SYSTEM BALANCE FOR EXTENDED LOGISTIC SYSTEMS*

S. C. Graves
Massachusetts Institute of Technology
Sloan School of Management

and

J. Keilson
University of Rochester
Graduate School of Management

March 1981 WP No. 1253-81

* Work supported by Air Force Office of Scientific Research on grant No. AFOSR-79-0043, and by Air Force Business Management Research Center.

Introduction and Summary

An extended logistic system is a well-defined configuration of complex equipment, supporting inventory levels of components and modules, supporting maintenance facilities, supporting transportation system between local and remote inventory and maintenance sites, and procedures governing the allocation and shipment of components from remote and local sites. Examples of extended logistic systems are aircraft programs, radar systems, or networks of communication satellites. For each of these systems, the basic unit of interest (an aircraft, a radar unit, or a satellite) is a complex combination of components which are subject to failure. For each component there are supporting inventory and/or repair facilities, and specific replacement procedures for such failures. The evaluation of system performance includes system availability and the logistic costs required to obtain that level of availability.

This paper reports extensions to our earlier work [2] in which we develop methods for studying the dynamic behavior of extended logistic systems. In particular, an optimization model is proposed here for examining system design and tradeoff decisions. We consider a single-echelon model for a multiple-component system where components are subject to failure. For the system to be operable, a prespecified number of each component must be available. The optimization model maximizes system availability subject to a budget constraint on system cost and a dynamic time constraint on system failure time. The latter constraint is an effort to incorporate our understanding of system dynamics into the optimization by restricting the mean time between failures for the entire system. The use of the optimization model for system design and tradeoff decisions is illustrated by three examples using representative data from an Air Force program. Within the optimization context, the notion of system balance is defined. In essence, a balanced system is a system for which any change in investment between any two system levels satisfying system constraints, raises total system cost.

The paper is organized into four sections. In the first section we introduce the problem setting and define the problem of interest. In Section 2, we propose an optimization model for the stated problem which gives rise to an integer programming problem, and in turn to a linear programming approximation. Section 3 presents three numerical examples illustrating the potential utility of the model. Finally Section 4 compares the proposed approach to alternative approaches and suggests possible extensions.

1. Problem Definition

The general problem setting is the logistic system designed to support some operating system, typically a military operation; for instance, we might consider the logistic system that supports the operations for a squadron of aircraft. The logistic system includes the basic operating entities (e.g. aircraft), spare-parts inventories, repair equipment both on site and at a central repair depot, repair personnel, operating procedures, operating objectives, and system constraints. We are concerned with questions and tradeoffs involved in designing such a logistic support system. In this paper we examine a single-echelon model, in which only the echelon stocks spare-parts inventories and there is only one repair facility, either on site or at a central repair depot. We consider the determination of the appropriate inventory stockage levels for repairable modules, and the identification of possible economic benefits from improved repair capability and/or improved module reliability. We are not concerned here with detailed operating procedures or manpower scheduling.

For this problem setting we define three identities: system, entity, and module. The system is the operating system of concern, such as a squadron of aircraft. An entity is the basic operating unit in the operating system; thus an entity could be an aircraft. Hence we could view the system as a collection of entities: for the system to be operable, a prespecified number of the entities must be available. A module is a basic repairable unit for the entity. For an aircraft, a module could be an engine or a component of the engine, a navigational module, or a communication module. The entity (e.g. aircraft) is now viewed as a collection of modules; in order for the entity to be operable, a prespecified subset of the modules must be operable.

The problem of interest is to determine the specifications for the logistic system which maximizes the performance of the operating system at the least logistic cost. The system specifications of prime concern to us are the inventory stockage levels for the repairable modules. For logistic cost we focus upon the procurement and holding costs for the repairable modules and the maintenance costs associated with the repair of these modules. For system performance, we consider two measures. The first measure is system availability, which is defined as the steady-state probability that the operating system is operable. For instance, for a squadron of aircraft consisting of 50 aircraft we may say that the system is operable (i.e. conduct normal operations) at a given time if at least 80% of the aircraft (40) are available to fly. In this example the desired operational level is 40 aircraft or 80% of the squadron. The system availability is the probability of being at or above the desired operational level.

The second measure of system performance is system persistence as measured by the time until the next system failure. Here, the operating system is said to fail when the number of entities available to operate drops below a minimum threshold level. In the above example we might say that the system fails when less than 25 aircraft (50%) are available to fly. In such instances a state of emergency would exist, and extreme measures would be taken to bring the system back to a minimally-satisfactory level. Clearly, the longer the time is between such system failures, the longer is the time between "emergencies" and the better the system is performing.

In [2], four random variables describing in different ways the persistence of the system in the satisfactory region (e.g. more than 25 available aircraft), were examined. We will employ one of these persistence times (called in [2] the ergodic exit time), selected for its analytical tractability, its simplicity, and its utility for planning, to use as a measure of system performance. In [2], we saw that, under reasonable conditions, this persistence time is nearly exponentially distributed. Consequently, only the mean persistence time (or mean time between system failures) is needed, since an exponential random variable is fully characterized by one parameter, its mean. Technical details defining both the system availability and system persistence time are given in the next section.

In view of the above considerations, we pose the following optimization problem:

(P) Maximize System Availability

Subject to:

$$\text{Mean Time Between System Failures (MTBSF)} \geq T$$
$$\text{Total Logistic System Cost} \leq B$$

T and B are the preset MTBSF target and budget allocation, respectively. The primary decision variables in (P) are the stockage levels for the repairable items that define an entity. An alternative formulation to (P) might minimize "total logistic system cost" subject to constraints on "system availability" and on "mean time between system failures"; in Section 3 we illustrate both formulations with examples. Note that in (P) there is only one cost constraint. This constraint might represent the initial budget for spares procurement, or an annual budget allocation for maintaining the logistic system, or an additional, unexpected budget surplus to be used for an ongoing operating system to provide a "quick-fix". We will see that (P) may be easily extended to more than one cost constraint, if the single budget constraint is not sufficient.

In the next section we develop an explicit formulation for (P). Furthermore, we show that, for a set of assumptions, (P) may be modeled as a separable nonlinear program, which may be closely approximated by a linear program. This linear program has M+2 linear constraints, where M is the number of distinct modules needed to define an entity; M of these constraints are generalized upper bound constraints, which are handled implicitly by most linear programming codes. The number of columns in this linear program may be quite large; however we propose a column generation scheme which makes the solution procedure quite tractable.

An important model consideration which is key to the viability of the model, is the definition of an entity. Earlier, we stated that an entity is a collection or set of repairable modules. Most entities of interest (e.g. aircraft) consist of tens of thousands of distinct repairable modules; our model cannot handle that many modules. Furthermore, even if the model could handle all repairable modules, we conjecture that the output could easily be meaningless due to the model assumptions and required level of detail. Hence we redefine an entity as a _small_ collection of critical _modules_. By small, we have in mind 20 to 50 modules. By critical, we mean those modules for which there are significant economic tradeoffs. A critical module is one which is _costly_, _unreliable_, and _essential_ to the operation of the entity. A critical module is one which may cause an entity to fail or not be operable. In this light, a module is _not critical_ if it is cheap since we could stock sufficient quantities of this module to make a runout of this module quite rare. If a module is reliable, it is also not critical since it rarely fails and hence will not cause an entity to fail. Finally a module that is not essential for the operation of the entity is clearly not critical.

We conjecture that a small set of critical modules can be identified which will encompass the major tradeoffs and savings in the design of the logistic system. We would draw an analogy with the ABC analysis in inventory practice, in which stock keeping units are classified into three groups. Typically the items in the "A" class comprise 5 - 20% of all items, yet account for 70 - 90% of all sales. Consequently the most managerial attention is placed on the items in this class since that is where significant inventory savings may be achieved. The "B" and "C" classes receive considerably less attention; the control of these classes is highly automated and relatively conservative in order to not require the time and attention of management. The suggestion made here for logistic systems is to identify similarly a set of "A" modules to which major attention is paid. The non-critical modules ("B" and "C" items) need to be handled separately from the critical modules ("A" items). Furthermore the non-critical modules should be managed in a way such that they are inconspicuous and cause no "trouble"; that is, the non-critical modules should very rarely cause an entity to be nonoperable. Consequently, the attention of the system designers and the attention of our model are focused on the critical modules, those modules for which significant tradeoffs exist and which we allow to cause failures of an entity.

A valuable by-product of the linear programming model is the shadow price information from its dual solution. From the formulation given by (P), we obtain shadow prices (π_T, π_B) where π_T will be seen to be the percentage change in system availability from a marginal change in T, and similarly for π_B. From this information, we can determine for each module the maximum value we would pay for a perfectly reliable version of that module (i.e. never fails). In addition we can use this information to aid in categorizing modules as critical versus non-critical. Section 3 illustrates these uses of the model.

2. Model Development

The key assumptions necessary for the model development are as follows:

(a) There is complete cannibalism of the inventories of the repairable modules. With this assumption, the requirement that k entities be operable (e.g. k aircraft ready to fly) is equivalent to requiring k available units of each repairable module, assuming that each entity requires exactly one of each repairable module.

(b) System failures are rare, i.e. sufficient resources are allocated to the logistic system to make the operating system operate with reasonable reliability. This assumption is necessary to ensure the appropriateness of the exponential approximation for the system MTBSF (see [2]).

(c) The failure and repair of distinct types of modules are independent. Consequently, the random variables $\{\tilde{I}_i(t)\}$, each of which denotes the available inventory at time t for repairable module i for $i=1,2,\dots,M$, are independently distributed.

(d) Each random variable $\tilde{I}_i(t)$ is modelable as a birth-death process on the state space $\{0,1,\dots,N_i\}$, where N_i is the number of units stocked for module i. We define the vectors $\underline{\lambda}_i = \{\lambda_{ij}\}$ and $\underline{\mu}_i = \{\mu_{ij}\}$ to denote the transition rates for this birth-death process. That is, λ_{ij} for $j=0,1,\dots,N_i-1$ is the upward trasition rate from state j to j+1 while μ_{ij} for $j=1,2,\dots,N_i$ is the downward transition rate from state j to j-1. The optimization requires no explicit functional form for the transition rates for the birth-death processes.

The above four assumptions do not seem to be very restrictive, but certainly need to be examined. Assumption (a), which assumes complete cannibalism, is never valid; however, neither is the assumption of no cannibalism. Partial cannibalism does go on, and is very difficult, if not impossible, to model. We choose full cannibalism over no cannibalism primarily for analytic convenience. Assumption (b) merely states that we want a reliable system, which is certainly valid for most important military and commercial systems. Assumptions (c) and (d) are standard assumptions for reliability models, and are made not only for their analytic convenience but for the robustness demonstrated by these models. Furthermore, we note that the optimization permits more complex stochastic models for the inventories $\tilde{I}_i(t)$. We could, for instance, model such an inventory level with a two-dimensional birth-death process. This extension would permit a wider class of repair time distributions, such as are encountered when module repair may take place at more than one location (e.g. base and depot repair); we discuss this extension in greater detail in Section 4.

We will be interested in the random process

$$\tilde{N}(t) = \min\{\tilde{I}_1(t), \tilde{I}_2(t), \ldots, \tilde{I}_M(t)\} . \tag{1}$$

Since we assume that each entity requires one unit of each module to be operable, $\tilde{N}(t)$ is the number of operable entities at time t. We define <u>system availability</u> by

$$A_S = \lim_{t\to\infty} \Pr[\tilde{N}(t) \geq k_1] \tag{2}$$

where k_1 represents the desired operational level for the system. Since the module inventories are independent we may write the system availability as

$$\begin{aligned} A_S &= \lim_{t\to\infty} \prod_{i=1}^{M} \Pr[\tilde{I}_i(t) \geq k_1] \\ &= \prod_{i=1}^{M} A_i \end{aligned} \tag{3}$$

where A_i is the steady-state availability of module i. We note that the module availability is a function of the parameters $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ which define the birth-death process that governs $\tilde{I}_i(t)$, and is easily computed from the ergodic analysis of the birth-death process.

We characterize system persistence by the random variable T_S which denotes the ergodic system failure time [2]. The ergodic system failure time is the time until the next system "failure" conditioned on the sole observation that the system is now "working". The system "fails" when less than k_2 entities are operable; otherwise the system is in an acceptable or "working" state. The survival function for the ergodic system failure time is given by

$$\Pr[T_S > \tau] = \lim_{t_1 \to \infty} \Pr[\tilde{N}(t) \geq k_2, t_1 < t \leq t_1 + \tau \mid \tilde{N}(t_1) \geq k_2] \tag{4}$$

where k_2 represents the minimum threshold level for the system.

From the independence of the module inventories, we can rewrite the survival function for the ergodic system failure time as

$$\Pr[T_S > \tau] = \prod_{i=1}^{M} \Pr[T_{Si} > \tau] \tag{5}$$

where T_{Si} is the random variable for the ergodic failure time of module i. In [2] we found that the distribution of T_{Si} is very nearly exponential for reliable modules. Hence we can approximate the survival function of T_{Si} by

$$\Pr[T_{Si} > \tau] \cong \exp(-\nu_i \tau) \tag{6}$$

where $E\{T_{Si}\} = 1/\nu_i$. We note that ν_i is a function of the parameters $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ for module i. In the appendix we show how to compute ν_i. Substituting (6) into (5) we have

$$\Pr[T_S > \tau] \cong \exp(-\tau \sum_{i=1}^{M} \nu_i). \tag{7}$$

Using this approximation for the distribution of T_S, we may focus on the expected system failure time, namely

$$E\{T_S\} \cong (\sum_{i=1}^{M} \nu_i)^{-1} . \tag{8}$$

Finally we model the annual system cost C_S as the sum of the costs for the individual modules; that is

$$C_S = \sum_{i=1}^{M} C_i \tag{9}$$

where C_i, the annual cost of module i, is a function of the parameters $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ and should include procurements costs, maintenance and repair costs and inventory holding costs.

We now state problem (P) using (3), (8), and (9) as

$$\text{(P1)} \quad \text{Max } A_S = \prod_{i=1}^{M} A_i \tag{10}$$

Subject to:

$$\left(\sum_{i=1}^{M} \nu_i\right)^{-1} \geq T \tag{11}$$

$$\sum_{i=1}^{M} C_i \leq B \tag{12}$$

$$(\underline{\lambda}_i, \underline{\mu}_i, N_i) \in X_i \qquad i=1,2,\ldots,M \tag{13}$$

The set X_i defines the optional* choices for the parameters $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ for each module i. Again we note that A_i, ν_i, and C_i are each functions of $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$. The decision problem posed in (P1) is to choose the optimal parameter set $(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ for each module i. We assume each set X_i is finite with cardinality R_i; hence we may write

$$X_i = \{(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r);\ r=1,2,\ldots,R_i\} \tag{14}$$

where $(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r)$ is the rth element in set X_i.

By taking the logarithm of (10) and inverting (11), we can transform

* "Optional" is here used in the sense of the system planner's choices and is unrelated to mathematical feasibility.

(P1) into a separable, nonlinear program (P2):

(P2) $$\text{Max} \quad \log A_S = \sum_{i=1}^{M} \log A_i \tag{15}$$

Subject to:

$$\sum_{i=1}^{M} \nu_i \leq 1/T \tag{16}$$

$$\sum_{i=1}^{M} C_i \leq B \tag{17}$$

$$(\underline{\lambda}_i, \underline{\mu}_i, N_i) \in X_i \qquad i=1,2,\ldots,M \tag{18}$$

Now, by defining for each module i

$$a_{ir} = \log A_i(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r) \tag{19}$$

$$\nu_{ir} = \nu_i(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r), \tag{20}$$

and $$c_{ir} = C_i(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r) \qquad \text{for } r=1,2,\ldots,R_i, \tag{21}$$

we can rewrite (P2) as a zero-one integer program (P3):

(P3) $$\text{Max} \quad \log A_S = \sum_{i=1}^{M} \sum_{r=1}^{R_i} a_{ir} x_{ir} \tag{22}$$

Subject to:

$$\sum_{i=1}^{M} \sum_{r=1}^{R_i} \nu_{ir} x_{ir} \leq 1/T \tag{23}$$

$$\sum_{i=1}^{M} \sum_{r=1}^{R_i} c_{ir} x_{ir} \leq B \tag{24}$$

$$\sum_{r=1}^{R_i} x_{ir} = 1 \qquad i=1,2,\ldots,M \tag{25}$$

$$x_{ir} = 0,1 \qquad \begin{array}{l} i=1,2,\ldots,M \\ r=1,2,\ldots,R_i \end{array} \tag{26}$$

In (P3) the decision variables are the zero-one variables $\{x_{ir}\}$, where x_{ir} equal to one denotes the choice of the r th parameter set, i.e. $(\underline{\lambda}_i^r, \underline{\mu}_i^r, N_i^r)$, for module i. Constraints (23) and (24) correspond directly to constraints (16) and (17) in (P2), while the constraint set (25) ensures that exactly one parameter set is chosen for each module.

Problem (P3) is an integer program with a very large number of zero-one variables. Rather than attempt an optimal solution to (P3), we propose a heuristic solution procedure in which we solve the linear programming relaxation to (P3), which we call problem $(\overline{P3})$. That is, we solve (P3) but with (26) replaced by

$$0 \leq x_{ir} \leq 1 \qquad \begin{array}{l} i=1,2,\ldots,M \\ r=1,2,\ldots,R_i. \end{array} \qquad (27)$$

If the linear programming solution is all integer, we have the optimal solution to (P3). Otherwise, we have a fractional solution which we round to get a solution to (P3). We first note that the optimal solution to the linear program is always nearly integer with <u>at</u> <u>most</u> two modules having fractional solutions and with <u>at</u> <u>most</u> four fractional decision variables, <u>independent</u> of the value of M. This is a consequence of the fact that a basic feasible solution to the linear program $(\overline{P3})$ has at most M+2 positive decision variables and that $(\overline{P3})$ has M generalized upper bound constraints. Second, as a result of the first observation, there are at most four possible integer solutions suggested by the linear program solution. Unfortunately, none of these integer solutions need be feasible in (P3) in that either constraints (23) or (24) may be violated. However, we expect the integer solutions to at least be "nearly feasible", and since constraints (23) and (24) are typically "soft" constraints, we expect that the rounded linear program solutions to be acceptable solutions to the original problem.

The linear program $(\overline{P3})$ may still be a very difficult problem to solve in that it could conceivably have millions of decision variables and thousands of constraints. We manage the enormity of this problem in two ways. First, we restrict the value of M (i.e. number of constraints) by only considering critical modules for the system, as discussed earlier. Second, we solve the linear program by means of a column generation procedure [7] so that we need not explicitly compute all of the decision variables (columns) of the linear program. This solution procedure is an iterative procedure in which we iterate between solving a master problem and solving a set of subproblems, one for each module i. We use the subproblems to generate the columns of $(\overline{P3})$ as needed. The master problem is a partial representation of $(\overline{P3})$, in which we have present only the columns of $(\overline{P3})$ that have been previously generated; at each iteration it is solved as a linear program with M generalized upper bound constraints. The subproblem for each module i is a discrete optimization problem with the general form

$$(SP_i) \quad \text{Max} \quad \{\hat{a}_i - \pi_T\hat{\nu}_i - \pi_B\hat{c}_i - \delta_i\} \tag{28}$$

Subject to:

$$\hat{a}_i = \log A_i(\underline{\lambda}, \underline{\mu}, N) \tag{29}$$

$$\hat{\nu}_i = \nu_i(\underline{\lambda}, \underline{\mu}, N) \tag{30}$$

$$\hat{c}_i = C_i(\underline{\lambda}, \underline{\mu}, N) \tag{31}$$

$$(\underline{\lambda}, \underline{\mu}, N) \in X_i \tag{32}$$

where the vector $\underline{\delta} = (\delta_1, \delta_2, \ldots, \delta_M)$, and where $(\pi_T, \pi_B, \underline{\delta})$ are the optimal shadow prices from the most recent solution to the master problem. The column generation procedure terminates once the optimal objective values for all of the subproblems are nonpositive; the procedure is guaranteed to terminate within a finite number of iterations [7].

The computational effectiveness of the column generation procedure depends on the ease with which we can solve the subproblems (SP_i). For general candidate sets X_i, the solution of (SP_i) may be quite difficult; indeed, conceivably we would have to evaluate explicitly each element in the set at each iteration. However, if the set X_i has some inherent structure, the solution of (SP_i) may be reasonably simple. Such a case occurs when there is an underlying class of birth-death models for each module. In [2] we examine four possible classes of birth-death processes for modeling a module's inventory level. As an example, suppose for each module i there is a basic repair rate and failure rate, say $\lambda(i)$ and $\mu(i)$, such that the birth-death process governing the inventory for module i, has the transition rates

$$\lambda_{ij} = (N_i - j) \cdot \lambda(i) \tag{33}$$

$$\mu_{ij} = j \cdot \mu(i) \tag{34}$$

where N_i is the stockage level for module i. That is, the transition rates for repair and failure are directly proportional to the number of failed units and the number of operable units, respectively. The set X_i is now fully specified by (33) - (34), and by the feasible range of values for N_i. The solution of (SP_i) is then a line search over the possible values for N_i. Unfortunately, the objective function (28) for such a class of birth-death processes need not be unimodal in N_i. However, our experience suggests that for realistic birth-death classes, such as examined in [2] and in the next section, this function* is unimodal over the values for N_i of interest. Consequently, we treat (28) as if it were unimodal and hence solve the subproblem (SP_i) for a given birth-death class by a standard line-search technique.

* In these instances we have always assumed that the cost function, $C_i(\underline{\lambda}, \underline{\mu}, N)$, is linear in N.

A more general instance in which the subproblem (SP_i) is readily solvable occurs when there are a limited number of types for module i. For example, we may have to choose for module i between an inexpensive, unreliable type and a more expensive, but more reliable type. A module type would be characterized by its basic repair rate and failure rate, by a functional form, such as (33) - (34), for defining the transition rates for its birth-death process, and by its cost function. The solution of (SP_i) now entails the choice of module type, as well as the number of units to stock. As above, for each type for module i, a line search over the values of N_i will determine the number of units to stock. The solution to (SP_i) is then given by choosing that module type which maximizes the objective function (28).

In the optimal solution to ($\overline{P3}$) we interpret the optimal dual value π_B as the percentage change in system availability from a unit change in B. To see this, note that for $A_S(B)$ being the system availability as a function of B, we may write for small $\varepsilon > 0$

$$\log A_S(B+E) = \log A_S(B) + \pi_B \varepsilon \quad . \tag{35}$$

After rearrangement, we obtain

$$\frac{A_S(B+\varepsilon) - A_S(B)}{A_S(B)} = e^{\pi_B \cdot \varepsilon} - 1$$

$$= \pi_B \cdot \varepsilon + o(\varepsilon) \quad . \tag{36}$$

A similar interpretation can be given for π_T.

The optimal dual values are also useful for determining the maximum amount to pay for a hypothetical perfectly reliable module. For a module that never fails, we would stock exactly $N_i = k_1$ units giving an availability A_i of 1 so that $\hat{a}_i = \log A_i = 0$, and a system failure rate $\hat{\nu}_i$ of zero. We would opt for the perfectly reliable item only if its reduced cost, as determined by the objective function in (SP_i), is positive; that is, we require

$$- \pi_B c_i^* - \delta_i > 0 \qquad (37)$$

where π_B, δ_i are optimal dual values from $(\overline{P3})$ and c_i^* is the total cost associated with stocking the perfectly reliable module. If the cost function c_i^* is linear in the number of units stocked, then from (37) we find that the maximum value we would pay for a perfectly reliable unit is $-\delta_i/k_i\pi_B$.

3. Numerical Examples

Three small examples using Air Force data are examined to illustrate the potential utility of the proposed approach. For the aircraft program under study, we first selected, with the aid of the program managers, nine critical modules. We then designated the aircraft (i.e. entity) to be the collection of these nine modules, with the operating system being a collection of these aircraft. We should reemphasize that this numerical study is purely illustrative; we make no claim that these nine modules are the most appropriate representation of the aircraft. Indeed we would expect an actual application of the model to include these modules in addition to up to another fifty critical modules.

Table 1 gives for each module i the unit cost and both the basic repair rate $\lambda(i)$ and the basic failure rate $\mu(i)$. For this example we assume that the cost function $C_i(\underline{\lambda}_i, \underline{\mu}_i, N_i)$ is a linear function of N_i, with the unit cost being the slope of this function. The failure rate assumes that each aircraft flies a fixed number of hours per day. As mentioned in Section 3, the inventory process $\{\tilde{I}_i(t)\}$ for each module is modeled as a finite birth-death process. If N_i is the number of units stocked for module i and if k_1 is the desired operating level for the system, then the transition rates for the birth-death process for module i are

$$\lambda_{ij} = (N_i - j)\cdot\lambda(i)$$

$$\mu_{ij} = \min(k_1, j)\cdot\mu(i)$$

where λ_{ij} is the repair rate and μ_{ij} is the failure rate when $\tilde{I}_i = j$. Here the repair rate depends directly upon the number of units failed, (N_i-j); this corresponds to assuming an infinite number of servers at the repair depot. The failure rate depends upon how many aircraft are flying which is the smaller of the number of aircraft available to fly (j) and the desired number of aircraft (k_1). Note that although we use this specification for the birth-death model, we could have used any general class of the birth-death process.

We have programmed our approach in FORTRAN on an interactive PRIME 400 minicomputer at M.I.T. All of the examples were solved within seconds in real time.

Example 1

For the first example we set the desired operational level (k_1) to be 25 aircraft and we drop the constraint on system MTBSF. Hence, the optimization model is to maximize system availability subject to a constraint on the total budget; our intent is to focus on the interaction between the budget and system availability. Table 2 reports the optimal system availability achieved over a range of budget values. This table also reports π_B, the shadow price on the budget constraint; for instance, for B = 4500 a shadow price of .0008 reflects the percentage change in system availability from an additional unit of the budget (i.e. B = 4501). Table 3 gives the detailed solution when B = 4500. Note how the number of units stocked for each module varies; if an equal number of units of each module were stocked, then N_i would equal 31 for each module and system availability would be only 70%, as opposed to the optimal 88%. Table 3 also gives for each module the value of a perfectly reliable unit; this value is the maximum amount we would be willing to pay per unit for a module which never fails. A final bit of analysis that we can do with this simple example is to ask whether we have properly defined our set of critical modules. For instance, suppose there is a module #10 which we initially classified as non-critical. Suppose this module has a per unit cost of 1.0 and that to ensure that this module never causes the system to drop below

k_1* we have set its stockage level N_{10} to be 50. Now we suggest a marginal analysis to examine whether our initial classification was appropriate. Suppose we reduce N_{10} from 50 to 49, with the consequence that $\Pr[I_{10} \geq k_1]$ drops from 1.0 to .995. Hence there is a percentage drop in system availability of about 0.5%, since system availability is given by $\prod_i \Pr[I_i \geq k_1]$. But reducing N_{10} by one frees up an amount 1.0 which can be applied to the budget for the critical modules. However, if B = 4500, from Table 2 we see that this additional 1.0 in the budget gives at most a 0.08% increase (π_B) in system availability. Since 0.5% > 0.08% we should keep N_{10} at 50 and module 10 is properly classified as non-critical. In general we can use the shadow price π_B to determine if non-critical modules should be reclassified as critical. This procedure for classifying modules can be easily extended to incorporate the MTBSF constraint.

Example 2

For this example we focus on the interaction between the two measures for system performance: system availability and system MTBSF. To do this, we restate (P) as

$(\hat{P})$ Min {Total Logistic System Cost}

Subject to:

$$\text{System Availability} \geq A$$

$$\text{Mean Time Between System Failures} \geq T$$

where A is the minimally-acceptable availability level and T is the minimally-acceptable system MTBSF. The solution procedure for $(\hat{P})$ is nearly identical

* Ideally, this implies that $\Pr[I_{10} \geq k_1] = 1.0$. Practically, however, we need an operational definition such as $\Pr[I_{10} \geq k_2] = .999$.

to that of (P). For this example we assume that the minimum threshold level (k_2) with respect to which system failures are recorded, and the desired operational level (k_1) are both equal to 25 aircraft. We then solve ($\hat{P}$) for varying sets of (A,T) to examine the solution's sensitivity to these constraints. Table 4 reports these results. Clearly, the two system measures are closely correlated in this example. If we increase the constraint for one, we naturally get an increase in the other measure. However, as seen in Table 4 neither constraint dominates the other, and it is possible to have solutions with both constraints binding. We would expect that this interaction would be more pronounced if $k_1 \neq k_2$, although we have not examined this case in detail.

Example 3

This example examines the utility of the optimization model for considering improved types of a particular module. As a base case we take the solution to ($\hat{P}$) from Table 4 with A = .90 and T = 50; Table 5 gives the detailed stockage levels for this solution. First we consider the impact of improved types of module 1. The original module 1 has per unit cost 40.0, a basic repair rate $\lambda(i) = 0.16$ and a basic failure rate $\mu(i) = 0.16$. We consider substitutes for item 1 with lower failure rates, but possibly higher per unit costs. Using these substitutes we recompute the optimal system configuration and cost, and report these results in Table 6. This exercise is repeated for module 9 and is reported in Table 7. From these tables, we get a feel for the cost of a module's unreliability. For module 1 we see that we have to cut the failure rate in half before we are even willing to pay 5% more (42.0) for each unit; for module 9 we would pay a premium of slightly over 10% to cut the failure rate in half. Overall, for these two modules, there is not a great savings from having more reliable units, assuming that these new units would be at least as expensive as the current versions.

TABLE 1: SPECIFICATIONS FOR MODULES

Module	$\lambda(i)$ (per day)	$\mu(i)$ (per 100 flying hours)*	Cost (per unit) (-000)
1	0.16	0.16	40.07
2	0.27	0.11	1.97
3	0.10	0.22	41.60
4	0.05	0.01	1.85
5	0.05	0.12	4.06
6	0.10	0.24	6.39
7	0.11	0.23	5.63
8	0.04	0.15	29.96
9	0.06	0.25	13.55
Total Cost for all modules			144.0

TABLE 2: OPTIMAL SYSTEM AVAILABILITY FOR RANGE OF BUDGET VALUES

Budget (B) (-000)	System Availability	Shadow Price
4000	0.071	.0152
4100	0.203	.0081
4200	0.415	.0050
4300	0.618	.0026
4400	0.777	.0019
4500	0.881	.0008
4600	0.945	.0008
4700	0.974	.0002
4800	0.990	.0001
4900	0.996	<.0001
5000	0.999	<.0001

* We assume five flying hours per day per aircraft.

TABLE 3: DETAILED SOLUTION FOR B = 4500

Module	N_i	Unit Cost	Value for Perfect Unit
1	28	40.07	46.8
2	29	1.97	2.3
3	30	41.60	52.2
4	28	1.85	2.1
5	34	4.06	5.6
6	33	6.39	8.6
7	32	5.63	7.4
8	33	29.96	40.6
9	36	13.55	19.8

TABLE 4: OPTIMAL SOLUTION TO $(\hat{P})$ FOR RANGE OF (A,T) VALUES

A	T (days)	System Cost *	System Availability**	System MTBSF**
.80	20	4429	-	22
.80	33	4483	.85	-
.85	33	4486	-	-
.85	50	4547	.90	-
.90	33	4533	-	40
.90	50	4548	-	-
.90	100	4663	.95	-
.95	50	4611	-	67
.95	100	4704	-	108

* This column reports optimal system cost found from $(\hat{P})$ assuming right-hand-side values (A,T).

** These columns report actual system availability and system MTBF achieved in the solution to $(\hat{P})$ for given values of (A,T). A "dash" indicates that the respective constraint was binding in the optimal solution.

TABLE 5: DETAILED SOLUTION TO $(\hat{P})$ WITH A = .90, T = 50.

Module	N_i
1	29
2	29
3	31
4	28
5	33
6	34
7	33
8	32
9	36

System Cost = 4548

TABLE 6: IMPACT OF ALTERNATE VERSIONS FOR MODULE 1

Base Case (N_i = 29)	$\lambda(i)$ (per day)	$\mu(i)$ (per 100 flying hours)	Item Cost	System Cost
	0.16	0.16	40.0	4548
	0.16	0.12	40.0	4523
	0.16	0.12	42.0	4580
	0.16	0.12	45.0	4665
	0.16	0.08	40.0	4494
	0.16	0.08	42.0	4549
	0.16	0.08	45.0	4632
	0.16	0.04	40.0	4454
	0.16	0.04	42.0	4508
	0.16	0.04	45.0	4589

TABLE 7: IMPACT OF ALTERNATE VERSIONS FOR MODULE 9

Base Case (N_i = 36)	$\lambda(i)$ (per day)	$\mu(i)$ (per 100 flying hours)	Item Cost	System Cost
	0.06	0.24	13.6	4548
	0.06	0.18	13.6	4519
	0.06	0.18	14.0	4535
	0.06	0.18	15.0	4569
	0.06	0.12	13.6	4490
	0.06	0.12	14.0	4505
	0.06	0.12	15.0	4536
	0.06	0.12	16.0	4567
	0.06	0.06	13.6	4455
	0.06	0.06	15.0	4497
	0.06	0.06	17.0	4555

4. Discussion and Extensions

In this paper we have proposed and demonstrated an optimization model for aiding in the evaluation of logistic system design decisions. We should first note the systems emphasis of this model, in that the optimization is concerned with system performance, with system objectives and with system constraints. Second, we note the tractability of the procedure in that the optimization requires the iterative solution of a series of small linear programs; indeed, all of the examples reported in Section 3 were done on an interactive PRIME 400 minicomputer. Finally, we mention the uses of the proposed model. As demonstrated with the examples, we can use the model not only for the original specification of a logistic system, but also for the evaluation both of changes in the operations of an ongoing system and of improved versions of individual modules. Furthermore, the model gives information both to aid in the classification of modules as critical versus non-critical, and to assess the impact of a module's unreliability.

We need to compare and contrast the approach presented here with alternative approaches to designing logistic systems. In particular, we consider the METRIC model proposed by Sherbrooke [8] and extended by Muckstadt [5]. We focus on this approach, since to various degrees, the Air Force has implemented METRIC. Demmy and Presutti [1] review the use of METRIC-like systems by the Air Force for determining spares levels for repairable items. Nahmias [6] has written a very nice survey of the literature for managing repairable item inventory systems, and gives both a nice overview to METRIC and a thorough review of other work in logistic systems.

The basic METRIC model considers a two-echelon system in which independent bases (lower echelon) are supported by a repair depot (upper echelon). This approach consists of solving the following problem:

(M) Min {Total Expected Backorders}

Subject to:

{Total Logistic System Cost} $\leq$ B

where a backorder occurs whenever a module's inventory drops below some specified level. By applying a Lagrange multiplier to the single budget constraint, the objective function separates into an unconstrained minimization for each repairable module. The procedure determines both the desired stockage levels at the independent bases and the stockage level at the repair depot for each module. The optimization procedure originally proposed by Sherbrooke is a marginal allocation scheme.

In comparing METRIC with the approach presented in this paper, METRIC seems to have two distinct advantages. First, METRIC explicitly models two types of repair (base and depot) with two corresponding inventories. Our approach, as illustrated here, has only one type of repair; however, as we will discuss, the framework is extendable to more complex representations of repair which we are currently exploring. Second, METRIC needs no assumptions about the repair time distributions other than stationarity and a finite mean. In our approach, we seem to be restricted to exponential distributions, and convolutions of exponentials; we intend to examine the consequences of this restriction when we model more complex repair time distributions.

There are three possible limitations to METRIC, which the current approach overcomes. First, METRIC has no system focus, other than through the budget constraint. In particular there is no accounting of system performance according to any measure. Total expected backorders are used as a surrogate for system performance; it is not at all clear how good a surrogate this is. Second, expected backorders for a module is a static measure, like availability, and does not reflect the dynamics of the system. Our approach incorporates a system MTBSF into the optimization model as a dynamic measure for the system. Third, METRIC need assume there are an infinite number of servers or infinite capacity at the repair depot; no such assumption is required here.

There are two additional aspects which distinguish our approach from the METRIC approach. First, there seems to be a difference in modeling philosophy. Our approach requires the identification of a small set of critical modules as representative of the basic entity. We then optimize the stockage levels for these critical modules, subject to system constraints and system criteria. Non-critical modules are treated separately and are assumed to be sufficiently stocked so that they rarely cause a system failure. On the other hand, the METRIC approach is set up to handle identically all modules, critical or non-critical, in a monolithic optimization. Clearly, though, METRIC could be redefined to identify distinct classes of modules, and treat them separately. The second aspect concerns the computational effort required by each approach. It is not clear how to make a meaningful comparison of the required computational effort since the two approaches solve different problems. However, we note that the basic component of our approach is the solution of a well-structured linear program, whereas the METRIC approach ultimately requires a multidimensional search over a nonlinear surface.

An important extension to the METRIC model is the LMI procurement model [4]. This model makes the same assumptions as the METRIC model but with a different criterion: the LMI model maximizes the expected number of operable aircraft, instead of minimizing expected backorders. Hence, the LMI model is similar to our approach in that it works with a system criterion. One difference, though, is that the LMI model assumes no cannibalism, whereas we have assumed complete cannibalism. Since the LMI model is based on the METRIC framework, it has the same advantages and disadvantages as METRIC when compared with our approach.

The work presented in this paper is by no means complete. As we have mentioned, there are several areas in need of more investigation. In particular, we need to better understand the impact of system MTBSF, as related to system availability and module specifications, within the optimization. We also need a better understanding, both theoretically and operationally, of how to categorize modules. Finally, we intend to extend the current model to more realistic representations of the repair process; this will give not only a broader class of models, but also the capability to explore the robustness of the simple birth-death model relative to the more complex representations. In particular, we note that the approach is extendable to Markovian models for each module's inventory that are more complex than the birth-death model. For instance, we might model the repair time distribution as an Erlang random variable. The only change in the solution procedure would occur with the solution of the subproblems (SP_i). The subproblems could still be solved by a line search over the values for N_i, but now the evaluation of the functions for a_i and ν_i is more complex. Similarly we could allow for two types of exponential repair, say a "fast" repair at the base and a "slow" repair at a distant depot, depending upon the severity of a module failure. In this instance, we would model the module's inventory as a two-dimensional birth-death process. Again, the basic solution procedure is valid, but the solution of the subproblems will be more involved.

References

1. W. S. Demmy and V. J. Presutti, "Multi-Echelon Inventory Theory in the Air Force Logistics Command", Working Paper, 1979.

2. S. C. Graves and J. Keilson, "A Methodology for Studying the Dynamics of Extended Logistic Systems", Naval Research Logistics Quarterly, Vol. 26, No. 2, June 1979.

3. J. Keilson, "A Review of Transient Behavior in Regular Diffusion and Birth-Death Processes: Part II", Journal of Applied Probability, Vol. 2, 1965, pp. 405-428.

4. LMI Task 72-3, Measurements of Military Essentiality, Logistics Management Institute, 4701 Sangamore Road, Washington, D.C. 20016, 78 pages.

5. J. A. Muckstadt, "A Model for a Multi-Item, Multi-Echelon, Multi-Indenture Inventory System", Management Science, Vol. 20, 1973, pp. 472-481.

6. S. Nahmias, "Managing Repairable Item Inventory Systems: A Review", Working Paper, 1979.

7. J. F. Shapiro, Mathematical Programming: Structures and Algorithms, John Wiley & Sons, New York, 1979, pp. 200-215.

8. C. C. Sherbrooke, "METRIC: A Multi-Echelon Technique for Recoverable Item Control", Operations Research, Vol. 16, 1968, pp. 122-141.

Appendix: Computation of the Expected Ergodic Failure Time

Consider a birth-death process $\tilde{I}(t)$ defined on the state space $S = \{0,1,\ldots,N\}$ with transition rates λ_j, $j=0,1,\ldots,N-1$, for going from state j to state j+1, and with transition rates μ_j, $j=1,2,\ldots,N$, for going from state j to state j-1. Suppose we split the state space S into a good set $G = \{k,k+1,\ldots,N\}$ and a bad set $B = S-G = \{0,1,\ldots,k-1\}$, such that the system is in a working state only if the system is in set G; otherwise the system is in set B, and the system has failed. Suppose that all we know about the system is that it is now working, and has been working for some time; then we define the ergodic failure time as the first passage time from set G to set B. In this appendix, we indicate how the mean ergodic failure time is computed; this derivation was originally given in [3].

Define T_{ij} to be the first passage time from state i to state j and let T_S be the ergodic failure time for the system. If we denote the expected passage times by $\overline{T}_{ij}$ and $\overline{T}_S$, then we have that

$$\overline{T}_S = (\sum_{j=k}^{N} e_j)^{-1} \cdot \sum_{j=k}^{N} e_j \overline{T}_{j,k-1} \tag{A1}$$

where e_j is the ergodic probability for being in state j. For $j \geq k$, the expected passage time $\overline{T}_{j,k-1}$ can be expressed as

$$\overline{T}_{j,k-1} = \overline{T}_{j,j-1} + \overline{T}_{j-1,j-2} + \cdots \overline{T}_{k,k-1} \tag{A2}$$

By substituting (A2) into (A1) and simplifying, we obtain

$$\overline{T}_S = (\sum_{j=k}^{N} e_j)^{-1} \cdot (\sum_{i=k}^{N} \overline{T}_{i,k-1} \sum_{j=i}^{N} e_j) \tag{A3}$$

By defining $\overline{E}_i = \sum_{j=i}^{N} e_j$, we rewrite (A3) as

$$\overline{T}_S = (\overline{E}_k)^{-1} \cdot \sum_{i=k}^{N} \overline{T}_{i,i-1} \overline{E}_i \tag{A4}$$

To use (A4) we need be able to compute both $\overline{E}_i$ and $\overline{T}_{i,i-1}$ for i=k,...,N. The computation of $\overline{E}_i$ is immediate from the standard computation of the ergodic probabilities for a birth-death process. In the following we show how also to compute $\overline{T}_{i,i-1}$ from the ergodic probabilities.

Define $\sigma_j(\ )$ to be the Laplace transform for the probability density function for $T_{j,j-1}$. It is easy to show for s > 0 that

$$\sigma_j(s) = \frac{\mu_j}{s+\lambda_j+\mu_j} + \frac{\lambda_j}{s+\lambda_j+\mu_j}\sigma_j(s)\sigma_{j+1}(s) \tag{A5}$$

and hence

$$\sigma_j(s) = \frac{\mu_j}{s+\lambda_j+\mu_j-\lambda_j\sigma_{j+1}(s)} \quad . \tag{A6}$$

From the fact that the expected passage time $\overline{T}_{j,j-1} = -\frac{d}{ds}\sigma_j(s)\Big|_{s=0}$ we find that

$$\overline{T}_{j,j-1} = \frac{1}{\mu_j}(1+\lambda_j\overline{T}_{j+1,j}) \tag{A7}$$

where $\overline{T}_{N+1,N} \equiv 0$. By multiplying each side of (A7) by e_j, and using the fact that $\lambda_j e_j = \mu_{j+1}e_{j+1}$, we find

$$\mu_j e_j \overline{T}_{j,j-1} - \mu_{j+1}e_{j+1}\overline{T}_{j+1,j} = e_j \quad . \tag{A8}$$

By summing (A8) for j=i,i+1,...,N, we obtain

$$\mu_i e_i \overline{T}_{i,i-1} = \overline{E}_i \quad . \tag{A9}$$

Now if we substitute (A9) into (A4) we have

$$\overline{T}_S = (\overline{E}_k)^{-1}\sum_{i=k}^{N}\frac{\overline{E}_i^2}{\mu_i e_i} \quad . \tag{A10}$$