Proceedings of the 2008 Industrial Engineering Research Conference
J. Fowler and S. Mason, eds.

An Object-Oriented Framework for Analyzing VARIMETRIC Systems

Mehmet Miman, Manuel D. Rossetti, Vijith Varghese, Edward A. Pohl
Department of Industrial Engineering,
4207 Bell Engineering Center,
University of Arkansas, Fayetteville, Arkansas 72701, USA

Abstract

The VARIMETRIC model, discussed in Sherbrooke [8] and also in Muckstadt [4], has been used to analyze multi-echelon spare part networks for the military and other organizations for close to 40 years. This paper puts a "new face on an old friend" by providing an open-source, object-oriented framework for the performance analysis and optimization of VARIMETRIC like inventory systems. The object-oriented design, combined with the power of Java, provides users enormous flexibility to improve the current algorithms with minimal additional effort as well as implement their own optimization procedures. Currently, two Lagrangian based algorithms and a marginal analysis algorithm have been implemented. The capabilities of the framework as well as its design are presented in this paper. In addition, simplified examples illustrate how to use the framework for performance analysis as well as optimization. Since the software is open-source and object-oriented, it can readily be extended for advanced modeling situations.

Keywords
Varimetric, optimization, object oriented

1. Introduction

Repairable spare parts - those parts that can be economically repaired after a breakdown or a failure and used again - constitute an important portion of all assets of a typical organization (armed forces, manufacturing companies, civil airlines etc.) with remarkable amounts of monetary value. Since the late 1960s, the management of repairable items has received an extensive amount of attention in the literature. Interested readers can refer to [1] and [2] for a review of this area. This research builds on this literature by examining the optimal stock allocations among bases and depots subject to budgetary limits through an object-oriented VARIMETRIC package. The VARIMETRIC model, first introduced by Graves [3], assumes a compound Poisson failure process at the lower echelon with a continuous review one-for-one (s-1,s) policy, and suggests a two-parameter negative binomial distribution to fit the distribution of backorders at the bases in a two-echelon inventory system for a repairable item. The optimal allocation of stock within this context is a non-separable, non-convex, nonlinear discrete knapsack problem. The described open-source software implements three algorithms described in Muckstadt [4] to provide reasonable solutions within an object oriented design in Java. This package provides optimization of stock allocation of spare parts according to the VARIMETRIC model. This provides a readily available method to embed optimization within a larger simulation based optimization framework for evaluating forecasts and the analysis of other supply chain issues [5][6].

2. Background

This section provides enough details so that the paper becomes self-contained and one can better conceptualize the developed software. We begin with the basic notation, which also describes the primary inputs to the software.

λ_{ij} : demand rate for LRU i at base j (i=1..n, j=1..m)

r_{ij} : repair probability of item i at base j (i=1..n, j=1..m)

λ_{i0} : demand rate for LRU i at the depot; $\sum_j (1-r_{ij})\lambda_{ij}$

B_{ij} :base repair cycle time for LRU i at base j .

`

D_i : depot repair cycle time for LRU i .

s_{ij} :LRU i stock level at base j (or depot if j=0)

A_{ij} : average order and shipping time for item i, from depot to base j.

$B_D(s_{i0})$: expected outstanding depot backorders for LRU i given s_{i0}.

T_{ij} : effective lead time; average LRU i re-supply time at base j; $r_{ij}Bij + (1-r_{ij})(A_{ij} + B_D(s_{i0}) / \lambda_{i0})$

μ_{ij} : base pipeline mean; $\lambda_{ij}T_{ij}$

σ_{ij}^2 : base pipeline variance; $\mu_{ij} + \left((1-r_{ij})\lambda_{ij} / \lambda_{i0}\right)^2 \left(Var(N_D | s_{i0}) - \left(B_D(s_{i0})\right)\right)$

p_{ij} : probability of success for the fitted distribution for the pipeline of item i at base j; μ_{ij} / σ_{ij}

r_{ij} : number of success for the fitted distribution for the pipeline of item i at base j; $\mu_{ij} p_{ij} / (1-p_{ij})$

$\beta_{ij}(s_{i0}, s_{ij})$: The expected back orders for item i at Base j when the depot stock is s_{i0} and the stock at base j is s_{ij}.

θ : Langrangian multiplier for the relaxed problem.

$\theta_{\min}$: minimum value considered for θ .

$\theta_{\max}$: maximum value considered for θ .

$\alpha_i(s_i, s_{i0})$: total bases' expected back orders of LRU i, where depot stock level is s_{i0} and total stock in the system is s_i and remaining (s_i-s_{i0}) units are allocated according to greatest marginal benefit among bases.

$\hat{\alpha}_i(s_i)$: total bases' expected back orders of LRU i, where the total of s_i units are allocated among bases and the depot optimally resulted from the enumeration over the depot stock level; $\min_{s_{i0}} \alpha(s_i, s_{i0})$

$\hat{\alpha}_i^c(s_i)$: piecewise linear convex minorant function of $\hat{\alpha}_i(s_i)$

S^i : set of total stocks levels of s to be considered for LRU i.

$\hat{S}_c^i$: convex set of total stock levels corresponding to $\hat{\alpha}_i^c(s_i)$

s_i^k : k^{th} convex point in $\hat{S}_c^i$

c_i : unit cost of item i.

b : available budget

ε : allowable budget tolerance

Based on the above notation, the problem of interest can be described by a mathematical problem (P) as illustrated in Exhibit 1.

$$\begin{aligned}
&P: \\
&Min \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij}(s_{ij}, s_{i0}) \\
&s.t. \\
&\sum_{i=1}^{n} c_i (s_{i0} + \sum_{j=1}^{m} s_{ij}) \le b \\
&s_{ij} \ge 0, \text{integer } \forall_i, \forall_j
\end{aligned}$$

Exhibit 1: Mathematical Problem for Stock Allocation

Exhibits 2-4 summarize the steps of three algorithms- Lagrangian Iterative (LI), Lagrangian Enumerative(LE) and Marginal Analysis (MA) implemented within the software. The interested reader should refer to Muckstadt [4] for further details. Some implementation issues and some open questions in these algorithms raised by Muckstadt [4] as well as our approach to these issues are mentioned in the following section.

```
Step 1.  Determine $\theta_{\min}$ and $\theta_{\max}$, set iteration number $l = 1$ and $\theta_l = (\theta_{\min} + \theta_{\max})/2$
Step 2.  Set $\theta = \theta_l$ and for each item $i$, find depot and base stock levels, $s_{ij}(\theta)$, by solving relaxed sub-problem
         i.e, determine $\min_\rho \left\{W^i_{\rho l}\right\}$ where $W^i_{\rho l} = \min_{s_{ij}=0,1\ldots} \left[\sum_{j=1}^{m}\left(\beta_{ij}(s_{ij},\rho) + \theta_l c_i s_{ij}\right)\right] + \theta_l c_i \rho$
Step 3.  Evaluate the main constraint in $P$, i.e., compute $C(\theta) = \sum_{i=1}^{n} c_i \sum_{j=0}^{m} s_{ij}(\theta)$
         if $\left|C(\theta) - b\right| \le \varepsilon$, stop; otherwise, if $C(\theta) > b$ set $\theta_{\max} = \theta_l$; else set $\theta_{\min} = \theta_l$
         and set $l = l + 1$, $\theta_l = (\theta_{\min} + \theta_{\max})/2$ and return Step 2.
```

Exhibit 2: Lagrangian Iterative Algorithm

```
Step 1.  Determine $\theta_{\min}$ and $\theta_{\max}$, and $N$ lagrangian multiplier values such that
         $\theta_{\min} < \theta_1 < \theta_2 < \ldots < \theta_l \ldots < \theta_N < \theta_{\max}$
Step 2.  For each item $i$, for each $\theta_l$, find depot and base stock levels, $s_{ij}(\theta_l)$, by solving relaxed sub-problem
         i.e, determine $\min_\rho \left\{W^i_{\rho l}\right\}$
Step 3.  Compute $C(\theta_l)$ and select the solution that has cost closest to the $b$.
```

Exhibit 3: Lagrangian Enumerative Algorithm

```
Step 1.  Set $l = 1$, and for all $i$, determine $\alpha_i(s_i, s_{i0}), \hat{\alpha}_i(s_i), \hat{\alpha}^c_i(s_i)$, thus, $\hat{S}^i_c$ and compute $C(l) = \sum_i c_i \hat{\alpha}^c_i(s^l_i)$
Step 2.  For all $i$, compute the mariginal benefits as $\Delta_i(s^{k_i}_i) = \left\{\hat{\alpha}^c_i(s^{k_i}_i) - \hat{\alpha}^c_i(s^{k_i+1}_i)\right\} / \left\{c_i(s^{k_i+1}_i - s^{k_i}_i)\right\}$
Step 3.  Select $i^*$ with greatest marginal benefits, i.e. $\Delta_{i^*}(s^{k_{i^*}}_i) = \max_i \left\{\Delta_i(s^{k_i}_i)\right\}$ and set $C(l+1) = C(l) + c_i(s^{k_{i^*}+1}_i - s^{k_{i^*}}_i)$
         Increment $k_{i^*} = k_{i^*} + 1$ and $l = l + 1$, update only the marginal benefits of $i^*$ by recomputing $\Delta_{i^*}(s^{k_{i^*}}_i)$
Step 4.  if C($l$) $\le b + \varepsilon$, return Step 3.; otherwise stop
```

Exhibit 4: Marginal Analysis Algorithm

3. Object-oriented Design and Implementation

The object oriented modeling for implementation of the algorithms is motivated by the fact that for a given langrangian multiplier and depot stock level, the relaxed sub problems are separable by item and bases as well as by the fact that marginal analysis is performed over each item as summarized in Exhibit 4. To reflect these computational issues in the algorithms as well as to capture the VARIMETRIC model performances readily, such as expected backorders and stock out probabilities at each stock keeping location, we first developed a class *VariMetricModel* reflecting the VARIMETRIC system which consists of stocking units allocated to the depot and bases. Note that an item that is stocked at the depot does not necessarily need to be stored at each base. Instances of items at the two levels are represented by the instance of the classes *VMitem* and *VMBaseItem* classes respectively.

As illustrated in Figure 1, an instance of *VarieMetricModel*, that defines the two-echelon multi item spare parts system based on the parameters defined in the notation list, is the main input for all algorithms. It holds for instances of *VMItem*, each of which is a depot stock keeping unit that can also be stored at the bases. *VMItem* and *VMBaseItem* are subclass of *BaseStockItemAbstract* which implements *BaseStockItemIfc*. The interaction between objects between *VMItem* and *VMBaseItem* is illustrated in the class diagram shown in Figure 1. In brief, *BaseStockItemIfc,* promises to implement some performance functions related to bases such as expected backorders, probability of stock out, expected backorder waiting time and so on. *BaseStockItemAbstract* is a concrete implementation of this interface which has some of the important fields, mentioned in the notation list, for a stock keeping unit such as stocking level, lead time distribution, demand rate etc. It provides behaviors to access and modify the fields and implements the methods promised by the *VMBaseItemIfc.* Due to space limitations, we do not present a detailed discussion of all the attributes, methods, and behavior of these classes.

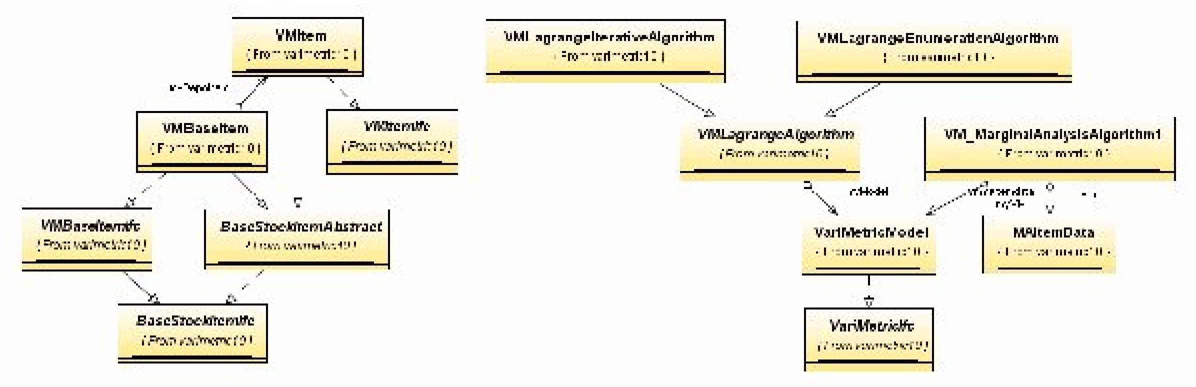

Figure 1: The Class Diagram of Varimetric Package

VMBaseItem provides lower echelon performance which can be aggregated through *VMItem* as explained next. It also provides the solution to the relaxed Lagrangian sub-problems by base given a depot stock level. As mentioned before *VMItem* refers to the overall LRU in the depot and has attributes and methods specific to the depot level. It contains instances of the associated *VMBaseItems* stored at bases (as illustrated in Exhibit 6) and provides the behaviors promised by *VMItemIfc* to compute the aggregated LRU-level performance such as total expected base backorders, total depot backorders, total base cost, total depot cost and total stocking cost while *VariMetricModel* further aggregates individual LRU results into over-all system level performance measures. That is, first the performance of individual LRUs at each base is computed, which is later aggregated through all bases to obtain the performance measures for each LRU, which are aggregated through all items to obtain the system level performance measures through *VMBaseItem*, *VMItem* and *VariMetricModel* respectively.

The basic advantage of *VariMetricModel* is to provide extensive statistics about the systems' performance. In addition, the main computations are repeated when the depot stock level is changed, i.e. the key driver for computations is the depot stock level for each item set by *VMItem*. This enables through the object-oriented structure other meta-heuristics, such as tabu search or genetic algorithms to be implemented easily depending on the user preference as post locally refining algorithms, which may improve solutions obtained from specialized algorithms mention in the following paragraphs.

Above algorithms, illustrated in Exhibits 2-4, are implemented, and hence can be executed, under object-oriented design using the same VariMetricModel that computes the characteristics of the two-echelon spare parts inventory systems. Both *VMLangrangianIterativeAlgorithm* and *VMLangrangianEnumerativeAlgorithm* are subclasses of the abstract *VMLangrangianAlgorithm* class, which provides fields and methods for both Lanrangian algorithms such as optimize(), searchThetaInterval(), evaluateConstraint() and optimizeLagrangeItemSubProblem(). In this design, for the item-level sub-problem, i.e. Step 2, in Exhibit 2 and 3, we set the depot stock level iteration limits to be within two standard deviation of the mean demand faced by the depot:

$$\rho \in \left[\max\{0, \lceil \lambda_{i0} D_i \rceil - 2\sqrt{\lceil \lambda_{i0} D_i \rceil}\}, \min\{\frac{b}{c_i}, \lceil \lambda_{i0} D_i \rceil + 2\sqrt{\lceil \lambda_{i0} D_i \rceil}\} \right] \tag{1}$$

Later each of the Lagrangian algorithms overrides the methods depending on the characteristics of the algorithm, for example, the theta search interval in LI is set according to constraint evaluation based on a bisection search, while in LE, the theta values are determined uniformly between its minimum and maximum values.

The MA algorithm transfers the *VariMetricModel* into *MAItemData* for each LRU, that contains fields and behaviors specific to marginal analysis implementation such as $\alpha_i(s_i, s_{i0})$, $\hat{\alpha}_i(s_i)$, $\hat{\alpha}_i^c(s_i)$, and $\hat{S}_c^i$. This *MAItemData* corresponds to each LRU in the system and computes the associated inputs. It also implements the convexification and next marginal benefit associated with each item. *VMMarginalAnlaysisAlgorithm* provides procedures that implement the steps described in Exhibit 4. The implementation allows for grid search over depot stock levels in the computation of $\hat{\alpha}_i(s_i)$ characterized by the incremental values provided as inputs for the algorithm.

As mentioned, the object oriented structure, combined with the power of Java, provides great flexibility for modifying the algorithms depending on the preferences and desired goals of specific implementations. For instance, without changing everything, one can easily plug in their own optimization procedure based on a meta-heuristic. This makes the research more distinguished from straight forward implementations of the optimization algorithms. In the following section, a sample case that tests the performance of each algorithm and demonstrates the ease of use of the objects is given.

4. Illustrative Example

To compare and illustrate the use of each algorithm, we provide simple test cases, whose optimal solution we can obtain through the total enumeration, which is very cumbersome and requires problem specific recursive formulation. The number of recursive loops in the total enumeration is $n(m+1)$ with total feasible solutions (lower bound) at the order of $\left(b/(\max\{c_i\})\right)^{n(m+1)}$.

Our test case consists of three LRUs, one depot and two bases as illustrated in Figure 2. Note that the dashed lines indicated failed LRUs while the shaded one represents the repaired items. Exhibit 5 provides the self explanatory code listing for how to use objects while the comparison of algorithms is provided in Table 1. All the algorithms are run with maximum number of iterations of 10. LI and MA provides the optimal allocation within the budget while the LE algorithm appears to be promising when the number of iterations increases. Overall, all of the algorithms implemented have plusses and minuses, which can be improved through the flexible design of the optimization package. These improvements are being explored as future research.

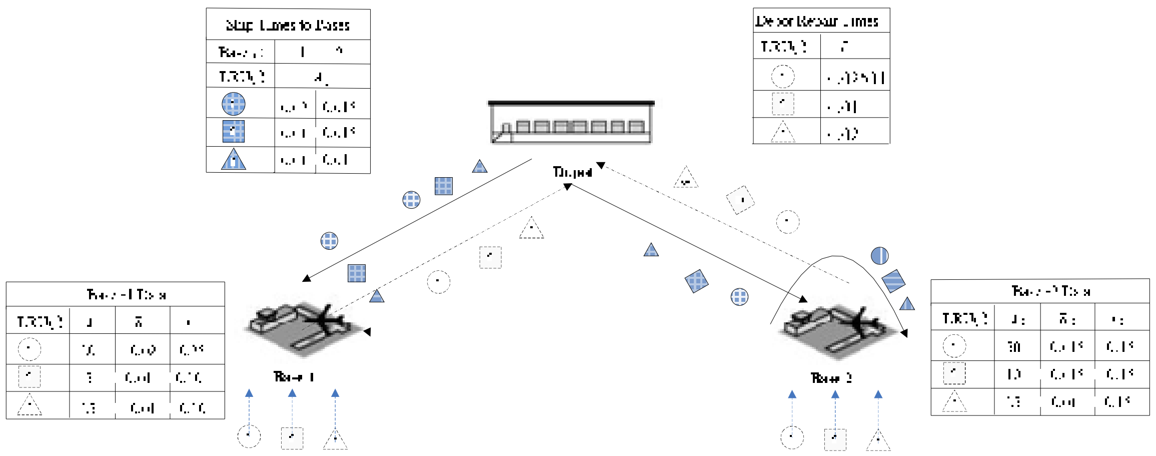

Figure 2: Illustrative Example

Table 1: Comparison of Algorithms for Illustrative Example

	s^*_{10}	s^*_{11}	s^*_{12}	s^*_{20}	s^*_{21}	s^*_{22}	s^*_{30}	s^*_{31}	s^*_{32}	Total Cost($)	Total Expected Base Backorders
LI	0	2	5	0	0	0	0	1	1	51	1.441180871
LE	0	2	5	0	0	1	0	1	1	54	1.266075189
MA	0	2	5	0	0	0	0	1	1	51	1.441180871

```
// construct the VARIMETRIC model
VariMetricModel vm = new VariMetricModel();  // build VARIMETRIC system
// construct LRU 1 at the depot where s10=0, c1=$5.00, D1=0.02531
VMItem di1 = new VMItem(1, 0, 5.0, 0.02531);
// add associated LRU1 in bases according to Figure 2:
di1.addBaseItem(20, 0.02, 0.25, 0.05, 0, 5.0);
di1.addBaseItem(50, 0.015, 0.15, 0.07, 0, 5.0);
vm.addVMItem(di1); // add item to the Varimetric Model
// (the other  LRUs can instantiated in a similar fashion where c2=$3.00, c5=$8.00)
// tell the model to optimize
double budget = 51.0;     //define budget
double minTheta = 0.01;   //user defined minTheta
double maxTheta = 0.125;  //user defined maxTheta
int maxIterations = 10;   //maximum number of iterations allowed
double budgetTolerance = 0.1; //budget tolerance
//define algorithm to use
VMLagrangeIterativeAlgorithm algo = new VMLagrangeIterativeAlgorithm(vm, budget,
maxIterations, budgetTolerance, minTheta, maxTheta);
boolean satisfied = algo.optimize(); //optimize the system using defined algorithm
```

Exhibit 5: Example Code for the use of Varimetric Package

5. Conclusions

In brief, although our development includes three currently existing algorithms: Lagrangian relaxation iterative procedure, enumerative procedure for Lagrangian multiplier and marginal analysis, the object oriented structure allows any user-defined algorithms to be plugged in. We have verified our package against the total enumeration and well-known cases in the literature. Overall, the package we developed provides an optimization tool for researchers as well as practitioners to utilize for the management of spare parts. As future research, we are combining this package with a package that enables the simulation of supply systems [5] in order to develop tools for updating policy parameters based on a variety of forecasting techniques. Further, we are also implementing similar performance analysis and optimization packages for a classical inventory systems (e.g. (r, Q)) rather than just for spare parts.

Acknowledgements

This material is based upon work supported by the U.S. Air Force Office of Sponsored Research and the Air Force Research Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

References

1. Guide, V.D.R., and Srivastava, R., 1997, “Repairable inventory theory: models and applications,” *European Journal of Operational Research,* 102, pp.1-20.
2. Kennedy, W.J., Patterson, J.W., and Fredendall L.D., 2002, “An overview of recent literature on spare parts inventories,” *International Journal of Production Economics*, 76, pp. 201-215.
3. Graves, S.C., 1985, “A multi-echelon inventory model for a repairable item with one-for-one replenishment,” *Management Science*, Vol.31, pp.1247-1256.
4. Muckstadt, J. A., 2005, *Analysis and Algorithms for Service Parts Supply Chains*, Springer Science+Media, Inc.
5. Rossetti, M., Miman, M., Varghese, V., and Xiang, Y. 2006. “An object-oriented framework for simulating multi-echelon inventory systems”, In *Proceedings of the 2006 Winter Simulation Conference*, L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds., Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
6. Rossetti, M. D., Varhese, V., Miman, M., and Pohl, E. 2007. “An Object-Oriented Framework for Simulating Inventory Systems with Forecast Driven Policy Optimization”, in preparation for the *Proceedings of the 2008 Winter Simulation Conference*.
7. Rossetti, M. D., 2007, "JSL: An Open-Source Object-Oriented Framework for Discrete-Event Simulation in Java," under review in *International Journal for Simulation and Process Modeling*
8. Sherbrooke, C. G. 1992. *Optimal Inventory Modeling of Systems: Multi-Echelon Techniques*, John-Wiley & Sons.