRUs in a system "consume" six different types of spare parts.

The system composition along with the failure data, repair/replacement data, repair at level B depot and shipment times are given in table 1. Note that only mean values are shown in the table, however, essentially each process should have a descriptive distribution as well. In the current model all the failure distributions are Exponential where the mean appears as the Mean Time To Failures (MTTF – sometimes appears as MTBF for Mean Time Between Failures), all Replacement distributions are Normal where the mean appears as Mean Time To Repair (MTTR) with relatively small standard deviations and all Repairs at the depot and the shipment times are assumed Constants.

					Shipment to level B		
LRU #	Type	MTTF	MTTR	Repair times at level B	FIELD1	FIELD2	FIELD3
1	1	3000	8	88	0	24	72
2	2	12000	12	74	0	18	54
3-5	3	500	8	16.8	0	45	135
6-7	6	6400	62	36	0	72	216
8-9	4	18000	24	100	0	24	72
10-11	5	2400	6	74	0	12	36

Table 1 - Failure repair and recycling data of the proposed model

Each field contains a local storage. Additionally, there is a repair Depot located at Field 1, but which serves equally all fields. The Depot has also its own storage. A diagram of the logistic cycle is shown in figure 2:

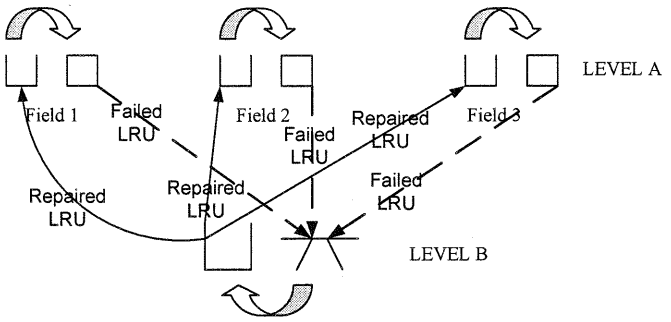

Figure 2 - Diagram of the logistic cycle of the proposed model.

This logistic cycle is a typical[2] supply chain structure for almost any industrial (civil or military) enterprise. Although enterprises may have different operating environments or management with their own special rules, still the general logistic structure can be described as such and any exceptions are implied through data or additional logical rules.

In this model, upon failure, the failed LRU is shipped to level B for a repair. At the same time two demands arise: one demand is for a spare part of the same type from the local storage (the field storage). The other demand is made by the field storage, for a spare of that type, from the level B storage; this will balance the cycle since it is expected that the repaired LRU at the depot will return into the depot's storage. There are two possibilities for each demand:

A) There is a spare available (instantly), in which case it responds to the demand and the demand is erased. With regard to the demand of the field from its local storage, a replacement/installation process will start whereas in the case of the field's storage, a spare will be sent off from the depot's storage to the storage of the demanding field.

B) There aren't any spares available, in which case the demand will hold until a spare arrives at the corresponding storage.

   Upon arrival of the failed LRU at the depot, a demand for a repair team arises as well and the same logic is then applied.

5 - a Proposed Solution – the Hybrid Approach

Let us propose a spare parts optimisation, which complies with definition 1 and 2.

Analytic prediction of the Waiting Time

[2] The generic structure can change to multiple levels (not necessarily limited to A and B) and multiple fields. There can also be multiple depots in each level. The levels are also known as echelons.

The starting point of the analytic approach (Waiting Time) suggests a single component that reaches a storage facility at a rate λ. This rate is assumed to be constant[3], representing the flow of units from the field. Upon arrival the failed unit is repaired and returned to storage. The time required for this process is the recycling time T_c.

If there are initially q spares in the storage, then a lack of spares will be caused only if upon the arrival of the (q+1)th unit to the storage the first one has not yet returned from the recycling process, namely,

$$T = \sum_{i=1}^{q} t_i < T_c \qquad q \geq 1$$

t_i is the time interval between the arrival of the i^{th} and the $(i+1)^{th}$ unit.

Since t_i may be assumed to be exponentially distributed, T has a Gamma distribution with q degrees of freedom. Hence,

$$f(T) = G_q(T) = \frac{\lambda(\lambda T)^{q-1} e^{-\lambda T}}{\Gamma(q)}$$

where $\Gamma(q) = (q-1)!$ for $q \geq 1$ and q is an integer.

If $T<T_c$ it means that the system is awaiting a spare and the time it would take is $Tc-T$. Thus, the average waiting time in a history would be

$$T_w = \int (T_c - T) \frac{\lambda(\lambda T)^{q-1} e^{-\lambda T}}{\Gamma(q)} \mathrm{d}T$$

$$= T_c D_q(T_c) - \frac{q}{\lambda} D_{(q+1)}(T_c)$$

where $D_K(X)$ represents the cumulative Gamma distribution. K is the order of the distribution and $X = \lambda T_c$ is the scale parameter.

If the component is a discarded component (namely is condemned upon failure) it will have a slightly different development, but essentially its turn around time can be regarded as infinity or even by considering it as the length of the service time (further developments of this theme can be found in Dubi, 2003).

Unavailability Sensitivity calculation by MCS

The ability of a Monte Carlo calculation to obtain time dependent statistical quantities enables the calculation of additional important metrics other than the system performance. One of these quantities is the Sensitivity. Here the Sensitivity is an array of fractions, such that every type has its own sensitivity. The Sensitivity can be related to any defined performance target. For example, the Unavailability Sensitivity of type i is defined as the relative portion of the down time (the time at which the system is unavailable), contributed by any component of type i. The following examples may further clarify this definition. Consider a system which comprises five components of three different types as shown in Figure 3. A dark component represents a failed component. The sensitivity will be

[3] A "Constant failure rate" leads to an exponentially distributed PDF.

calculated as follows. Each time the system is failed, for each failed component in the system, repair the component ad hoc, and if the system is repaired as a result of this, then this component is responsible for the failure. If the Sensitivity is a Failure Sensitivity then a unity is added to the type of the component. If the Sensitivity is an Unavailability Sensitivity then when the system will be repaired again, the downtime since the last failure will be added to all those types which were responsible for the failure. The following three cases in figure 3 are examples for three different Sensitivity outcomes.

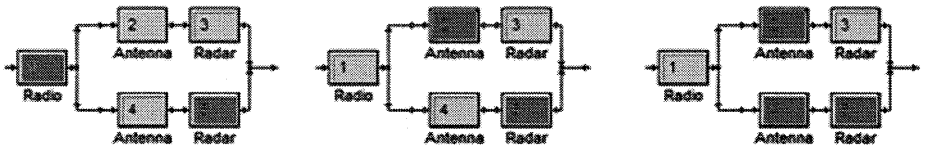

Figure 3 - Different system failure configurations and responsible components.

In case A, components 1 and 5 are failed. However, repairing (ad hoc) component 5 will not bring the system back to operation, so this component is not regarded as "responsible" for the system failure. Only component 1 (Radio type) is. In case B, 2 and 5 are failed. Repairing either one of them will repair the system. Hence, both (Antenna and Radar types) are regarded as responsible for the system failure. In case C, 2, 4 and 5 are failed. Note that this can happen only if 2 failed last or else the failure point of the system would have been when 2 failed and only one of 4 or 5 failed too (as in case B). In this case repairing 4 or 5 will not repair the system, so only 2 (Antenna type) will be held the failure cause.

This concept does not have a simple explicit mathematical definition but experience indicates that it is an excellent measure of the relative contribution of each type to systems failures. The average system down time for each type, T_{di}, accumulated by the above process is then normalised by dividing it by the sum of all types, to obtain the sensitivity of type i, namely:

$$s_i = \frac{T_{di}}{\sum_{j=1}^{n} T_{dj}}$$

(n is the number of different components types in the system).

The total average down time of the system, over the system's life time $T_{\max}$ is now assumed to be $T_d = \sum_{i=1}^{n} T_{dj} < T_c$ and the average unavailability is the ratio of the down time to the mission time $U = \frac{T_d}{T_{\max}}$.

In fact, the Sensitivity array is an output related to the system. It summarises the whole scenario, with data, system structure, logical rules, Resources policy and whatever else affects the system performance through its life history. Therefore, the analytic calculation does not "care" about the system structure or any of the problem complexities. The

sensitivity and the unavailability are the only quantities, which are relevant for the optimisation process as parameters.

Let us define a new quantity u_i, that is the unavailability contributed by type i, as $u_i = U \times s_i$.

Approximately $U = \frac{\sum_i T_{di}}{T_{\max}}$ and by its definition $u_i = \frac{T_{di}}{T_{\max}}$.

The former is thus a function of the latter:

$$U = \sum_{i=1}^{n} u_i$$

the Hybrid Link

The assumption is that u_i is a linear function of the Waiting Time (the average time interval since demanding a spare, until the spare is available) of type i, T_{Wi}, namely: $u_i(q_i) = A_i \cdot T_{Wi}(q_i) + B_i$ where A_i and B_i are constants to be determined in the course of the optimisation.

Thus, it is assumed that the unavailability as function of spare parts is a surface in the spare parts domain, having the form:

$$U = \sum_{i=1}^{n} \left(A_i \cdot T_{Wi}(q_i) + B_i \right) \qquad (1)$$

Let $U^{(j)}$ denote a specific MCS calculation with a given storage strategy. Let u_i^j and u_i^k be the partial type unavailability in the j^{th} and k^{th} calculations respectively, then the following equations provide two unknowns A_i and B_i.

$$u_i^{j_1} = A_i \cdot T_{Wi}(q_i^{j_1}) + B_i$$

$$u_i^k = A_i \cdot T_{Wi}(q_i^k) + B_i$$

One extreme, U^∞ denotes a calculation with unlimited spares (the calculation provides the maximum number of spares used, and this finite number is denoted by q_i^∞). In this case, $T_{Wi}(q_i^\infty) = 0$ since it will result in never needing to wait for spares. The other extreme would be zero spares $q_i = 0$, denoted by U^0.

The equations then take the forms

$$\begin{cases} u_i^\infty = B_i \\ u_i^0 = A_i T_{Wi}(0) + B_i \end{cases} \qquad (2)$$

Eq. (2) is the starting point of the process, A_i and B_i are inserted into expression (1) and marginal analysis is performed analytically. Moreover, an additional benefit of U^∞ and U^0 is that they indicate the boundaries of the performance. The process is then to search the cheapest way while progressing towards U^∞. It is worth mentioning that the MC

simulation provides the lower bound U^{∞} with the least cost out of the very large number of combinations. Therefore, it suggests that progressing towards that, doing the "best" step every time, wil Every logistic element, which can support the system availability (e.g. spare parts, depots, repair teams, tools, vehicles etc.) may be regarded as resource. Resources may also be referred to as supporting other target functions rather than Availability, such as power generation, profit etc.; in such cases, Resources can also be 'good weather', papers (on which a 'good contract' is put together) and so on.I end up in meeting the optimum.

Marginal Analysis

Once the unavailability can be predicted for any given allocation, the optimisation algorithm is carried out by marginal analysis; that is allocating at each step a resource (a spare parts or a repair team) to the type which seems to contribute the highest availability per cost unit. So, the more sensitive and cheaper a certain type is, the more likely that this kind is to be added.

Let us denote the unavailability as function of its Resources by $U(\mathbf{Q})$, and q_{ji} is the i^{th} element in the j^{th} row of the matrix $\mathbf{Q}$, representing the number of Resources of type i in field j. Denote by $U(\mathbf{Q}^{ij})$ the unavailability with a unit added to the ij^{th} element in the matrix $\mathbf{Q}$, then its marginal contribution to the availability will be $\beta_{ji} = \frac{U(\mathbf{Q}) - U(\mathbf{Q}^{ij})}{c_{ji}}$, where c_{ji} is the cost of the unit added. The optimisation algorithm will select the element with the maximal marginal contribution and will add one unit to it. Then, this process will keep going until a new simulation will provide accurate results with the yielded unavailability and its corresponding set of sensitivities. This will be repeated until the required criteria have been reached.

thePenalty of the Compromise

Notwithstanding the optimum may never be found, since a level of accuracy must be sacrificed by making the analytical predictions, however, the justification for this sacrifice lays in the argument that the optimum point is lying in a flat-wide valley, and reaching that valley is as effective. This argument can be supported by the idea that a system which does not have such a flat-wide valley but a rather dramatic behaviour around the real optimum would be too sensitive to survive a justifiable period of service time, particularly where the metrics are stochastic.

The Limitation to Spare Parts and Repair Teams

The Hybrid Optimizer is very efficient for problems involving spare parts and repair teams optimisation in multi-field and multi-echelon logistic environment. Yet, it is limited to optimise only those resources since it does not consider other supportive resources or parametric factors, such as preventive maintenance or different policy of management etc. This stems from the difficulty in developing analytic approaches for predicting the performance for multi-variable environment. It demands an extensive study of how dependently or independently each resource generically influences the performance whereas the nature of every model can be totally different from another, which makes the development of a general recipe unrealistic.

6 - Obtaining the Density of the Resources Space by Random Samples

Our starting point for an improved approach is to sample points in the Resources space to create Unavailability density in that space. By doing so, we should be able to perceive what the density of the average unavailability in the Resources space is, and to comprehend what the order of magnitude of the computational cost is of building up such scaffoldings for optimisation or search process (a deeper discussion can be found in Keane and Nair, 2005).

Here the Resources space is transformed into cost with a simple Cost function. Therefore, the density diagram will show Unavailability against Cost. It is worth mentioning that the graphic illustration of the density as a function of the Cost is not as informative as that of the Resources, because some similar Costs may result from completely different Resources allocations (distant in the Resources space). However, it is straight forward to break down the Costs to their corresponding Resources allocations when required. Also, apparently it is the only way for viewing that density in a two-dimensional diagram.

the Sampling Process

The number of allocated Resources strongly affects system performance and for every single allocation, a complete Monte Carlo simulation which comprises a significant number of Histories must be carried out. In the current process we randomly sample the number of each Resource and execute a simulation. Of course, if we sample each Resource independently and uniformly in its corresponding region then the resulting Cost will not be uniformly distributed since the Cost is a summation over 25 independent variables and hence has the properties of the Sample Average where according to the Central Limit Theorem its probability density function approaches a Normal distribution around the mean value. Then, certainly this would not be a uniform distribution. In order to gain a more even coverage we carry out conditional sampling, starting with the Cost (Uniformly) as illustrated in figure 4 and walking backwards. The range of Costs lays between zero and the maximum Cost. The finite maximum Cost is determined by the Cost function applied for the 98.5% Sufficiency point. Each type is then allocated with maximum needed spares (with 98.5% probability of never having a lack when required), and the same principle is applied for the repair teams.

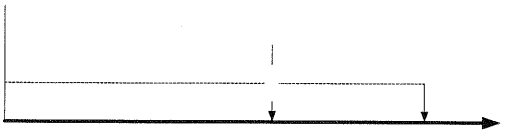

Figure 4 - Sampling the Cost uniformly between zero to Max.

Having sampled the Cost we then investigate what the distribution among the types would need to be. This is done by sampling a random number for each type and normalising it with the sampled cost. The Pie chart in figure 5 illustrates an example for the cost distribution.

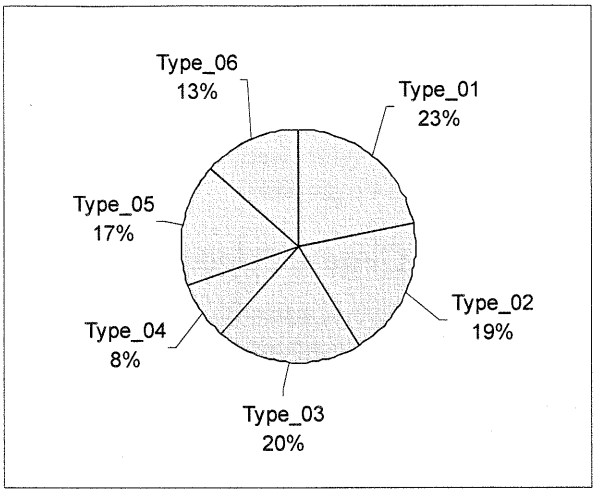

Figure 5 - Pie chart of the Cost distribution among the different Line Replaceable Units types in the problem.

The number of spares of each type will be the size of its corresponding segment divided by the cost of that type and rounded. The same principle is then applied to obtain the distribution among the four different storages of spares of a given type.

Having done that, we have a spares allocation for all storages and we obtain the Unavailability as a function of the Cost using Monte Carlo simulation. Repeating this process many times, each time with new sampled allocation, we then create the topology of the Unavailability in the Resources space. This is clearly seen in figure 6.

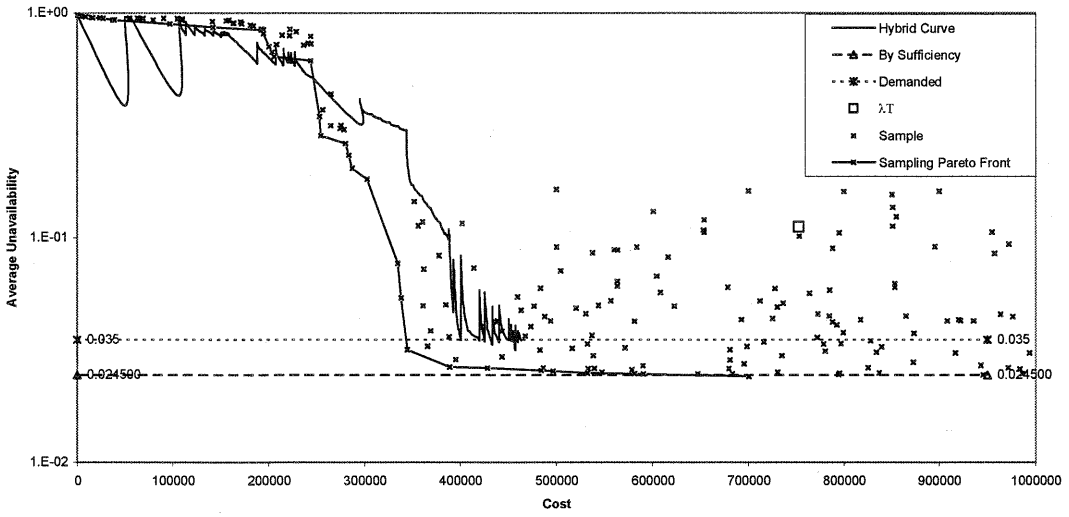

Figure 6 - Unavailability in the Resources space when sampled uniformly, from which the resulting Pareto Front is compared with the Hybrid Curve, λT is added to illustrate a common allocation which is far away with respect to the optimum.

These results show the Unavailability density in the Cost space, plus the optimisation curve obtained by the Hybrid Optimizer and a Pareto front fitted on the sampled space for comparison. One can conclude that most points in the space are far away from the optimal region and indeed it is not trivial to obtain the optimum and that indeed the Hybrid optimisation curve meets this pattern quite well. Still, the Hybrid curve is not the best curve and some better allocations can be proposed.

7 - Search Process Using Genetic Algorithms

The above diagram shows that the Hybrid optimisation can be improved upon. However, computation-wise it demands many more simulations. The good news is that the majority of the simulations are "redundant". We are interested only in those which are close to the optimal curve (if that indeed is the optimal one). So, if we could "get rid" of them we would be able to study the process with considerably cheaper computational cost. The next step is to sample significantly less points in the Resources space and instead of proceeding to sample randomly, we use GA to create more new points as descendants of existing ones. These descendants are the result of bringing together two points, where the closer a point is to the optimum the better chance it has to be selected for reuse. The GA "pushes" the density towards lower Unavailability and cheaper Cost. By doing so, it is expected that less calculations will have to be carried out since most of the "unnecessary" ones are unlikely to take part. Still, one of the underlying principles of GA is that there should not be deterministic decisions when selecting new points, but to give some chance to the "less promising" points. This prevents degeneration or dead-ends in the process.

Recall that in the current example problem there are 25 different entries, each of which can vary between zero to its maximum, we break down this vector to a binary vector which represents the same values. Therefore, each resource will thereby be allocated with a stack of binary bits, where the number of bits of every resource should cover the corresponding maximum allocation[4] (sufficiency), and all the stacks together create a string of binary values. The next step is ranking the strings according to a criterion. There are many ways to determine the criterion by which the strings should be sorted. At this stage we attribute each allocation with a "mark" which is derived from the simulation. The mark is a relative (normalised) function value $\alpha(\underset{\sim}{Q})$, of the Resources allocation $\underset{\sim}{Q}$ of each string (spares allocation in binary units) in the form: $\alpha_i(\underset{\sim}{Q}_i) = \alpha_i(U_i(\underset{\sim}{Q}_i), C_i(\underset{\sim}{Q}_i)) = \frac{c_1}{U_i \times C_i}$, where c_1 is a the normalising constant $c_1 = \frac{1}{\sum_j \frac{1}{U_j \times C_j}}$ such that $\sum_j \alpha_j = 1$

Hence, the set α can be regarded as set of probabilities of mutually exclusive and complementary events such that it is possible to sample a string from the entire population

[4] It is worth noting that upon allocating the stack for each resource, it may increase the degree of freedom since the maximum values of binary stacks are greater or equal the corresponding sufficiencies; however, this only slightly increases the search process.

with probability α_i for the ith string. Next, two strings are selected, with the condition that they are different, and merged as described below.

Then we select for both strings, the place at which they will cut. If each string is built up from Z binary elements, then the current sampling is with uniform distribution between 1 to $Z-1$. Let the cut point be M, then we cut both strings to two pieces, piece A from 1 to M and piece B from M+1 to Z. We then switch between piece B of the two strings and create two new-born strings. We repeat this process until we have created an equal number of strings as per the original population (we maintain the same number in the population), and simulate for each new-born string. We should say that a genuine parent can be sampled multiple times as long as it does not get merged with itself. By this method (see more in Goldberg, 1989), the population will evolve to obtain results more and more toward lower Unavailability and lower Cost at the same time (note that these two objectives are in competition).

Figure 7 shows the results of the above described GA process for the same model:

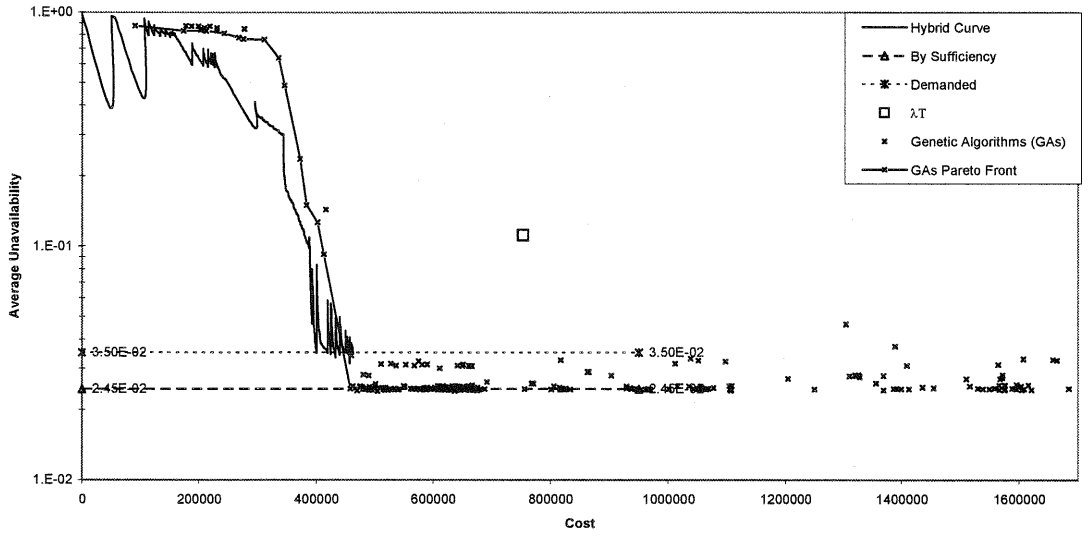

Figure 7 - Unavailability in the Resources space produced by GA, from which the resulting Pareto Front is compared with the Hybrid Curve, and λT is added again.

Figure 7 clearly shows that the space has been extended. As explained above, this is caused by the replacement of the allocation vector with binary one which covers more values. It is clearly seen that less "wasteful" samples were carried out and that the population approaches massively towards better (lower) Unavailability and to lower Cost.

Still, these results are not completely satisfying and we believe can be improved upon. Two main factors can be suggested for improvement: changing the determination of α and attempting to increase the dispersion of the points. This requires a further study that may consider other search methods, such as the Ranking method.

8 - Summary & Conclusions

We have seen that current state-of-the-art optimisation techniques in the RAM problems arena can be improved upon. This was illustrated by comparing random samples with the Hybrid optimisation (as illustrated in figure 6). Sampling the Resources space randomly clearly shows that better designs can be found. Still, the random sample is not regarded as a competitive search technique, since we achieved that through a considerably high computational cost. Hence, better search algorithms could be explored, which would allow for better designs while maintaining a reasonable execution time.

We proposed Genetic Algorithms as these benefit from a degree level of random sampling but at the same time channel this randomness towards a desired direction in an educated manner. As a result, we saw that fewer random points were shown in the results and that the search channelled towards the desired objective. However, since that brought up a competition between two objectives, we identified that the optimal results were not as good as those of the random search. This implies that the Genetic Algorithms technique can be improved upon, with an alternative search technique, such as Ranking.

Apart from the improvement of the results and of the effectiveness of the search process, the benefit of applying GA as the search engine to the RAM problems makes the optimisation process useful for any model scenario, because it does not require a preliminary knowledge of the physics of the problem. As a result, the optimisation is valid for any desired objective function and for any factors without regard to their effect on that objective function.

References

J.S. Bendat and A.G. Piersol. *Measurement and Analysis of Random Data.* John Wiley & Sons, 1966.

A Dubi. *Monte Carlo Applications in Systems Engineering.* John Wiley & Sons, 2000.

A. Dubi. *System Engineering Science, Analytical Principles & Monte Carlo Methods.* MIRCE Science Limited, 2003.

A.J. Keane and P.B. Nair. *Computational Approaches for Aerospace Design, The Pursuit of Excellence.* John Wiley & Sons, 2005.

D.E. Goldberg, *Genetic Algorithms in Search, Optimisation, and Machine Learning,* Addison-Wesley Publishing Company, Inc, 1989.