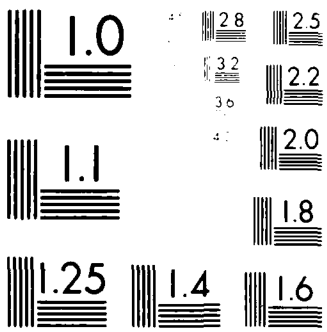
AD-A170 357 INTEGRATED COMMUNICATION NAVIGATION AND IDENTIFICATION AVIONICS RESOURCE ALLOCATION(U) ANALYTIC SCIENCES CORP READING MA B E GRIFFITHS ET AL JUL 86 AFHRL-TR-86-10 1/1

UNCLASSIFIED F33615-82-C-0002 F/G 9/3 NL


1.0
2.8
2.5
3.2
2.2
3.6
2.0
1.1
1.8
1.25
1.4
1.6

AFHRL-TR-86-10

AIR FORCE

AD-A170 357

HUMAN RESOURCES

INTEGRATED COMMUNICATION, NAVIGATION, AND IDENTIFICATION AVIONICS RESOURCE ALLOCATION

Barry E. Griffiths
David E. Miller

The Analytic Sciences Corporation
One Jacob Way
Reading, Massachusetts 01867

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

July 1986

Final Report for Period March 1982 - March 1984

Approved for public release; distribution is unlimited.

LABORATORY

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

NOTICE

This report has been reviewed and is approved for publication.

JAMES C. McMANUS
Contract Monitor

BERTRAM W. CREAM, Technical Director
Logistics and Human Factors Division

DENNIS W. JARVI, Colonel, USAF
Commander

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFHRL-TR-86-10
6a. NAME OF PERFORMING ORGANIZATION The Analytic Sciences Corp. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Logistics and Human Factors Division
6c. ADDRESS (City, State, and ZIP Code) One Jacob Way Reading, Massachusetts 01867	7b. ADDRESS (City, State, and ZIP Code) Air Force Human Resources Laboratory Wright-Patterson Air Force Base, Ohio 45433-6503
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Human Resources Laboratory 8b. OFFICE SYMBOL (If applicable) HQ AFHRL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-82-C-0002
8c. ADDRESS (City, State, and ZIP Code) Brooks Air Force Base, Texas 78235-5601	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62205F; PROJECT NO. 1710; TASK NO. 00; WORK UNIT ACCESSION NO. 26

11. TITLE (Include Security Classification)
Integrated Communication, Navigation, and Identification Avionics Resource Allocation

12. PERSONAL AUTHOR(S)
Griffiths, Barry E.; Miller, David E.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Mar 82 TO Mar 84	July 1986	30

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
05	08	
14	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
communication; fault-tolerant avionics; identification avionics; mean time between critical failure; mean time between failure; mean time to repair

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Integrated Communication, Navigation, and Identification Avionics (ICNIA) architecture is being designed to replace a number of discrete avionics components with an integrated, modular system. In order to maximize the usefulness of ICNIA-supported functions, a new resource allocation technique is needed. This report presents a preliminary assessment of several methods of reallocation and describes measures of performance for all of these techniques.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Nancy A. Perrigo, Chief, STINFO Office	22b. TELEPHONE (Include Area Code) (512) 536-3877 22c. OFFICE SYMBOL AFHRL/TSR

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Item 18 (Concluded):

logistics analysis
mission completion success probability

SUMMARY

The Integrated Communication, Navigation, and Identification Avionics (ICNIA) architecture is being designed to replace several discrete avionics components with an integrated, modular system. The goals of the project include improved system reliability and decreased size and weight. In order to meet these objectives, the ICNIA technology must allocate available resources in order to respond to changes in the mission environment or to compensate for the loss of system components. Thus, the reliability of ICNIA depends on both hardware design and on the efficiency of the resource allocation technique.

This report describes the ICNIA resource allocation problem and presents the mathematics that can be used to approach a solution. In particular, mathematical programming methods, sequential allocation algorithms, and combinational algorithms are each evaluated for their ability to solve this problem. A preliminary analysis of these techniques was completed by defining several measures of performance. The results of this analysis are given, along with several recommendations for improving the design of an ICNIA resource allocation technique.

PREFACE

This report documents research into the behavior of resource allocation algorithms for Integrated Communication, Navigation and Identification Avionics (ICNIA) and methods for evaluation of the performance of such algorithms. These findings are results of the Fault Tolerance Analysis task of the Impact Analysis of ICNIA, Air Force Contract No. F33615-82-C-0002. This work is jointly supported by the Air Force Human Resources Laboratory and the Air Force Wright Aeronautical Laboratories. The guidance and support of Mr. James C. McManus and Lt Lee H. Dayton are greatly appreciated.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
1.1 Background	1
1.2 Approach	1
1.3 Organization of Report	2
2. OPTIMAL ALLOCATION FRAMEWORK	2
2.1 Algorithms and Implementations	2
2.2 Dynamic Optimization	3
2.2.1 Priority Sets	4
2.2.2 State of System Health	5
2.2.3 Resource Strings	5
2.2.4 Control Vector	5
2.2.5 Form of Objective Function	6
2.2.6 Characteristics of Optimal Solutions	7
2.3 Static Optimization	7
2.3.1 Problem Simplification	7
2.3.2 Example	9
3. STATIC ALLOCATION ALGORITHMS	11
3.1 Mathematical Programming	11
3.2 Sequential Allocation	12
3.3 Combinatorial Methods	13
3.3.1 Computational Trade-Offs	13
3.3.2 Combinatorial Compromise	14
3.4 Performance Comparison	14
4. ALGORITHM EVALUATION	16
4.1 Performance Measures	16
4.1.1 Measures of Algorithm Effectiveness	16
4.1.2 Computability Measures	19
4.2 Evaluation Methods	20
5. SUMMARY AND RECOMMENDATIONS	21
5.1 Summary	21
5.2 Recommendations	22
REFERENCES	23

LIST OF FIGURES

Figure		Page
1	Dynamic Optimization	4
2	Example: System State (Full-up)	10

LIST OF TABLES

Table		Page
1	Static Optimization Example Function Set Priorities	9
2	Example: Resource Strings	10
3	Algorithm Performance Comparison	15
4	Algorithm Effectiveness Measures	17

1. INTRODUCTION

1.1 BACKGROUND

The Integrated Communication, Navigation, and Identification Avionics (ICNIA) system is being designed to replace a number of discrete avionics components with an integrated, modular system. This project has several design goals, including decreased system size and weight, improved reliability, and decreased logistical support requirements. The approach being taken is that of active redundancy; that is, available system resources will be reallocated to perform particular tasks in response to changes in mission requirements or to compensate for the loss of system components. As such, the reliability of ICNIA in providing designated functions depends on both the hardware design and the efficiency of the resource allocation technique.

New methodologies have been required to project the probable performance of alternative ICNIA designs. One such tool, the Missionized Reliability Model (MIREM) (Veatch, Calvo, Myers, and McManus, 1985), is used to evaluate several measures of the reliability of ICNIA hardware designs in providing designated functions. MIREM is designed to be independent of ICNIA resource allocation techniques and allows the best performance achievable with a particular hardware design to be determined. The objectives of the current effort are to complement MIREM by defining measures of performance for resource allocation techniques, and to provide a preliminary assessment of the strengths and weaknesses of several such alternative techniques.

1.2 APPROACH

The objective of the resource allocation technique is viewed here as maximization of the expected usefulness to the operator of the ICNIA-supported functions. The approach taken in evaluating the resource allocation problem is to employ the framework of optimal control theory. In order to make use of the substantial body of work on optimal control theory, certain notions such as "usefulness" are first made concrete. Reasonable simplifications are made in order to improve the analytical tractability of the problem.

The overall optimization framework is then used to formulate measures of allocation performance for general suboptimal allocation techniques. Since these proposed measures are derived directly from the previously stated optimization problem, they capture important features of allocation performance. In addition, several measures of the computational burden of allocation techniques are proposed. In this way, an explicit evaluation of the allocation benefits and computational costs of any allocation technique can be made. Such evaluations can be used in making trade-off analyses during the design cycle of ICNIA resource allocation techniques.

Because no well-defined allocation techniques have yet been developed by ICNIA contractors for evaluation, this report provides preliminary assessments of the performance of several generic resource allocation techniques. Each of the techniques -- mathematical programming methods, sequential allocation algorithms, and combinatorial algorithms -- is evaluated for its ability to provide a good-quality solution to the ICNIA resource allocation problem.

1.3 ORGANIZATION OF REPORT

This report is organized in five chapters. The introductory chapter establishes the scope of the effort and the approach taken. Chapter 2 formulates the optimal control theory framework for the ICNIA resource allocation problem and includes a simple example of these concepts (Section 2.3.2). Chapter 3 addresses the preliminary assessment of alternative allocation techniques, and several measures of allocation performance are proposed in Chapter 4. A summary and recommendations are presented in Chapter 5.

2. OPTIMAL ALLOCATION FRAMEWORK

2.1 ALGORITHMS AND IMPLEMENTATIONS

The techniques to be applied in allocating available resources to accomplish desired functions can be divided conceptually into two phases. The first phase is algorithmic solution; the second phase is implementation.

The algorithmic solution phase describes how allocation decisions are made. An allocation algorithm is defined in terms of its inputs, calculations, and outputs. Inputs to such an algorithm include diagnostic information about the state of system health and operator interactions. Outputs include statements about how functions will be supported by the system resources. The calculations describe the way in which the outputs are derived from the inputs.

The implementation phase describes how a particular algorithm will be performed in a particular hardware system. Implementation includes the software and hardware partitioning of the algorithm. In general, although there may be many possible implementations of a given algorithm in a given hardware system, they will all have identical performance from the standpoint of providing ICNIA functions. Proper partitioning is needed to satisfy the computational capabilities of the system. Consequently, the allocation algorithm must be designed before an implementation can be selected.

The allocation algorithm must be optimized within the framework imposed by the ICNIA system constraints. These constraints can be stated as follows. The algorithm must allocate limited system resources to perform multiple functions. It must provide for dynamic reconfiguration to support operator-defined changes in preferred functions and to allow graceful degradation of overall system functionality as resources fail. The timing of resource failure is not known, although the statistical reliability of resources is known. This problem can best be formulated in terms of dynamic stochastic optimization.

In Section 2.2, the ICNIA resource allocation problem is put in the form of dynamic stochastic optimization, and some of the characteristics of an optimal solution of this problem are noted. Under certain conditions, this problem can be simplified considerably. This results in a static deterministic optimization problem, as shown in Section 2.3.

2.2 DYNAMIC OPTIMIZATION

The fundamental concept in using dynamic optimization techniques in allocating ICNIA resources is illustrated in Figure 1. That concept is that the "usefulness" of an ICNIA resource allocation depends on the priority attached to the functions implemented. Ordering the priorities of all possible sets of functions is the responsibility of

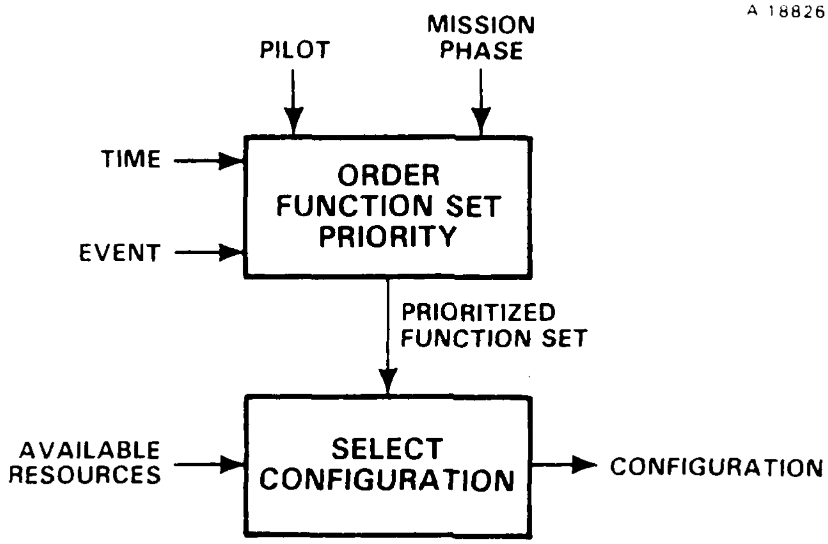


Figure 1. Dynamic Optimization.

the pilot and/or mission planner; in this way, the allocation algorithm is told what is desired. In turn, the allocation algorithm must use both the function set priorities and information about available resources to select the best allocation available.

2.2.1 Priority Sets

It is important to note that sets of functions, rather than individual functions, must be arranged in order of priority. This approach is more general, as it includes the ordering of individual functions as a special case. By ordering sets of functions, it is possible to allow the selection of two functions, each individually slightly less important, rather than one function which is individually slightly more important. For example, the "limp home" functions may include any one communication system and any one navigation system. Ordering functions individually might result in the selection of two communication systems or two navigation systems, which would be less desirable.

2.2.2 State of System Health

It has been pointed out (Veatch et al, 1985) that the condition of the system hardware at any time can be specified by the number of healthy units in each pool of resources. ICNIA will contain facilities for Built-In Test (BIT), which will allow the allocation algorithm to be informed as to the current state of system health. It should be noted, however, that there may possibly be device failures which are not detected by BIT. Thus, the state of system health may never be known perfectly.

2.2.3 Resource Strings

There are generally a limited number of ways in which any ICNIA function can be accomplished. Each such way can be specified in terms of how many units are required from each pool of system hardware. Such a specification is called a resource string in this work. If desired, all of the possible resource strings for each function could be formed into a table of resource requirements of the form r(i,j,k), where

i is the function number

j is the resource string number (for the ith function)

k is the pool number (for the jth resource string for the ith function)

r(i,j,k) is the number of units required in pool k to accomplish function i using resource string j.

Note that resource requirements can be fractional, representing time-shared resources. By specifying pools rather than individual devices, the allocation problem is held to a practical dimension. Selection of particular devices within a pool can be handled by a very simple local scheduling algorithm.

2.2.4 Control Vector

An allocation decision must specify not only which functions are to be performed but also which resource strings or paths will be used to perform them. This is because not all resources of one type can be connected to all resources of another type; only certain interconnections are possible.

If all possible resource strings for each function are listed, then the allocation decision consists of selecting which particular string is to be used for each function. The list of all of these resource string selections, taken together, constitutes the control vector.

2.2.5 Form of Objective Function

All of the above components of the dynamic stochastic optimization problem can now be assembled. An optimal allocation algorithm generates a function, $\underline{h}$, that maps all available information about system health, I, into a control vector, $\underline{u}$:

$$\underline{u} = \underline{h}(I) \tag{1}$$

in order to maximize the objective function

$$J(T) = E\left\{\int_0^T p(f(\underline{x}(t),\ \underline{u}(t)),\ t)dt\right\} \tag{2}$$

where

f denotes the system functions actually operational given a control vector $\underline{u}$ and a system health state $\underline{x}$

p denotes the priority of this set of functions

T is the time between maintenance actions

E denotes the statistical expectation due to uncertainties about $\underline{x}$.

This objective function reflects all of the essential characteristics of the ICNIA allocation problem. It is important to note that the allocation algorithm may need to take into account explicitly the probabilistic nature of system health. It is also important to note that the algorithm must be concerned with the performance of the system over the whole time between maintenance actions. This is particularly significant if ICNIA-equipped aircraft are expected to operate from austere bases, where replacement modules may not be available after each mission.

2.2.6 Characteristics of Optimal Solutions

Optimization problems of this general type have been addressed in the literature (Chizeck, 1981; Griffiths, 1983; Rishel, 1978; White, 1974). Some of the characteristics of the solutions to these problems should be pointed out.

First, reconfiguration time is implicitly considered in the optimization, since the whole time interval is of interest. For time-critical functions such as Identification-Friend-or-Foe (IFF) Transpond, this may result in the simultaneous operation of a function in two or more independent resource strings in order to provide an instantaneous backup. Furthermore, similar considerations would apply to the time required to turn off and reinitialize system resources.

Second, an optimal algorithm can make use of all information about the state of health of the system. This may include BIT, techniques for soft failure detection, and the monitoring of functional outputs. These sources of information are combined using probability models both for resource failures and for the effectiveness of the fault detection methods employed.

Third, the allocation algorithm may actually select a control vector partly in order to learn more about the state of system health. This "dual control" aspect (Feldbaum, 1960/1961; Griffiths, 1983) results from an implicit trade-off between the cost of reduced functionality now versus improved knowledge of system health and improved functionality in the future.

Fourth, due to the probabilistic and dual-control aspects of this problem, optimal solutions are generally very difficult to obtain.

2.3 STATIC OPTIMIZATION

2.3.1 Problem Simplification

The difficulty in solving the dynamic stochastic problem arises from two factors. First is the implicit recognition of the importance of speed of reconfiguration; second is the imperfect knowledge of system health. Consequently, the problem can be greatly simplified if two approximations can be made to hold. If there is an insignificant penalty for downtime while the system reconfigures and if there is an insignificant penalty for decoupling resource allocation and failure detection (i.e., only currently available resource health information is used), then the dynamic stochastic optimization problem resolves into a sequence of static problems.

To simplify the discussion, it will also be assumed that there is perfect knowledge of system health (100% BIT effectiveness), although static optimization can also be applied to the case of imperfect BIT. For the static case, the reconfiguration algorithm is solved each time there is a change in function set priorities or in the state of system health. A control vector $\underline{u}$ is selected in order to maximize

$$J = p(f(\underline{x}(t),\ \underline{u}(t)), t) \qquad (3)$$

where all terms are defined as before. Note that this optimization problem itself is not concerned with anticipating future effects, nor does it have a random element. This is because the temporal and random effects enter via the changes in priorities and in system health state, which signal when to perform the static optimization.

For the static problem, it can be seen that the control vector selected must not require more resources from any pool than are available. That is, referring to Section 2.2.3,

$$\sum_i r(i,\ u_i,\ k) \leq x_k \text{ for all } k \qquad (4)$$

where u_i is the resource string selected for the i^{th} function, and x_k is the number of healthy units in the k^{th} pool. This condition forms a set of constraints for the static optimization problem.

It should be noted that good system design can force the approximation conditions to hold. The requirement that ICNIA be able to reconfigure in 10 seconds is a way to guarantee that there is an insignificant penalty for system downtime. Similarly, if all known device failure modes are detectable through BIT, the penalty for decoupling resource allocation and failure detection is probably insignificant.

2.3.2 Example

In this subsection, a simple example of resource allocation will be presented to point out various aspects of function set priorities, state of system health, table of resource strings, and control vector. In Chapter 3, this example will be used to examine some of the strengths and weaknesses of alternative allocation algorithms.

In the example depicted in Table 1, there are three functions: F1, F2, and F3. Thus, there are eight function sets to be ranked by relative priority. One such ranking is presented here. Different priority rankings can be generated for different mission phases, and rankings can be modified by the pilot. (It is not yet known how the pilot interaction will be managed in ICNIA.) Note that it is sometimes preferable to have one function (Priority 4) rather than two functions (Priority 5). Note also that no single function has absolute priority in this example.

There are four pools of resources in this example. The state of system health depicted in Figure 2 can be represented as $\underline{x} = (1, 2, 1, 2)$, summarizing the number of healthy units in each pool (P1, P2, P3, P4). If one unit were to fail in pool P4, the system state would be represented as $\underline{x} = (1, 2, 1, 1)$.

Table 1. Static Optimization Example Function Set Priorities

Function Set Priorities	Functions
1	F1, F2, F3
2	F1, F2
3	F2, F3
4	F2
5	F1, F3
6	F1
7	F3
8	None

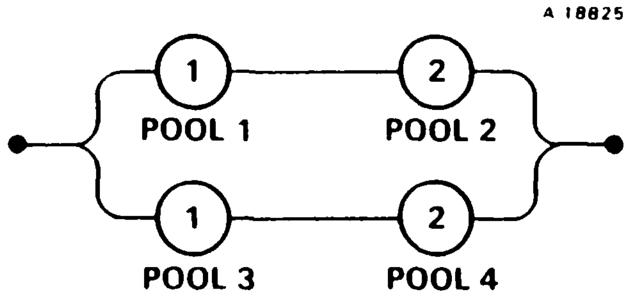


Figure 2. Example: System State (Full-Up).

The resource strings representing all possible ways of performing each of the three functions in the given system are presented in Table 2. Each of the functions can be performed in either of the two chains in the system. In addition, F3 can be performed cooperatively in both chains, although this results in increased total resource requirements due to increased overhead (compare resource strings 1 and 2 of function F3 with string 3).

Table 2. Example: Resource Strings

Function	Resource String Designation	Resource Utilization			
		Pool P1	Pool P2	Pool P3	Pool P4
F1	1	1	1	0	0
	2	0	0	1	1
F2	1	0.5	1	0	0
	2	0	0	0.5	1
F3	1	0.5	1	0	0
	2	0	0	0.5	1
	3	0.25	0.7	0.25	0.7

It can be seen that two different control vectors can exercise all three functions when all system resources are healthy. These two control vectors are u = (1, 2, 2) and u = (2, 1, 1). If u = (1, 2, 2) is selected, then:

1. F1 is performed via resource string 1 (pools P1 and P2)

2. F2 is performed via resource string 2 (pools P3 and P4)

3. F3 is performed via resource string 2 (pools P3 and P4).

If $\underline{u}$ = (2, 1, 1) is selected, the reverse will take place. It may be expected that for the general case, control vectors are non-unique -- that is, that several possible allocations will perform equally well.

3. STATIC ALLOCATION ALGORITHMS

Three categories of static resource allocation algorithms have been identified in this task. These categories are mathematical programming, sequential, and combinatorial. In this chapter, an initial assessment is made of the advantages and disadvantages of these kinds of algorithms in the ICNIA resource allocation problem.

3.1 MATHEMATICAL PROGRAMMING

The mathematical programming category includes a number of related methods. Among these are linear programming, integer programming, and a number of nonlinear programming techniques. All of these methods are well documented, and there is a great deal of available software (Kuester and Mize, 1973). When these methods are applicable to a problem, they generally yield good-quality solutions.

Unfortunately, these techniques do not appear to be applicable to the ICNIA resource allocation problem. Linear programming requires that constraints be linear in the control variables. In this application, linear programming would require that the resource requirements in each pool would increase (or decrease) steadily with changing selection of the allocation control vector. Clearly, this requirement is not met.

More generally, mathematical programming techniques require a smooth objective function and/or a convex feasible region (Luenberger, 1973). By a smooth objective function, it is meant that any two control vectors that are "close" will yield objective function values that are "close." By "convex feasible region," it is meant that any control vector interpolated between two control vectors meeting the constraints of Equation 4 will also meet those constraints. If either of these assumptions is not met, these methods will yield poor-quality results, if they can be made to work at all.

It can be seen in the example of Section 2.3.2 that the static optimization problem does not generally have a convex feasible region. The control vectors (1, 0, 2) and (2, 0, 1) are feasible, but (1, 0, 1) and (2, 0, 2) are not, indicating the nonconvexity of the feasible region. In addition, this example demonstrates that the objective function can exhibit significant discontinuities, and is thus unsmooth. As a result, it appears that standard mathematical programming methods are not applicable to the ICNIA resource allocation problem.

3.2 SEQUENTIAL ALLOCATION

The class of sequential allocation algorithms is intuitively appealing. First, priorities are assigned to each individual function. Second, an available resource string is allocated to the highest priority function. These resources are not available for subsequent allocation to functions with lower priority. Third, an available resource string is found for the function with second highest priority, and so on.

The advantages of this algorithm are that it is simple to implement and fast to execute. The fast execution speed is due to the fact that only a limited number of combinations of resource strings must be examined for feasibility, given the current state of system health.

However, the fact that the algorithm requires priorities for each individual function limits its flexibility. As was noted in Section 2.2.1, there are many situations in which such an individual ordering could yield unacceptable results.

In addition, the algorithm itself can result in unnecessary functional degradation. Because the algorithm does not consider what resources will be needed for a lower priority function while it is selecting a resource string to be allocated for a higher priority function, the algorithm may not implement all functions even when sufficient ICNIA devices are available. An example of this situation will be presented in Section 3.4.

3.3 COMBINATORIAL METHODS

In order to avoid the difficulties presented by mathematical programming and sequential allocation methods, a new type of allocation algorithm was developed. These are called combinatorial methods.

In these methods, priorities are assigned to sets of functions, as recommended in Chapter 2. All possible control vectors can then be arranged in order of desirability, according to which set of functions will be supported. Note that there will usually be several control vectors which, if they meet the constraints of Equation 4, will provide the same set of functions; these are therefore of equal desirability. All that remains is to test each control vector for feasibility given the current state of system health, starting with the most desirable control vector. The first feasible control vector is selected for implementation.

For the static case, this approach is completely optimal. Moreover, the algorithm should be reasonably simple to implement. The disadvantage of combinatorial methods is that the number of possible control vectors is likely to be extremely large. Although not all control vectors need to be examined for each reallocation, this approach may place an excessive burden on available computer resources.

3.3.1 Computational Trade-Offs

As in almost all computer applications, it is possible to trade real-time computational requirements for memory. The most direct method of implementing a combinatorial algorithm is to completely calculate the optimal control whenever a reallocation is required (due to either device failure or change in priorities). Although this approach would require little computer memory, a severe processor load would be imposed. A calculation time of several minutes is possible, although the maximum calculation time has not been determined.

At the other extreme, optimal controls could be calculated off-line and stored in onboard memory. Using this approach, memory would be required for each possible state of system health and function set priority. Although this approach would result in a negligible processor load, several megabytes might be required for table storage.

3.3.2 Combinatorial Compromise

Rather than imposing a large peak processor load or a large memory requirement, it may be possible to implement a combinatorial algorithm using modest processor loads and memory. The approach would be to precompute and store the optimal control vectors for only the next several possible changes in system health state and function set priorities. These controls would be ready for immediate use.

Such an algorithm might reside in an intermediate-level maintenance facility, in a flightline computer, or in an onboard computer. If the algorithm resides in a maintenance facility or flightline computer, a new table can be computed between missions and downloaded into the onboard computer.[1] The new table would account for all device failures and maintenance actions. If the algorithm resides in the onboard computer, the new table would be computed as a background task after each reconfiguration.

Clearly, the size of the required table increases geometrically with the number of possible failures and priority changes to be anticipated. The design of a combinatorial compromise algorithm must balance required memory, compute time, and the risk of "running off the table" after an unanticipated event. Note that a small, fast backup algorithm can also be put in place, if needed, to avoid catastrophic delays.

The combinatorial compromise offers the possibility for fast reconfiguration and moderate memory requirements. In addition, calculation of the optimal allocation as a background task would smooth processor load. A comparison of the combinatorial method with the sequential allocation method for a simple example is presented in Section 3.4.

3.4 PERFORMANCE COMPARISON

In this section, the sequential and combinatorial methods are compared for the simple example of Section 2.3.2. This example involves three functions, to be implemented in six devices. The six devices are organized in four pools in two chains as depicted in Figure 2.

---

[1] Flightline programming of avionics equipment is currently performed in advanced Electronic Countermeasures (ECM) systems.

For five of the possible system health states, the selected allocations and supported functions for both algorithms are presented in Table 3. Note that the sequential allocation algorithm is inherently less flexible than the combinatorial algorithm, as it requires functions to be assigned priorities individually. Therefore, it cannot trade off one function for two slightly less important functions. Note also that there is one case where the sequential algorithm cannot support all three functions, even though the combinatorial algorithm can. No ordering of resource strings and priorities is possible to avoid this situation; it results from the sequential nature of the search. Thus, it is seen that the combinatorial algorithm is both more flexible and more efficient than the sequential algorithm.

Table 3. Algorithm Performance Comparison

System State	Sequential[a]		Combinatorial Compromise	
	Control	Functions	Control	Functions
1, 2, 1, 2	1, 2, 2	F1, F2, F3	1, 2, 2	F1, F2, F3
1, 2, 1, 1	1, 2, 0	F1, F2[b]	2, 1, 1	F1, F2, F3
1, 2, 0, 2	1, 0, 0	F1[c]	0, 1, 1	F2, F3
1, 1, 1, 2	1, 2, 2	F1, F2, F3	1, 2, 2	F1, F2, F3
0, 2, 1, 2	2, 0, 0	F1[c]	0, 2, 2	F2, F3

[a] Priorities: F1, F2, F3
[b] Suboptimal due to algorithm inefficiency
[c] Sequential cannot trade one function for two slightly less important functions.

4. ALGORITHM EVALUATION

4.1 PERFORMANCE MEASURES

Chapter 2 has stated the optimal resource allocation problem and established the framework in which it must be solved. There are two general categories of measures for evaluating the performance of resource allocation algorithms; these are:

1. Measures of algorithm effectiveness: They evaluate how nearly the algorithm's behavior resembles that of an optimal algorithm.

2. Measures of computability: They evaluate the burden that would be placed on the aircraft computer systems if the algorithm were implemented.

In selecting a resource allocation algorithm for onboard allocation, both types of performance measures must be considered. Clearly, if one algorithm is superior to another in terms of both effectiveness and computability, that algorithm is to be preferred. If several algorithms are evaluated as providing different mixes of effectiveness and computability, then trade-off studies can be performed in order to select the preferred algorithm. Thus, these two types of performance measures can be valuable during the system design cycle.

Measures of algorithm effectiveness are found in Section 4.1. Measures of computability are presented in Section 4.2.

4.1.1 Measures of Algorithm Effectiveness

Three general algorithm effectiveness measures have been identified in this effort. They are summarized in Table 4. All are directly related to the optimization problem discussed in Chapter 2. The first of these is the Composite Utility measure, which provides a single figure of merit representing the average effectiveness of the algorithm over a predetermined time horizon. The second is the Worst Case measure. This measure evaluates the largest deviation between the functions supported by the optimal algorithm and the functions supported by the algorithm under study, over a predetermined set of credible device failures. The third measure of algorithm effectiveness is Probability of Success. This measure is closely related to the performance measures used in MIREM and represents the probability that a predetermined set (or sets) of functions will be supported over a predetermined time interval.

All of these measures are implicitly related to possible maintenance policies. If all ICNIA devices are to be repaired after each mission, then the time interval of interest is the mission duration, and the set of credible device failures should be selected accordingly. However, if it is desirable to allow missions to begin with failed devices, then the time interval of interest is the maximum maintenance interval, and the set of credible device failures is significantly larger. It is quite possible that a suboptimal algorithm which is superior over short periods will be inferior over longer periods. Algorithm effectiveness measures are discussed in detail in the following sections.

Table 4. Algorithm Effectiveness Measures

Measure	Description
Composite Utility	Provides a single figure of merit representing the average effectiveness of the algorithm over a predetermined time horizon.
Worst Case	Evaluates the largest deviation between the functions supported by the algorithm under study and an optimal algorithm over a predetermined set of credible device failures.
Probability of Success	Represents the probability that a predetermined set (or sets) of functions will be supported over a predetermined time interval. (This measure is closely related to MIREM performance measures.)

Composite Utility. The approach taken in this performance measure is to evaluate the expected utility attained using the resource allocation algorithm under study, according to Equation 2. This method averages over the probabilities of all device failures, and accounts for the relative priority of all sets of supported functions. Note that the composite utility is explicitly a function of the time between maintenance actions.

This performance measure has several advantages. It is derived directly from the resource allocation optimization problem so no additional heuristic inferences are required. It results in a single, comprehensive figure of merit so it is easy to compare the effectiveness of several different algorithms using this measure.

Unfortunately, the fact that there is only one figure of merit means that this performance measure does not permit a detailed analysis of the suboptimal behavior. No indication is given of what factors cause the algorithm to work poorly. Similarly, no indication is given of whether suboptimalities are consistent and small, or occasional and large. These factors may be significant in selecting which suboptimal algorithm should be implemented.

Worst Case. The Worst Case performance measure is simply the maximum difference, over a predefined set of device failures, between the functional set priority supported by the optimal static algorithm and the functional set priority supported by the algorithm under evaluation. This measure is also directly related to the optimal resource allocation problem, being defined from the integrand of Equation 2. In addition, this measure is deterministic -- that is, it does not depend on the statistics of device failures.

The advantages and disadvantages of the Worst Case measure are essentially the reverse of those of the Composite Utility measure. First, it is straightforward to use the Worst Case measure to analyze the conditions of system health that result in poor performance. Second, the Worst Case measure informs the designer directly about situations resulting in extremely poor performance.

On the other hand, the Worst Case algorithm has a number of disadvantages. First, an optimal static resource allocation algorithm is required for comparison (such as the algorithm presented in Section 3.3). In order to use the Worst Case measure, an optimal reference algorithm must be developed and run, which would involve substantial costs. Second, the Worst Case algorithm cannot be used to assess the average performance of the system; average performance depends on the statistics of device failure. Thus, the Worst Case measure can help in the design of an algorithm that avoids catastrophic allocation decisions, but it cannot determine whether the algorithm will perform well under average conditions.

Probability of Success. The Probability of Success performance measure is the probability that a predefined function set is supported over the predefined period. This can be calculated from Equation 2 by assigning zero priority to non-relevant function sets.

The Probability of Success measure allows direct comparison with MIREM. The MIREM measure is the probability of success over a predefined period, given the best possible algorithm. Thus, comparison with MIREM yields a direct measure of the effectiveness of the algorithm under test. If the Probability of Success is computed for several different sets of functions, a detailed analysis of algorithm suboptimalities is possible. Moreover, if the Probability of Success is computed for all sets of functions, direct calculation of the Composite Utility measure can be accomplished.

4.1.2 Computability Measures

These performance measures are designed to indicate whether a proposed allocation algorithm can be implemented in available computer resources without imposing an excessive burden. There are two categories of computability measures: memory requirements and processor load.

Memory is required both for program space and for data space. Program and data may require different types of memory (for example, read-only memory versus random-access memory) and should be calculated separately.

Similarly, processor load should be assessed both on the basis of peak load and average load. Peak load will probably occur when a system reconfiguration is required, whereas average load is relevant to allocation algorithms that perform background calculations (as, for example, precomputation of several likely possibilities for the next reconfiguration).

Peak processor load directly affects the reconfiguration delay. ICNIA specifications require configuration delay to be limited to a maximum of 10 seconds, including failure detection, resource allocation, and reinitialization.

Some algorithms may require more time for some combinations of system health state and function set priorities than for others. For this reason, allocation delay should be calculated for both the average case and for the maximum.

The evaluation of these factors is not a straightforward numerical comparison. Growth potential must also be considered. Sensitivity to such factors as added functions, changes in resource/function definitions, redefinition of functional set priorities must also be evaluated and weighed against the ability of the system to expand, if necessary, to accommodate the changes.

4.2 EVALUATION METHODS

A software tool is needed to evaluate measures of algorithm effectiveness for a wide range of suboptimal algorithms. Since many types of algorithms might be proposed, very few statistical assumptions can be used to simplify evaluation.

Any of the performance measures proposed in Section 4.1.1 can be evaluated using Monte Carlo methods. These methods involve actually implementing the algorithm and operating it over a statistically significant number of resource failure sequences. Thus, the evaluation software must include resources strings, statistical models of resource failure, and the true priorities attached to sets of functions, in addition to the algorithm under evaluation. Although there are no significant theoretical difficulties with this approach, software development is required, and evaluation run times could be substantial. Furthermore, true priorities have not yet been established by the Air Force.

Although it would be desirable to evaluate the Probability of Success measure in the same computational framework used by MIREM, this may not be possible due to computational shortcuts in the MIREM software. These shortcuts take advantage of the fact that in MIREM only the best possible allocation is of interest. Such shortcuts are clearly inapplicable in the context of evaluating the performance of allocation algorithms.

5. SUMMARY AND RECOMMENDATIONS

5.1 SUMMARY

The ICNIA system approach to improved reliability and decreased logistical support requirements employs an active redundancy concept which relies on a resource allocation process to respond to changing mission requirements and compensate for the loss of system components. This report establishes that the optimal solution to this problem requires a dynamic stochastic optimization. The optimal algorithm would consider timing and fault isolation as part of a global view of the ICNIA system. However, solutions to this overall optimization problem are very difficult to compute and do not appear to be feasible.

Under two simplifying assumptions, the dynamic stochastic optimization problem can be reduced to a sequence of static problems that can be solved readily. First, the total reconfiguration time must be small relative to the allowable downtime of any function; second, resource failure detection must be separated from the resource allocation algorithm. Any separate fault isolation process then results in an updated system health state and triggers a reconfiguration event.

Three performance measures which evaluate the effectiveness of the resource allocation algorithms have been derived from the optimization framework. Each of these performance measures explicitly depends on the time between maintenance actions, thus allowing the interaction between the resource allocation algorithm and maintenance policies to be considered. Software using Monte Carlo statistical methods can be developed to evaluate the performance of alternative allocation algorithms against these performance measures. Of these measures, the Probability of Success measure is the most versatile, and its results can be compared directly with MIREM in the algorithm and system design cycle.

In addition to algorithm performance, a system designer must consider the ability to implement the algorithm within the ICNIA system constraints. Implementation characteristics identified here include memory requirements, processor load, and total reconfiguration time. In addition, the margin of safety and the sensitivity of the implementation to changes in system constraints must be considered.

Three potential algorithm design approaches were evaluated against the basic criteria of effectiveness and computability. The first approach, including linear programming and associated techniques, was found to be inapplicable to the ICNIA problem. The second approach -- that of sequentially assigning functions on the basis of individual priorities -- was demonstrated to result in unnecessary functional degradation even for a very simple example. In addition, this approach is intrinsically incapable of allowing trade-offs between sets of functions, as in the "limp home" situation.

A third method, termed the Combinatorial Method, was proposed. This method would store in memory the optimal allocation for the resources currently available and for the next several possible sequences of resource failures. After each reallocation for resource failure, the algorithm would recompute the optimal allocation for the next set of resource failures as a background processing task. This approach, if feasible, will provide an optimal static resource allocation.

5.2 RECOMMENDATIONS

Currently, the sets of functions required to support each mission phase have been defined. To design or evaluate an allocation algorithm, a priority list by set of functions, rather than individual functions, must be established within each mission phase. This list will permit the algorithm to delineate what subset of these functions (or substitute functions) is the next most desirable, down to a set of "limp home" functions. Until this is done, algorithms cannot be practically designed or consistently compared in evaluating performance. It is recommended that the Air Force take steps to establish such function set priorities.

It is further recommended that:

1. A Monte Carlo-based model be developed for evaluating the performance of specific reconfiguration algorithms in terms of Probability of Success.

2. The feasibility of using a combinatorial reconfiguration algorithm in ICNIA be investigated, and its benefits be quantified and compared with any other proposed algorithms.

REFERENCES

Chizeck, H. (1981). Fault-tolerant optimal control (Doctoral dissertation, Massachusetts Institute of Technology).

Feldbaum, A. (1961) Theory of dual control (Parts I, II, III, IV). Telemkh., Automat., 21(9,11), 1960; 22(1,2).

Griffiths, B.E. (1983). Optimal control of jump-linear gaussian systems (Doctoral dissertation, Case Western Reserve University).

Kuester, J.L., & Mize, J.H. (1973). Optimization techniques with fortran. New York: McGraw-Hill.

Luenberger, D.G. (1973). Introduction to linear and nonlinear programming, Reading, MA: Addison-Wesley.

Rishel, R. (1978). The minimum principle, separation principle, and dynamic programming for partially observed jump processes. Journal of Mathematical Analysis and Applications, 65.

Veatch, M.H., Calvo, A.B., Myers, J.F., and McManus, J.C. (1985). Logistics engineering analysis techniques for fault-tolerant avionics systems (AFHRL-TR-84-60, AD-A161981). Wright-Patterson AFB, OH: Logistics and Human Factors Division, Air Force Human Resources Laboratory.

White, C.C. (1974). Finite-state, discrete-time optimization with randomly varying observation quality. Automatica, 10.

☆U.S. GOVERNMENT PRINTING OFFICE 1986—659-055/40019

END

DTIC

8–86